An Efficient Particle Swarm Optimizer for Multi-level Redundancy Allocation Problem

Zai Wang, Ke Tang and Xin Yao

***Abstract*— Redundancy allocation problem (RAP) has attracted much attention during the past thirty years due to its wide and valuable applications to improve the reliability in designing phase of various engineering systems. Up to now, most simulated systems in these attempts have focused on single-level systems whereas real world engineering systems always contain multiple levels where the entire system at the highest level, components at the lowest level, and subsystems locate at levels in between. Thus, it is desirable to study the redundancy allocation problem on multi-level systems, which is referred to as multi-level redundancy allocation problem (MLRAP). Now there are only two approaches to tackle with the MLRAP, however, when the complexity of MLRAP increases (i.e., the infeasible solution space becomes very small, or the employed simulation system becomes very large), both approaches cannot work well. In this paper, we proposed an efficient particle swarm optimizer (ePSO) to address the MLRAP on the basis of the hierarchical genotype representation. The ePSO is comprised of a new version of classical particle swarm optimizer which is tailored to MLRAP, a novel constraint-handling method and a problem-specific infeasible-solution-repairing technique. The experimental results demonstrated that ePSO significantly outperformed the latest two approaches on a new complex multi-level system.**

ACRONYMS

RAP	Redundancy Allocation Problem
MLRAP	Multi-Level Redundancy Allocation Problem
PSO	Particle Swarm Optimization
MA	Memetic 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 $U_{i,m}$

The authors are with the Nature Inspired Computation and Applications Laboratory, the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (emails: wangzai@mail.ustc.edu.cn, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).

Corresponding author: Ke Tang (Phone: +86–551–3600754).

I. INTRODUCTION

Redundancy allocation problem(RAP) is one of the most important reliability optimization problems regarding improving the reliability during the designing phase of real-world systems, such as electrical systems, mechanical system, software systems, etc [1], [2]. For now, the most commonly studied configurations of RAP are single-level systems, including the parallel-series systems, k-out-of-n:G(F) systems, general network systems and so on [1], [2]. However, in practice, the systems such as communication systems, computing systems, control systems and critical power systems [3], not only contain multiple levels, where the entire system is at the top level (system level), the components are at the bottom level, and subsystems locate at levels in between, but also allow allocating redundancy in each level. Fig. 1 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 [4]:

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

RAP is very important in reliability optimization of various systems. Also it is a well-known NP-hard problem [5]. Thus

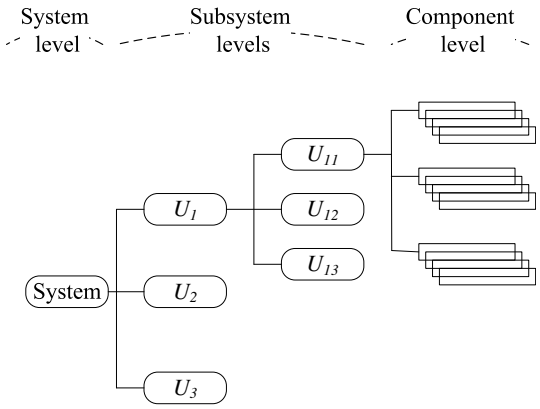

Fig. 1. The structure of a tri-level system

it is desirable to develop efficient optimization methods to address RAP. However, many classical mathematical methods have failed in coping with nonconvexities and nonsmoothness in RAP. As an alternative, the meta-heuristics, such as evolutionary algorithms [6], [7], ant colony optimization [8], artificial immune system [9], tabu search [10] and fuzzy system [11] have been widely and successfully employed in handling RAP. Comprehensive literature reviews on RAP have been carried out in [1], [2]. From these reviews, we can observe that the research on MLRAP lacks, which is inconsistent with the practical importance of MLRAP. In fact, now there have been only four papers on using meta-heuristics to address MLRAP [12], [13], [4], [14]. In [12], Levitin proposed an algorithm based on a genetic algorithm framework with a universal generating function technique for system survivability evaluation, to solve multi-level protection cost minimization problems subject to survivability constraint. Later, Yun and Kim [13] proposed a restricted multi-level redundancy allocation model, and addressed a tri-level redundancy allocation problem using a customized GA. However, to match the vector description of variables in customized GA, they made an additional constraint that the redundancy could be allocated to just one unit in a direct line, which is defined as a set of units from the system-level unit down to a component unit. In a direct line each unit is a child of its previous unit except the system-level unit. Though the targets of the work in [12], [13] are to cope with the MLRAP, there are too strong assumptions about the redundancy allocation model, which does not accord with the reality. Recently, Ranjan *et al.* [4] proposed a new multi-level redundancy allocation problem model and employed a hierarchical genetic algorithm as the problem solver. In [4], the redundancy allocation can be implemented onto every level of the multi-level system. Then a simple GA was adopted to tackle the problem, in which a solution was represented by a hierarchical genotype. In [14], Wang *et al.* proposed a memetic algorithm (MA) [15] for MLRAP. Based on the hierarchical genotype representation similar to that used in [4], they proposed the breadth-first-search crossover and mutation operators together with a novel problem-specific local search operator and incorporated them into the MA. Experimental results showed their MA outperformed the latest algorithms on two multi-level systems.

Though both algorithms in [4], [14] perform well on the two benchmark multi-level systems, their effectiveness on the MLRAP can still be improved significantly. First, the redundancy of the high-level units of the solutions obtained by these two algorithms are very low under the predefined cost constraints, which means that the solution space is not fully explored. Second, the two algorithms cannot work on more larger multi-level systems (i.e., with more than 4 levels). To overcome the above two drawbacks, in this paper, we propose an efficient particle swarm optimizer (ePSO) to address MLRAP. As a new optimization technique, particle swarm optimization (PSO) mimics a flock of birds which communicate with each other as they fly to find foods [16]. PSO was reported to outperform the conventional GAs with respect to solution quality, success rate and computational efficiency [17], [18]. Unlike the conventional GAs which are not really ergodic in practice due to multiple steps required [19], PSO can efficiently handle a wide svariety of real-world problems [20], [21], [22], [23]. Furthermore, in [18], PSO was showed to generally outperform a kind of MA as well. Therefore, we are going to devise a PSO for MLRAP and expect its performance to be better than GAs and MAs. In our work, based on the hierarchical genotype representation of the variables employed in [4], we first tailor a classic particle swarm optimizer to the problem, and then incorporate both a novel constraint handling method and a problem-specific infeasible-solution-repairing technique. To evaluate the performance of our ePSO on complex system, we further designed a five-level system (the largest is a four-level system for now) and compared our ePSO with the latest algorithms proposed in [4] and [14] on it. Experimental results showed that our approach outperformed the latest algorithms on the new employed system.

The rest of this paper is organized as follows. Section II introduces the problem formulation of MLRAP. Section III describes the performance degradation of existing approaches on more difficult MLRAPs. Our proposed ePSO is presented in Section IV, including the particle representation, the initialization of the population, the PSO operation designed for MLRAP, the constraint handling method, and the infeasible-solution-repairing operator. The simulation results and analysis are given in Section V. Section VI concludes this paper.

II. Problem Formulation

The configuration for MLRAP here is a multi-level serial system, which is comprised of multiple hierarchical levels. The system and the components are at the topmost and the lowest levels respectively, and subsystems are at the intermediate level. The system, subsystems and components are all referred to as units. Each unit except component has some serial next lowest level units. This unit is called the parent unit, and the lower level units are called child units. Fig. 1 presents a general multi-level serial system. As shown

in this figure, the system unit containing 3 serial child units (U_1 to U_3 at its next lowest level). This structure is replicated to all of the units expect the component units.

On basic multi-level serial configurations, redundancy can be allocated to the units at each level. Each unit except the component can have any number of subordinate units, and the relationships between these subordinate units may be in serial, in parallel or mixture of these two. Fig. 2 describes a sample of allocating redundancy on a bi-level serial system. The top part in Fig. 2 is the basic configuration, where U_1 is a unit at the system level and has two serial child units U_{11} and U_{12} at the next lowest level (component level). The middle part of Fig. 2 illustrates the redundancy allocation on the basic configuration. U_1^1 and U_1^2 are two redundant units of U_1. Similarly, U_{11} and U_{12} have 3 and 1 redundant units in parent unit U_1^1, and both have two redundant units in parent unit U_1^2, respectively. The bottom part describes the obtained system corresponding to the redundancy allocation scheme presented in the middle part.

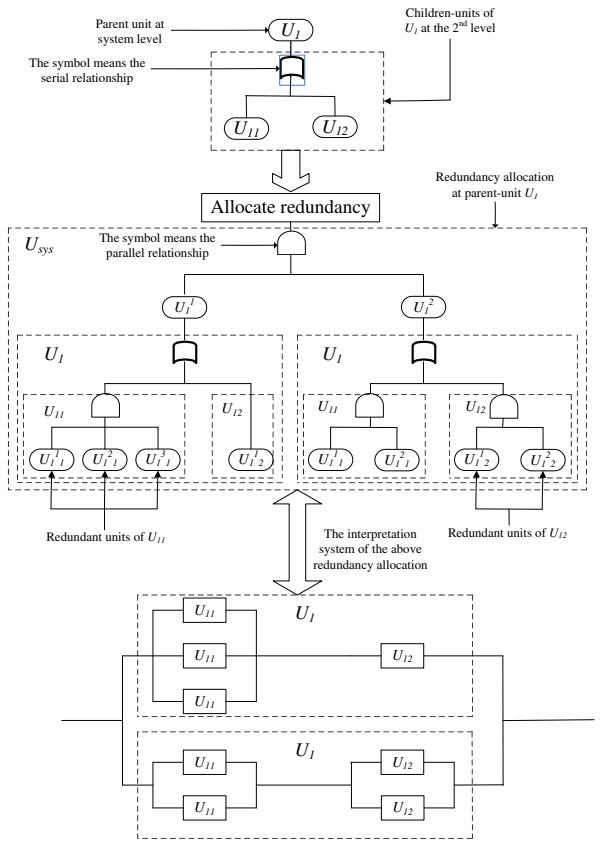

Fig. 2. An example of redundancy allocation on a bi-level model

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 Eq. (1):

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

where $R_{i,m}^j$ is the reliability of the jth redundant unit of $U_{i,m}$.

Using Eq. (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. 2 is:

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

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:

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

where $C_{i,m}^j$ is the cost of the jth redundant unit of the mth child unit of U_i and $\lambda_{i,m}$ is a constant predefined for $U_{i,m}$. For example, the cost of the final system in Fig. 2 is

$$\begin{aligned} C_{sys} = & \left(\sum_{j=1}^{3} C_{1,1}^j + \lambda_{1,1}{}^3 + C_{1,2}^1 + \lambda_{1,2}\right) + \\ & \left(\sum_{j=1}^{2} C_{1,1}^j + \lambda_{1,1}{}^2 + \sum_{j=1}^{2} C_{1,2}^j + \lambda_{1,2}{}^2\right) \end{aligned} \tag{4}$$

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

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

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

III. Performance Degradation of Existing Approaches on Harder MLRAPs

For MLRAP, there are two latest approaches: the hierarchical genetic algorithm (HGA) in [4], and the memetic algorithm (MA) in [14]. The performance of HGA and MA were evaluated on two multi-level systems (one with three levels and the other with four levels). Though the performance of HGA and MA are good on the two employed systems, we further want to know if these two algorithms can also work well on more complex MLRAPs. Because MLRAP is a constraint problem, so we could lift the restriction of the constraint to make the MLRAP more difficult. In our work, the cost constraint values are set in a smaller region (from 50 to 150 with the adjacent value 10) on the four-level system in [14]. The parameter settings are same as that in [14]. Each algorithm ran 30 times. The successful rate (the times one algorithm can find a feasible solution in 30 runs) of the two algorithms were recorded in Table I.

From Table I, it can be observed that HGA and MA cannot get the feasible solutions at most times in the 30 runs, which implies that the searching ability of both algorithms is not strong enough to reach the feasible solution space when the cost constraint is very strict. Because the solution space of the MLRAP is exponentially increased by the scale of the employed systems, so we can deduce that when the scale of the employed system become larger (more than 4 levels), HGA and MA may not work well within the acceptable constraint region, thus it is desirable to find more efficient approaches to tackle with MLRAPs. In our work, we proposed an eSPO to cope with the MLRAPs. The reason why the ePSO can solve the harder MLRAP has been analyzed in Section I. We will describe the details of ePSO in next section.

IV. Our Efficient PSO for MLRAP

A. Particle Swarm Optimization

In recent years, particle swarm optimization has been to be an effective approach to solving complex optimization problems. The PSO algorithm simulates the dynamics of a population of particles in a D-dimensional search space. Each particle represents a candidate solution to the optimization problem. A particle's flight is influenced by both the best position it has found thus far and the best position the whole population (or the neighborhood of this particle) has found so far. In a population of size N, the ith particle's position is denoted as $\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})$, the best position it has found is denoted as $\vec{p_i} = (p_{i1}, p_{i2}, \ldots, p_{iD})$ (we call it personal best position in this paper), the best position the whole population (or the current particle's neighborhood) has found is denoted as $\vec{p_g} = (p_{g1}, p_{g2}, \ldots, p_{gD})$, and the rate to change the position of this particle is called velocity and is denoted as $\vec{v_i} = (v_{i1}, v_{i2}, \ldots, v_{iD})$, where $i = 1, 2, \ldots, N$. Applying the constriction factor PSO [24], at each iteration step t, the ith particle updates its dth dimension of velocity according to Eq. 6 and position in the search space according to Eq. 7 as follows:

$$v_{id}(t) = \chi(v_{id}(t-1) + c_1 r_1(p_{id} - x_{id}(t-1)) + c_2 r_2(p_{gd} - x_{id}(t-1))) \quad (6)$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t), \quad (7)$$

where

$$\chi = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|}, c = c_1 + c_2, c > 4.0. \quad (8)$$

The constriction factor χ provides a damping effect on a particle's velocity and ensures the particle will converge over time. c_1 and c_2 are constants, typically 2.05, and thus, $\chi = 0.729844$ according to Eq. 8. r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$. Moreover, the velocity $\vec{v_i}$ can be constricted within the range $[-V_{MAX}, +V_{MAX}]$. In this way, the likelihood of a particle's flying out the search space is reduced. The value of $\pm V_{MAX}$ is usually set to be the lower and upper bounds of the allowed search ranges as suggested in [16].

B. Particle Representation

The PSO described above is usually used to optimize continuous problems. Since MLRAP is set in a search space featuring discrete distinctions between variables, in order to address it, some modifications must be made to the conventional PSO. For simplicity, we did not use the discrete PSO developed by Kennedy and Eberhart [25], in which a particle has its position to be represented by a binary variable, and its velocity to be the probability vector deducing the chance of taking the value one of each corresponding bit. Instead, we kept using the conventional PSO to operate on the real values. When evaluating a particle, we only considered the integer rounded by the real value. Based on the above modification, we propose an efficient PSO for MLRAP. In the next, we first introduce the designed particle representation for MLRAP. Then we present the initialization of the population. Finally, we describe the PSO operation for MLRAP based on our particle representation. Since the MLRAP is an optimization problem under a certain constraint, we will also present our constraint handling method and the proposed repair operator which is only applied to infeasible particles.

Conventional PSO usually uses vector genotype structures. However, for MLRAP, the decision variables have different lengths, since the redundancy can be allocated to each unit at each level. Therefore, the vector genotype representation is not suitable for MLRAP. To conform to the hierarchical relationship among the system, subsystems and components in the multi-level redundancy allocation model, the hierarchical genotype coding method proposed in [4] is the proper choice. The genotype coding representation used in [4] is an improvement of the coding method proposed in [26]. Based on this coding method, we can express every possible combination of multi-level redundancy allocation on the basic multi-level configurations.

TABLE I

THE PERFORMANCE OF HGA AND MA FOR MLRAP WITH SMALL CONSTRAINT VALUES ON THE FOUR-LEVEL SYSTEM EMPLOYED IN [14]

Cost constraint value	50	60	70	80	90	100	110	120	130	140	150
HGA	0	0	0	1	3	3	3	4	6	4	3
MA	0	0	0	0	3	2	2	6	4	7	4

In this paper, the hierarchical genotype representation in [4] was employed as the particle representation. Furthermore, in order to facilitate the conventional PSO operation, two additional parameters, velocity v_i and the real value r_{k_i} corresponding to the design variable k_i (the redundancy to each unit U_i), were incorporated into the solution representation. Hence, conventional PSO operation can be applied to v_i and r_{k_i}. Fitness evaluation is based on k_i which can be obtained by rounding r_{k_i}. The detailed description of the hierarchical representation can be observed in [4].

Fig. 3 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. v is the velocity of each unit, and r_k is the real value corresponding to the design variable k. A variable in the form of $x^e_{i,m}$ represents the redundancy allocated to the mth child unit of the eth redundant unit of U_i. In other words, the variables $x^e_{i,m}$'s are to be optimized in an MLRAP. Since the units at the lowest level (i.e., the component level) have no child unit, only the redundancy allocated to themselves are given.

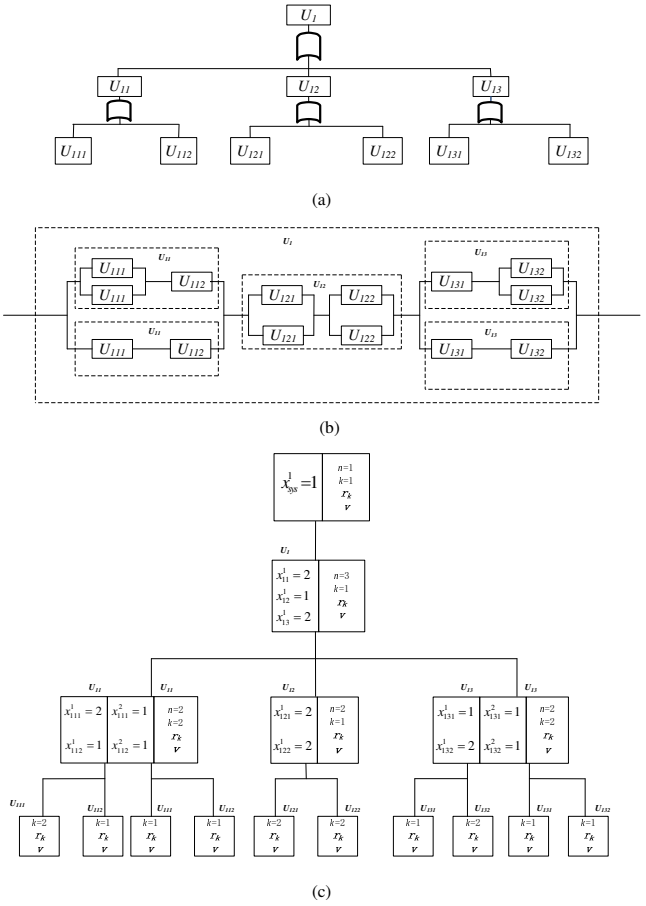

Fig. 3. An example of the hierarchical representation scheme on a tri-level serial system

C. Initial Population

A particle in the initial population is a hierarchical genotype tree representing a case of redundancy allocation on the basic multi-level configuration. The process of initializing a particle is also hierarchical. First we initialize the node at the highest level of the hierarchical genotype tree by randomly generating $n_i k_i$ integers for x values, so we can get the redundancy (k_i) of its child units at the lower level, and then we generate the x values of these child units. The value of r_{k_i} is made the same as k_i, and the value of v_i is randomly generated from the range $[-V_{MAX}, V_{MAX}]$, where V_{MAX} is set to be the distance from the lower bound to the upper bound of the allowed search range [16]. This procedure is repeated until the lowest level of the hierarchical tree is reached.

D. PSO Operation for MLRAP

We used the *gbest* topology in the proposed PSO. According to Eq. 6, the velocity of a particle is updated based on the difference between personal best position and global best position. Given the particle representation described above, we propose to update velocities of a particle's nodes in breadth-first order. The details are given below.

When a particle is to be updated, each node of it is checked against the corresponding nodes of the personal best particle and the global best particle. When a node of this particle is being checked, it is marked as "visited". If the corresponding values of the three k_is are the same, the checking procedure is applied to the next unvisited node according to the breadth-first order. Otherwise, the values of v_i and r_{k_i} are updated according to Eq. 6 and Eq. 7, and the value of k_i is obtained by rounding r_{k_i}. Based on the new value of k_i, there are two situations: (a) When the new value of k_i is same as old one, the parameters of this node including those of its subtree remain unchanged. (b) When the new value of k_i is different from the old one, the subtree of this node is reconstructed accordingly. The above checking and updating procedure is

then applied to every other unvisited nodes in breadth-first order.

E. Constraint Handling

As shown in Section II, the multi-level redundancy allocation problem is a constrained problem. The ability of handling constraints is one of the most important issues, which means that it is critical to guide the search around the feasible region. In literature, there are many useful constraints handling methods such as penalization techniques, repair techniques, separation techniques, and hybrid techniques [27], [28]. In our work, we conditionally employed a penalty function proposed by Gen and Cheng [27]. Concretely, the fitness function, $f(\boldsymbol{x})$, is mathematically expressed as

$$f(\boldsymbol{x}) = \begin{cases} R(\boldsymbol{x}) \times p(\boldsymbol{x}), \\ \quad if\ C(\boldsymbol{x}) \leq (1+\alpha) \times Cost\_Constraint\_Value \\ Cost\_Constraint\_Value - C(\boldsymbol{x}), \quad otherwise \end{cases} \tag{9}$$

where $R(\boldsymbol{x})$, $p(\boldsymbol{x})$, $C(\boldsymbol{x})$ and $\boldsymbol{x}$ are the system reliability, penalty function, cost function and a set of design variables respectively. The parameter α is used to control the pressure of the penalty on the population and its allowed value is within the range $[0, 1)$. Since some particles which only slightly violate the constraints may have potentials to fly into the promising area of the search space, we protect them via controlling the value of α. When $\alpha = 0$, the penalty function is the same as that proposed by Gen and Cheng [27].

F. Infeasible-solution-repairing Operator

As described above, some infeasible particles may have potentials to fly into the promising area of the search space, so at each generation, we applied a repairing operator to every infeasible particle before evaluating it according to Eq. 9. We describe the process of the repairing to an infeasible particle as follows.

We check every node of an infeasible particle from the top level to the bottom level. On each level, we repair the node having the highest value of k_i. If there are several nodes having the same highest values of k_is, we randomly select one to repair. We decrease the value of its k_i by one and reconstruct its subtree according to the new value of k_i. If the redundancy of a node (i.e. k_i) is equal to 1, it is marked as "checked". This process is repeating from the top level to the bottom level until any of the two conditions listed below is satisfied.

- A repair of a node makes the particle feasible.
- There are no unchecked nodes.

Note that the repairing operator may not necessarily pull an infeasible particle back into the feasible region, and it sometimes just pulls the particle back near the boundary between feasible and infeasible regions. At the same time, a particle that has been pulled back into the feasible region may fly into the infeasible region again. These features can make particles extensively search the areas near the boundary between feasible and infeasible regions, which is good for finding more optimal solutions.

G. Framework of the Algorithm ePSO for MLRAP

The algorithm ePSO for MLRAP works under the framework of the conventional PSO. It starts by initializing a swarm of hierarchical tree-based particles. At each iteration, all particles are evaluated according to Eq. 9, and then each particle's personal best position and the global best position of the swarm are updated. Based on the new information of the updated best positions, each particle's parameters are updated according to Eq. 6 and Eq. 7. Finally, the repairing operator is applied to every infeasible particle. The details of ePSO for MLRAP are shown in Algorithm 1, where t_{max} is the maximal allowed number of generations.

1: Initialize a swarm of particles based on the hierarchical tree-based genotype representation.
2: **while** $t < t_{max}$ **do**
3: Evaluate particles according to Eq. 9.
4: Update particles' personal best positions and the global best position of the swarm.
5: Update particles' parameters according to Eq. 6 and Eq. 7.
6: Apply the repairing operator to infeasible particles.
7: **end while**

Algorithm 1: The pseudo-code of ePSO for MLRAP

V. Empirical Studies

In this section, the performance of ePSO was evaluated on a new employed five-level redundancy allocation example. The results were compared with two latest algorithms, namely HGA in [4] and MA in [14]. The results indicate that ePSO significantly outperforms the two algorithms on complex systems.

A. Brief Descriptions of Studied Five-level Redundancy Allocation Problem

In our work, ePSO was evaluated on a new employed five-level redundancy allocation benchmark system. This problem is called *Problem-A*. The basic configuration of *Problem-A* is shown in Fig. 4, respectively. For each problem, the redundancy allocated to each unit is between one and five. The search space is huge, and the experimental design and results will be shown in the next two subsections.

B. Experimental Design of Problem-A

In our study, HGA, MA and ePSO were employed to solve *Problem-A*. For all the three algorithms, the maximum generation was 500. In HGA and MA, the population size was set to 100, crossover and mutation rate values were set to be 0.8 and 0.05, respectively. For MA, the number of child generated by breadth-first crossover and mutation

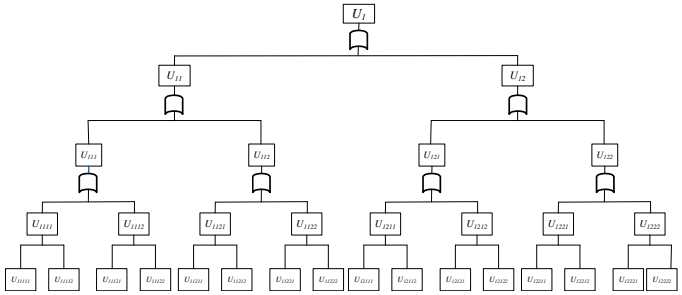

Fig. 4. The basic multi-level configuration of *Problem-A*

were 50 and 10, respectively. In our ePSO, the number of particles were set to be 50, the velocity was allowed within the range $[-V_{MAX}, V_{MAX}]$, where V_{MAX} was set to be 4 (i.e. the difference between the upper and the lower bounds [16]). Other PSO parameters (i.e. c_1, c_2 and χ) were set to be the same values as described in the Section III. The control parameter α of the penalty function was set to 0.3. The experiments were designed so as to investigate the following three issues.

1) A Single Case Study: To this point, the cost constraint value (C_0) was fixed to be 1500, and the input data of the basic system of *Problem-A* is shown in Table II. To observe the convergence of the optimal solutions obtained by the three algorithms, we ran all algorithms with the same set of initial population. 500 generations were allowed for each algorithm and the best solution obtained in each generation was recorded. In our experiment, we find that HGA and MA cannot work under this cost constraint, we just recorded the best solutions obtained by ePSO in each generation, and presented in Fig. 5, where x-axis represents the number of generations, and y-axis represents the highest system reliability obtained in each generation. Fig. 5 tells us that our ePSO not only can work on this problem, but also offers a good solution for the designers.

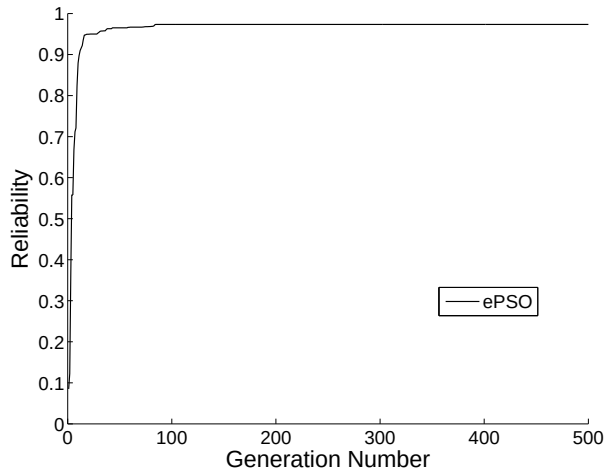

Fig. 5. The convergence of ePSO on *Problem-A*

2) Performance Over Different Constraint Values: To assess the influence of cost constraints on *Problem-A*, we varied the cost constraint values from 500 to 2400, with 20 values and the difference between two adjacent values was set to 100. Other input data for *Problem-A* were kept fixed. For each cost constraint value, we ran all algorithms 30 times and selected the best solution as the final solution. The obtained final solutions of the 20 cases are shown in Table III.

Furthermore, with the purpose to observe the statistical results obtained by the three algorithms, we calculated the means and variances of the reliability values of the thirty runs under each case study, which are shown in Table IV.

From Table III and Table IV, we find that ePSO obtained better or the same solutions compared with the other two algorithms under all the cost constraint values, which means that ePSO can offer reliable systems to complex system de-

TABLE II

THE INPUT DATA OF THE BASIC MULTI-LEVEL SYSTEM FOR *Problem-A*

Problem-A			
Unit	Reliability	Cost	λ
U_{11111}	0.60	5	2
U_{11112}	0.65	4	2
U_{11121}	0.50	5	3
U_{11122}	0.65	3	2
U_{11211}	0.70	6	2
U_{11212}	0.60	5	2
U_{11221}	0.65	4	3
U_{11222}	0.60	5	3
U_{12111}	0.60	4	2
U_{12112}	0.65	3	2
U_{12121}	0.55	5	3
U_{12122}	0.65	6	2
U_{12211}	0.65	6	2
U_{12212}	0.60	5	2
U_{12221}	0.65	4	2
U_{12222}	0.65	5	3

TABLE III

BEST SOLUTIONS OBTAINED BY HGA, MA AND EPSO WITH DIFFERENT CONSTRAINT VALUES ON *Problem-A* (THE BEST RESULT IS EMPHASIZED IN **BOLDFACE**, '*' MEANS THAT THE SOLUTION IS NOT A FEASIBLE SOLUTION)

Problem-A						
Cost Constraint	HGA		MA		ePSO	
	Reliability	Cost	Reliability	Cost	Reliability	Cost
500	0.990431*	3493*	0.994873*	3618*	**0.441363**	498
600	0.996017*	5432*	0.990432*	3843*	**0.568023**	598
700	0.994382*	4893*	0.995494*	3334*	**0.654334**	685
800	0.997439*	4388*	0.994329*	4339*	**0.716695**	700
900	0.994343*	3552*	0.997438*	5342*	**0.823558**	798
1000	0.998933*	3932*	0.999032*	4449*	**0.928021**	999
1100	0.993096*	3843*	0.99743*	5003*	**0.927118**	1100
1200	0.996438*	3543*	0.994321*	5480*	**0.950805**	1199
1300	0.994393*	3753*	0.998549*	4839*	**0.950543**	1300
1400	0.999032*	6032*	0.997899*	4858*	**0.969083**	1395
1500	0.994332*	4324*	0.998931*	3954*	**0.973356**	1500
1600	0.997438*	2988*	0.997332*	3332*	**0.975745**	1593
1700	0.998439*	3694*	0.998439*	5099*	**0.98549**	1698
1800	0.996643*	3211*	0.997438*	4873*	**0.990503**	1800
1900	0.998329*	3593*	0.994822*	3289*	**0.9914**	1900
2000	0.999322*	5432*	0.995342*	2998*	**0.993184**	1999
2100	0.998432*	3732*	0.998329*	3849*	**0.995652**	2099
2200	0.998432*	3192*	0.996481*	4390*	**0.997251**	2196
2300	0.990329*	4921*	0.992314*	3698*	**0.99769**	2298
2400	0.999423*	5321*	0.990439*	3437*	**0.999477**	2391

TABLE IV

STATISTICAL RESULTS OBTAINED BY HGA, MA AND EPSO UNDER DIFFERENT CONSTRAINT VALUES ON *Problem-A* (THE BEST MEAN RESULT IS EMPHASIZED IN **BOLDFACE**),'–' MEANS THAT THE SOLUTIONS OF THIS ALGORITHM ARE INFEASIBLE, AND THE STATISTICAL RESULTS OF THESE INFEASIBLE SOLUTIONS ARE MEANINGLESS.

Problem-A						
Cost	HGA		MA		ePSO	
Constraint	Mean	Variance	Mean	Variance	Mean	Variance
500	–	–	–	–	**0.322608**	1.57e-002
600	–	–	–	–	**0.43065**	2.40e-002
700	–	–	–	–	**0.53654**	2.05e-002
800	–	–	–	–	**0.670659**	2.46e-002
900	–	–	–	–	**0.751346**	1.91e-002
1000	–	–	–	–	**0.854941**	2.64e-003
1100	–	–	–	–	**0.883308**	7.70e-003
1200	–	–	–	–	**0.936425**	5.65e-004
1300	–	–	–	–	**0.951189**	3.62e-004
1400	–	–	–	–	**0.96081**	2.30e-004
1500	–	–	–	–	**0.971923**	2.45e-004
1600	–	–	–	–	**0.976328**	7.93e-005
1700	–	–	–	–	**0.981693**	5.71e-005
1800	–	–	–	–	**0.987784**	3.78e-005
1900	–	–	–	–	**0.990569**	1.84e-005
2000	–	–	–	–	**0.991662**	1.54e-006
2100	–	–	–	–	**0.99378**	4.31e-006
2200	–	–	–	–	**0.9959**	2.20e-006
2300	–	–	–	–	**0.996743**	5.59e-006
2400	–	–	–	–	**0.998217**	2.15e-006

signers. Also, from the statistical results shown in Table IV, it can be observed that ePSO performed significantly better than the other two algorithms in all the cases.

3) Comparison of Three Algorithms Using Different System Parameters: To examine the robustness of ePSO on *Problem-A*, we varied the component unit reliability and kept other input data unchanged in basic multi-level system. For each component unit, the reliability value was randomly selected from four choices $(0.80, 0.85, 0.90, 0.95)$. We randomly utilized ten cases. For each case study, we ran all the three algorithms for 30 times and selected the best solutions, which are shown in Table V. We also did the nonparametric Wilcoxon rank sum tests to do statistical analysis, and the results are shown in Table VI. From Table V and Table VI, we can observe that the solutions obtained by ePSO are better than solutions obtained by HGA and MA, i.e., our ePSO is the most robust within the three algorithms.

VI. CONCLUSIONS

In literature, the RAP has been intensively investigated on single-level system, while the practical systems are usually with multiple levels. However, only limited studies have been conducted in this scenario. In this paper, the redundancy allocation problems for systems with multiple levels were investigated, and an efficient PSO was proposed to cope with this type of problems. In our ePSO, we tailored a kind of classic PSO to the MLRAP based on the hierarchical genotype representation [4], proposed a controllable constraint handling method and an infeasible-solution-repairing operator only applied to infeasible particles. Based upon our experimental studies on two examples, two main conclusions can be drawn. First, ePSO is capable of achieving better

TABLE V

BEST SOLUTIONS OBTAINED BY HGA, MA AND EPSO WITH DIFFERENT INPUT DATA ON *Problem-A* (THE BEST RESULT IS EMPHASIZED IN **BOLDFACE**, '*' MEANS THAT THE ALGORITHM CANNOT GET A FEASIBLE SOLUTION)

Problem-A						
Cost	HGA		MA		ePSO	
Constraint	Reliability	Cost	Reliability	Cost	Reliability	Cost
1	*	*	*	*	**0.968345**	1495
2	*	*	*	*	**0.979798**	1497
3	*	*	*	*	**0.939239**	1491
4	*	*	*	*	**0.973383**	1497
5	*	*	*	*	**0.95953**	1499
6	*	*	*	*	**0.973331**	1499
7	*	*	*	*	**0.966794**	1498
8	*	*	*	*	**0.994318**	1495
9	*	*	*	*	**0.925553**	1498
10	*	*	*	*	**0.968148**	1591

TABLE VI

STATISTICAL RESULTS OBTAINED BY HGA, MA AND EPSO WITH DIFFERENT INPUT DATA ON *Problem-A* (THE BEST MEAN RESULT IS EMPHASIZED IN **BOLDFACE**),'–' MEANS THAT THE SOLUTIONS OF THIS ALGORITHM ARE INFEASIBLE, AND THE STATISTICAL RESULTS OF THESE INFEASIBLE SOLUTIONS ARE MEANINGLESS.

Problem-A						
Cost	HGA		MA		ePSO	
Constraint	Mean	Variance	Mean	Variance	Mean	Variance
1	–	–	–	–	**0.958548**	2.35e-3
2	–	–	–	–	**0.959903**	4.86e-3
3	–	–	–	–	**0.92555**	5.44e-3
4	–	–	–	–	**0.965802**	3.65e-3
5	–	–	–	–	**0.950022**	2.77e-3
6	–	–	–	–	**0.967373**	3.81e-3
7	–	–	–	–	**0.959549**	5.37e-3
8	–	–	–	–	**0.988382**	4.66e-3
9	–	–	–	–	**0.913482**	3.84e-3
10	–	–	–	–	**0.960116**	4.85e-3

solutions than the state-of-the-art algorithms. ePSO is a very promising approach to MLRAP and deserves more in-depth investigation. Second, the robustness of ePSO is good, according to the experimental studies on MLRAP with different input data.

Two issues deserve further discussion and investigation. First, in experimental study, we only considered systems with hierarchical serial structures. However, changing the structure of system will result in different reliability and cost functions of the corresponding optimization problem. Since our ePSO has been shown to be effective on this hierarchical serial structure, we plan to extend it to hierarchical systems with other types of structures, such as hierarchical parallel-serial structure and hierarchical networks. Second, the formulated multi-level redundancy allocation problem is just a single-objective problem with the only goal to maximize the system reliability, while the designing cost is considered as a constraint. However, as the size of the systems kept growing in the past decades, especially for multi-level systems, it is unrealistic to overlook either the reliability or the designing cost. Unfortunately, given a budget of redundancy degree, more designing cost is usually inevitable if we want to improve the reliability of the system. Hence, it is impossible

that a single solution is optimal in terms of both reliability and designing cost. Instead, a multi-objective problem can be formulated, then a tradeoff between the system reliability and designing cost can be obtained through the problem-solving process of these multi-objective problems. We will investigate these two issues in our future work.

ACKNOWLEDGEMENT

This work was supported by IEEE Walter Karplus Summer Research Grant, an EPSRC grant (No. EP/D052785/1) on "SEBASE: Software Engineering By Automated SEarch", a National Natural Science Foundation of China grant (No. U0835002), and the Fund for International Joint Research Program of Anhui Science & Technology Department (No. 08080703016).

REFERENCES

[1] W. Kuo and V. Prasad, "An Annotated Overview of System-reliability Optimization," *IEEE Transactions on Reliability*, vol. 49, no. 2, pp. 176–187, 2000.

[2] W. Kuo and R. Wang, "Recent Advances in Optimal Reliability Allocation," *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, vol. 37, no. 2, pp. 143–156, 2007.

[3] W. Wang, N. J. Loman, and P. Vassiliou, "Reliability Importance of Components in a Complex System," Los Angeles, California, USA, January 26-29, 2004, pp. 6–11.

[4] K. Ranjan, K. Izui, M. Yoshimura, and N. Shinji, "Multilevel Redundancy Allocation Optimization Using Hierarchical Genetic Algorithm," *IEEE Transactions on Reliability*, vol. 57, no. 4, pp. 650–661, 2008.

[5] 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.

[6] D. W. Coit and A. E. Smith, "Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm," *IEEE Transactions on Reliability*, vol. 45, no. 2, pp. 254–260, 1996.

[7] ——, "Genetic Algorithm to Maximize a Lowerbound for System Time-to-failure with Uncertain Component Weibull Parameters," *Computers and Industrial Engineering*, vol. 41, no. 4, pp. 423–440, 2002.

[8] Y. C. Liang and A. E. Smith, "An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem," *IEEE Transactions on Reliability*, vol. 53, no. 3, pp. 417–423, 2004.

[9] T. C. Chen, "IAs Based Approach for Reliability Redundancy Allocation Problems," *Applied Mathematics and Computation*, vol. 182, no. 2, pp. 1556–1567, 2006.

[10] A. E. S. S. K. Konak and D. W. Coit, "Efficiently Solving the Redundancy Allocation Problem Using Tabu Seach," *IIE Transactions*, vol. 35, no. 6, pp. 515–526, 2003.

[11] G. S. Mahapatra and T. K. Roy, "Fuzzy Multi-objective Mathematical Programming on Rliability Optimization Model," *Applied Mathematics and Computation*, vol. 174, no. 1, pp. 643–659, 2006.

[12] G. Levitin, "Optimal Multilevel Protection in Serial-Parallel Systems," *Reliability Engineering and System Safety*, vol. 81, no. 1, pp. 93–102, 2003.

[13] W. Y. Yun and J. W. kim, "Multilevel Redundancy Optimization in Series Systems," *Computers and Industrial Engineering*, vol. 46, pp. 337–346, 2004.

[14] Z. Wang, K. Tang, and X. Yao, "A Memetic Algorithm to Multi-level Reudandancy Allocation Problem," *Submited to IEEE Transcation on Reliability*, 2009.

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

[16] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Fransisco, CA, US: Morgan Kaufmann, 2001.

[17] O. D. W. R. Hassan, B. Cohanim and G. Venter, "A Comparison of Particle Swarm Optimization and the Genetic Algorithm," in *Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference*, 2005.

[18] T. H. E. Elbeltagi and D. Grierson, "Comparison among Five Evolutionary-based Optimization Algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[19] R. C. Eberhart and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization," *Lecture notes in computer science*, pp. 611–618, 1998.

[20] ——, "Particle Swarm Optimization: Developments, Applications and Resources," in *Proceedings of the 2001 congress on evolutionary computation*, vol. 1. Piscataway, NJ, USA: IEEE, 2001, pp. 81–86.

[21] I. A. A. Salman and S. Al-Madani, "Particle Swarm Optimization for Task Assignment Problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[22] P. S. S. Kannan, S. M. R. Slochanal and N. P. Padhy, "Application of Particle Swarm Optimization Technique and Its Variants to Generation Expansion Planning Problem," *Electric Power Systems Research*, vol. 70, no. 3, pp. 203–210, 2004.

[23] S. M. J. H. Y. del Valle, G. K. Venayagamoorthy and R. G. Harley, "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[24] M. Clerc and J. Kennedy, "The particle swarm – explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[25] J. Kennedy and R. C. Eberhart, "A Discrete Binary Version of the Particle Swarm Algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics, 'Computational Cybernetics and Simulation'.*, vol. 5, 1997.

[26] M. Yoshimura and K. Izui, "Smart Optimization of Machine Systems Using Hierarchical Genotype Representations," *ASME Journal of Mechanical Design*, vol. 124, no. 3, pp. 373–384, 2002.

[27] 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.

[28] 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.