Expert Systems with Applications 36 (2009) 9192–9200

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems

Wei-Chang Yeh *

e-Integration and Collaboration Laboratory, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, P.O. Box 24-60, Hsinchu 300, Taiwan, ROC

ARTICLE INFO

Keywords:
Reliability
Redundancy allocation problem (RAP)
Particle swarm optimization algorithm (PSO)
Series systems

ABSTRACT

Nowadays, the redundancy allocation problem (RAP) is increasingly becoming an important tool in the initial stages of or prior to planning, designing, and control of systems. The multiple multi-level redundancy allocation problem (MMRAP) is an extension of the traditional RAP such that all available items for redundancy (system, module and component) can be simultaneously chosen. In this paper, a novel particle swarm optimization algorithm (PSO) called the two-stage discrete PSO (2DPSO) is presented to solve MMRAP in series systems such that some subsystems or modules consist of different components in series. To the best of our knowledge, this is the first attempt to use a PSO to MMRAP. The proposed PSO used a totally new, very simple, effective and efficient mechanism to move to the next position without velocity. The result obtained by 2DPSO has been compared with those obtained by genetic algorithm (GA) and binary PSO (BPSO). Computational results show that the proposed 2DPSO is very competitive and performs well in the number of times it finds the best solutions, the average numbers of the earliest finding of the best solutions, and computation times.

1. Introduction

Considerable effort has been expended during the last two decades to develop reliability techniques and criteria to measure the quality level for generation, transmission, and distribution of composite systems (Kuo & Prasad, 2000; Kuo, Prasad, Tilman, & Hwang, 2001; Kuo and Wan, 2007; Ravi & Reddy, 2000).

The optimal reliability design is to determine a system structure which has the highest reliability for the minimum cost of manufacturing. Three kinds of reliability optimization problems are discussed in the literature based on the types of their decision variables: reliability allocation, redundancy allocation, and reliability-redundancy allocation (Kuo & Prasad, 2000; Kuo and Wan, 2007; Kuo et al., 2001; Ravi & Reddy, 2000). A general formulation of the reliability optimization problem can be formulated as the following integer nonlinear programming problem:

$$\text{Maximize } f(\mathbf{r},\mathbf{x}) \tag{1}$$

$$s.t.\ \ g_i(\mathbf{r},\mathbf{x}) \leqslant b_i \text{ for } i = 1,2,\ldots,m \tag{2}$$

$$l_j \leqslant x_j \leqslant u_j, x_j \text{ is a positive integer number for } j = 1,2,\ldots,n \tag{3}$$

$$rj \in (0,1) \text{ for } j = 1,2,\ldots,n \tag{4}$$

where n and m are the number of components and the number of constraints, respectively; r_j is the component reliability of the jth component; x_j is the number of identical redundant components, i.e. the number of redundancies, at the jth component; $f(\bullet)$ is an objective function of the problem; g_i is the ith constraint function; and b_i is the maximum allowable amount of the ith resource.

If component reliabilities, r_js for all j, are the only variables, the problem is called reliability allocation problem; if the number of redundancies, x_js for all j, is the only variable, the problem becomes redundancy allocation problem (RAP); and if the decision variables of the problem include both the component reliabilities and redundancies, the problem is called a reliability-redundancy allocation problem.

The redundancy allocation problem (RAP) involves setting reliability objectives for components or subsystems in order to meet the resource consumption constraint, e.g. the total cost. It was proven to be a #*P*-hard problem (Chern, 1992) with computational effort growing exponentially with the number of nodes and links in the system. To avoid numerical difficulties and reduce the computational burden, efforts have been devoted in finding high-quality solutions in a reasonable computational time by heuristic optimization techniques instead of finding an optimal solution. To achieve a better solution quality, modern meta-heuristics have been presented for the reliability optimization problems (Onishi & Kimura, 2007) such as the response surface methodology (Yeh, 2003), simulated annealing

* Tel.: +886 3 572 2204.
E-mail address: yeh@ieee.org

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.12.024

(Kim, Bae, & Park, 2006), tabu search (Bland, 1998), genetic algorithm (Hsieh, Chen, & Bricker, 1998; Yun, Song, & Kim, 2007), immune algorithm (Chen, 2006), ant colony optimization (Liang & Smith, 2004), and iterated local search (Agarwal & Gupta, 2005; Ha & Kuo, 2006; Hsieh, 2002; Liang & Chen, 2007; Yun & Kim, 2004).

A system can be functionally decomposed into system, module (composing the system) and component (composing the module) levels in a reliability logic-diagram sense. An available unit for redundancy can be a system, module or component. For example, as shown in Fig. 1, each square node is called a component, each branch within the red dashed-line box is called a module, and the whole tree is a system.

The object of RAP is limited to the component redundancy due to a well-known principle that the component redundancy is more effective than the system redundancy. In recent studies, component redundancy allocation problems (RAPs) are mainly considered because it is more difficult to improve the component reliability. Examples include electrical and telecommunication networks, web structures, and airline routes (Yun & Kim, 2004; Yun et al., 2007).

Yun and Kim (2004) considered the modula redundancy under a strong restriction that only one level among component, module and system can be a candidate for redundancy in RAP. In this new kind of RAP (called the multi-state RAP, MRAP in short), failures of all the available units are assumed to be statistically independent. Yun et al. (2007) extended MRAP to the multiple MRAP (MMRAP) by relaxing the strong restriction such that all available units for redundancy can be chosen simultaneously. Therefore, the function of an available unit can be replaced by the function of its entire series of child nodes. For example, there are twelve feasible structures (Fig. 2a–l) if node S is failed, and nodes A and B are functioned in Fig. 1.

Duplicating a designed module requires less time and skill than duplicating systems and/or components composing the module in the modular system (Yun & Kim, 2004; Yun et al., 2007). Hence, modula redundancy can be more effective and easier than the component redundancy and/or system redundancy. Therefore, we only focus on MMRAP this paper.

Yun et al. (2007) also present a simple GA (SGA) to solve MMRAP. In their SGA, there is no information about how they deal with the cost constraint and their solution representation may cause a lot of unfeasible structures. The need for a simple method for this special MMRAP, which is used to maximize system reliability and meet the cost constraint, thus arises.

In this study, we proposed a two-stage discrete particle swarm optimization algorithm (2DPSO) to solve MMRAP. The paper is organized as follows: In Section 2, the acronym, notations and assumptions are described. In Section 3, the detailed information of PSO is given. In Section 4, the methodology of the proposed 2DPSO is discussed. In Section 5 the solving of MMRAP using the proposed 2DPSO through one numerical benchmark example is illustrated. Finally, in Section 6, the concluding remarks are summarized.

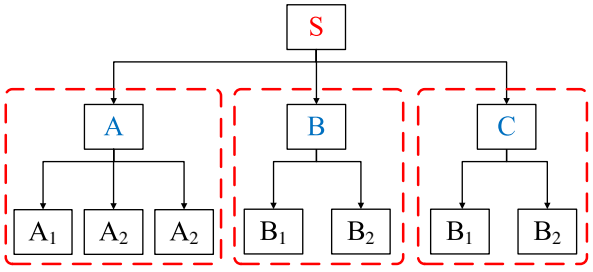

Fig. 1. A system with eleven items.

2. Acronym, notations and assumptions

2.1. Acronym

RAP	redundancy allocation problem
MRAP	multi-state RAP
MMRAP	multi-level MRAP
PSO	particle swarm optimization algorithm
DPSO	discrete particle swarm optimization algorithm
2DPSO	two-stage discrete particle swarm optimization algorithm

2.2. Notations

n	the number of dimensions in the proposed 2DPSO
N	the number of particles in the proposed 2DPSO in each replication.
M	the total number of independent replications in the proposed 2DPSO
r_i	the reliability of the ith type of components
$X(\bullet)$, x_i	the number of redundancies of the component $\bullet$ and the ith type of components, respectively
Child($\bullet$)	the set of all child-nodes of node (component) $\bullet$ in MMRAPs. For example, Child(S) = {A, B, C}
$T(\bullet)$	$T(\bullet) = 1$ if $X(\bullet) > 0$, or $X(\bullet) = 0$ and $T(\bullet_s) = 1$ for all $\bullet_s \in$ Child($\bullet$). Otherwise, $T(\bullet) = 0$. For example, $T(S) = 1$ since $T(A) = T(B) = T(C) = 1$ in Fig. 2 even $X(S) = 0$
ρ_1, ρ_2	the random numbers uniformly distributed in [0,1]
c_1, c_2	the cognition learning factor and the social learning factor, respectively
w	the inertia weight
x_{ij}^t	the value of the jth dimension of the position of particle i at iteration t, where $i = 1,2,\ldots,N$, $j = 1,2,\ldots,n$, and $t = 1,2,\ldots,M$
X_i^t	$X_i^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t)$ is the position of particle i at iteration t, where $i = 1,2,\ldots,N$ and $t = 1,2,\ldots,M$
v_{ij}^t	the value of the jth dimension of the velocity of particle i at iteration t, where $i = 1,2,\ldots,N$, $j = 1,2,\ldots,n$, and $t = 1,2,\ldots,M$
V_i^t	$V_i^t = (v_{i1}^t, v_{i2}^t, \ldots, v_{in}^t)$ is the velocity of particle i at iteration t, where $i = 1,2,\ldots,N$ and $t = 1,2,\ldots,M$
P_i^t	$P_i^t = (p_{i1}^t, p_{i2}^t, \ldots, p_{in}^t)$ is the best solution of particle i so far until iteration t, i.e. *pbest*, where $i = 1,2,\ldots,N$ and $t = 1,2,\ldots,M$
G^t	$G^t = (g_1^t, g_2^t, \ldots, g_n^t)$ is the best solution among $P_1^t, P_2^t, \ldots, P_N^t$ at iteration t, i.e. *gbest* where $i = 1,2,\ldots,N$ and $t = 1,2,\ldots,M$
$R(X_i^t)$	the reliability of X_i^t, where $i = 1,2,\ldots,N$ and $t = 1,2,\ldots,M$
$F(X_i^t)$	the fitness function value of X_i^t, where $i = 1,2,\ldots,N$ and $t = 1,2,\ldots,M$
C_{UB}	the number of the required budget
$C(\bullet)$	the total cost of $\bullet$. If $\bullet$ represents the number of the ith type of components, i.e. x_i, then $C(x_i) = c_i x_i + \lambda_i^{x_i}$, where c_i and λ_i are two given constants
$C_U(\bullet)$	the upper-bound of the total cost of $\bullet$
$C_L(\bullet)$	the lower-bound of the total cost of the subtree structure with root component $\bullet$ if $T(\bullet) = 1$
$U(\bullet)$, $L(\bullet)$	the upper and lower bounds for $\bullet$, respectively

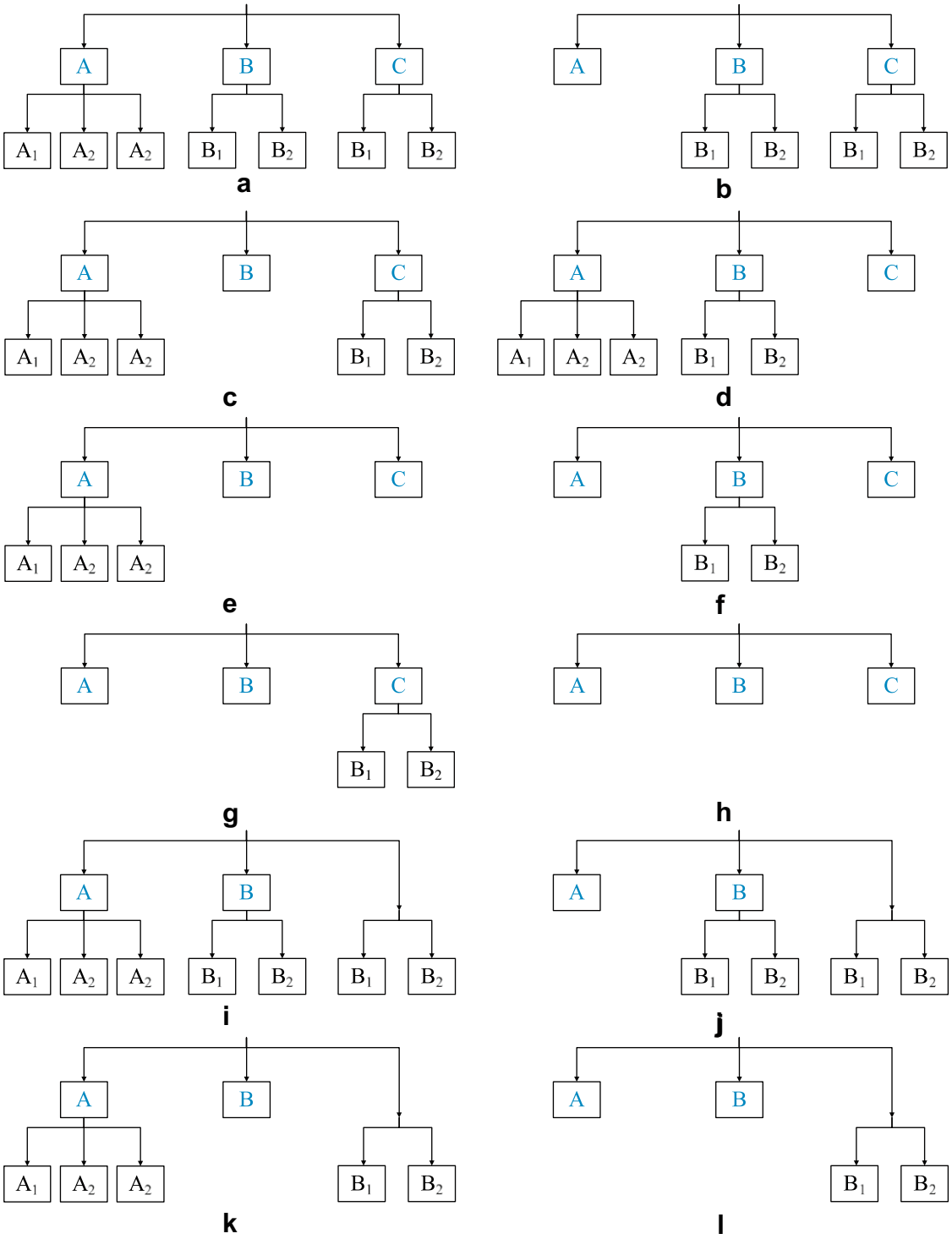

Fig. 2. All feasible structures while S is failed, and A and B are functioned in Fig. 1.

2.3. Assumption (Yun et al., 2007)

(1) The combination of the items for redundancy should satisfy the function at the system level. If an item is used, all its sibling items should be used or its function should be satisfied by the corresponding child items.
(2) The items for redundancy should be used in parallel at one combination.

3. Particle swarm optimization

The particle swarm optimization (PSO) technique is a population based stochastic optimization technique first introduced by Kennedy and Eberhard (1995). It belongs to the category of Swarm Intelligence methods; it is also an evolutionary computation method inspired by the metaphor of social interaction and communication such as bird flocking and fish schooling. The details will be given in the following sections.

In PSO, a solution is encoded as a finite-length string called a particle (Allahverdi & Al-Anzi, 2006; Eberhart and Shi, 2001; Kennedy and Eberhard, 1995; Kennedy, Eberhard, & Shi, 2001; Kennedy and Eberhart, 1997; Moore & Chapman, 1999; Sousa, Silva, & Silva, 2004; Trelea, 2003). All the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles (Allahverdi & Al-Anzi, 2006; Eberhart and Shi, 2001; Kennedy and Eberhard, 1995; Kennedy and Eberhart, 1997; Kennedy et al., 2001; Moore & Chapman, 1999; Sousa et al., 2004; Trelea, 2003). PSO is initialized with a population of random particles with random positions and velocities inside the problem space, and then searches for optima by updating generations. It combines local search and global search yielding in high search efficiency. Each particle moves towards its best previous position and towards the best particle in the whole swarm in every iteration. The former is a local best and its value is called *pbest*, and the latter is a global best and its value is called *gbest* in the literature (Allahverdi & Al-Anzi, 2006; Eberhart and Shi, 2001; Kennedy and Eberhard, 1995; Kennedy and Eberhart, 1997; Kennedy et al., 2001; Moore & Chapman, 1999; Sousa et al., 2004; Trelea, 2003). After finding the two best values, the particle updates its velocity and position with the following equation in continuous PSO:

$$v_{ij}^t = w \cdot v_{ij}^{t-1} + c_1 \cdot \rho_1 \cdot (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 \cdot \rho_2 \cdot (g_j^{t-1} - x_{ij}^{t-1}) \quad (5)$$

$$x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t. \quad (6)$$

The values $c_1\rho_1$ and $c_2\rho_2$ determine the weights of the two parts, and $c_1 + c_2$ is usually limited to 4 (Kennedy & Eberhard, 1995). To apply PSO, several parameters including the number of population (m), cognition learning factor (c_1), social learning factor (c_2), inertia weight (w), and the number of iterations or CPU time should be properly determined. We conducted the preliminary experiments, and the complete computational procedure of the PSO algorithm can be summarized as follows:

1. *Initialize:* Initialize parameters and population with random position and velocities.
2. *Evaluation:* Evaluate the fitness value (the desired objective function) for each particle.
3. *Find the pbest:* If the fitness value of particle i is better than its best fitness value (*pbest*) in history, then set the current fitness value as the new *pbest* to particle i.
4. *Find the gbest:* If any *pbest* is updated and it is better than the current *gbest*, then set *gbest* to the current value.
5. *Update velocity and position:* Update velocity and move to the next position according to Eqs. (5) and (6).
6. *Stopping criterion:* If the number of iterations or CPU time is met, then stop; otherwise go back to step 2.

Since PSO was first introduced to optimize various continuous nonlinear functions by Kennedy and Eberhard (1995), it has been successfully applied to a wide range of applications (Allahverdi & Al-Anzi, 2006; Eberhart and Shi, 2001; Kennedy and Eberhart, 1997; Kennedy et al., 2001; Moore & Chapman, 1999; Sousa et al., 2004; Trelea, 2003). However, the major obstacle of successfully applying a PSO is its continuous nature. To remedy this drawback, Kennedy and Eberhart (1997) and Kennedy et al. (2001) developed a discrete version of PSO (BPSO). In BPSO, the particle is characterized by a binary solution representation, and the velocity must be transformed into the change of probability for each binary dimension to take a value of one.

Basically, BPSO updates the velocity according to Eq. (5), but particles are represented by binary variables and without using w. Furthermore, the velocity is constrained to the interval [0,1] by using the following sigmoid transformation:

$$S(v_{ij}^t) = \frac{1}{1 - \exp(-v_{ij}^t)} \quad (7)$$

where $S(v_{ij}^t)$ denotes the probability of bit v_{ij}^t taking 1. To avoid $S(v_{ij}^t)$ approaching 0 or 1, a constant $V_{\max}$ is used to limit the range of velocity. Typically, $V_{\max}$ is often set at 4 such that $v_{ij}^t \in [-4, 4]$ and

$$v_{ij}^t = \begin{cases} V_{\max} & \text{if } v_{ij}^t > V_{\max} \\ -V_{\max} & \text{if } v_{ij}^t < -V_{\max} \\ v_{ij}^t & \text{otherwise} \end{cases}. \quad (8)$$

Each bit of particles, at each time step, changes its current position according to Eq. (9) instead of according to Eq. (6) based on Eq. (7) as follows (Kennedy and Eberhart, 1997; Kennedy et al., 2001):

$$x_{ij}^t = \begin{cases} 1 & \text{if } \rho < S(v_{ij}^t) \\ 0 & \text{otherwise} \end{cases}. \quad (9)$$

4. Proposed 2DPSO

This paper aims at creating a novel PSO in which each particle is coded in positive integer numbers and has a feasible system structure. The proposed 2DPSO can overcome the drawbacks of conventional PSO in order to solve MMRAP and the details are presented below.

4.1. Objective function

The RAP reliability can be determined by the following four common system configurations (see Figs. 3–6) (Yeh, 2003):

Case 1: The simple series system (see Fig. 3):

$$R(\mathbf{r}) = \prod_{i=1}^{n} \mathrm{r}_i, \quad (10)$$

Case 2: The simple parallel system (see Fig. 4):

$$R(\mathbf{r}) = 1 - \prod_{j=1}^{n} (1 - \mathrm{r}_j), \quad (11)$$

Case 3: The series-parallel system (see Fig. 5):

$$R(\mathbf{r}) = \prod_{i=1}^{n} \left[1 - \prod_{j=1}^{n_i} (1 - \mathrm{r}_{i:j}) \right], \quad (12)$$

Case 4: The parallel-series system (see Fig. 6):

$$R(\mathbf{r}) = 1 - \prod_{i=1}^{n} \left[1 - \prod_{j=1}^{n_i} \mathrm{r}_{i:j} \right], \quad (13)$$

Therefore, after selecting all the components of the solution, the reliability is calculated recursively from the lowest level to the highest level according to the above configurations.

4.2. Penalty function

One of the major drawbacks of the SGA proposed in Yun and Kim (2004) is that it causes many systems with unfeasible

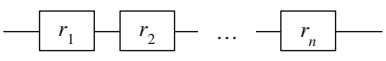

Fig. 3. The simple series system.

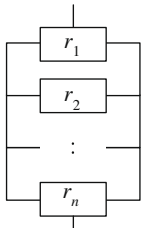

Fig. 4. The simple parallel system.

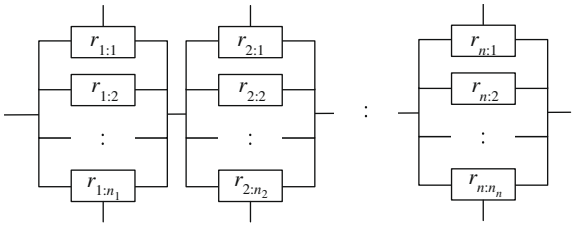

Fig. 5. The series-parallel system.

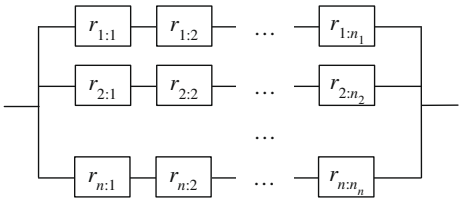

Fig. 6. The parallel-series systems.

structure and may converge only very slowly. To prevent this drawback and to guide the search towards unexplored regions in the solution space, the penalty function is implemented. For the particle i with the total system cost C_i that exceeds C_{UB}, a penalized reliability is calculated as

$$F(X_i^t) = \begin{cases} R(X_i^t) & \text{if } C(X_i^t) \leqslant C_{UB} \\ R(X_i^t) \cdot \left[\frac{C_{UB}}{C(X_i^t)}\right]^2 & \text{otherwise} \end{cases} \tag{14}$$

where the exponent 2 is a preset amplification parameter. This penalty function encourages the particles to explore the feasible region and infeasible region that is near the border of the feasible area to help the search not to go too far into the infeasible region. Thus, promising feasible and infeasible regions of the search space are explored efficiently and effectively to identify an optimal or near optimal solution.

4.3. Solution representation

One of the most important issues when designing the PSO algorithm is its solution representation (Liang & Chen, 2007). To construct a direct relationship between the problem domain and the PSO particles for the MMRAP, each dimension in the solution stands for a component, and its value is a positive integer denoted by its redundancy number. For example, $(S, A, B, C, A_1, A_2, A_3, B_1, B_2, C_1, C_2) = (0,2,2,3,0,0,0,1,1,0,0)$ represent the following structure of the MMRAP in Fig. 1.

$(S, A, B, C, A_1, A_2, A_3, B_1, B_2, C_1, C_2) = (0,2,2,3,0,0,0,1,1,0,0)$ in Fig. 1.

4.4. First stage in the proposed 2DPSO

The system performance depends on the combination of the items and the redundancy degree. The number of available redundancy structure increases exponentially as the number of basic items becomes large. To exclude unwanted redundancy structures and to reduce solution search space, two stages are used in the proposed 2DPOS. The first stage is employed to generate feasible structures of MMRAP. Then, the second stage is implemented in the proposed DPSO to decide the duplicated number of each available unit in the corresponding feasible structures of the MMRAP to maximize the system reliability under a cost constraint. The following simple recursive formula is the core of this study:

$$\text{If } T(s) = 1, \text{ then either } X(s) > 0, \text{ or } X(s) = 0 \text{ and } T(t) = 1 \text{ for all } t \in \text{Child}(s). \tag{15}$$

For example, if $X(A) > 0$, then $T(A_i) \in \{0,1\}$ and $X(A_i) \geqslant 0$ for $i = 1,2,3$ in Fig. 1. If $X(B) = 0$, then $T(B_1) = T(B_2) = 1$, $X(B_2) > 0$, and $X(B_2) > 0$. Eq. (15) is implemented to verify and limit each particle to guarantee that its corresponding structure is feasible in Section 4.5 and to generate a upper-bound for each dimension in Section 4.4.

4.5. Upper bound

In general, the number of solutions is quite large. To reduce the feasible search space, an upper-bound formulation of each dimension, say x_k, is generated based on Eq. (15) such that

$$C(U(x_k)) \leqslant C_U(x_k) < C(U(x_k) + 1) \tag{16}$$

where

$$C(x_k) = \begin{cases} c_k x_k + \lambda_k^{x_k} & \text{if } x_k > 0 \\ 0 & \text{if } x_i = 0 \end{cases} \tag{17}$$

$$C_L(x_k) = \begin{cases} c_k x_k + \lambda_k^{x_k} & \text{if } x_k \text{ is a leaf-node} \\ \text{Min}\{c_k x_k + \lambda_k^{x_k}, \sum_j C_L(x_j)\} & \text{otherwise} \end{cases} \tag{18}$$

$$C_U(x_k) = C_U(x_h) - \sum_i C_L(x_i) \tag{19}$$

x_h, x_i, and x_j are a parent-node, brother-node, and child-node of x_k, respectively.

For example, let Table 1 be the corresponding data for Fig. 1. Since A_1, A_2, A_3, B_1, B_2, C_1 and C_2 are the leaf-nodes in Fig. 1, we have for

$$C_L(A_1) = C(A_1) = 5 + 3^1 = 8, \tag{20}$$
$$C_L(A_2) = C(A_2) = 6 + 4^1 = 10, \tag{21}$$
$$C_L(A_3) = C(A_3) = 5 + 4^1 = 9, \tag{22}$$
$$C_L(B_1) = C(B_1) = 6 + 4^1 = 10, \tag{23}$$
$$C_L(B_2) = C(B_2) = 7 + 4^1 = 11, \tag{24}$$
$$C_L(C_1) = C(C_1) = 8 + 3^1 = 11, \tag{25}$$
$$C_L(C_2) = C(C_2) = 7 + 4^1 = 11, \tag{26}$$

and

Table 1
Data for Fig. 1.

i	Component	Parent component	r_i	c_i	λ_i
0	S	–	0.40029	72	2
1	A	S	0.72675	26	2
2	B	S	0.76500	19	3
3	C	S	0.72000	21	2
4	A_1	A	0.90000	5	3
5	A_2	A	0.95000	6	4
6	A_3	A	0.85000	5	4
7	B_1	B	0.90000	6	4
8	B_2	B	0.85000	7	4
9	C_1	C	0.90000	8	3
10	C_2	C	0.80000	7	4

$$C_L(\mathrm{A}) = \mathrm{Min}C(\mathrm{A}) = 26 + 2^1 = 28,$$
$$C_L(\mathrm{A}_1) + C_L(\mathrm{A}_2) + C_L(\mathrm{A}_3) = 27 = 27, \quad (27)$$
$$C_L(\mathrm{B}) = \mathrm{Min}C(\mathrm{B}) = 19 + 3^1 = 22,$$
$$C_L(\mathrm{B}_1) + C_L(\mathrm{B}_2) = 21 = 21, \quad (28)$$
$$(\mathrm{C}) = \mathrm{Min}C(\mathrm{C}) = 21 + 2^1 = 23, C_L(\mathrm{C}_1) + C_L(\mathrm{C}_2) = 22 = 22. \quad (29)$$

Also, we have

$$C_U(\mathrm{S}) = 150, \quad (30)$$
$$C_U(\mathrm{A}) = 150 - C_L(\mathrm{B}) - C_L(\mathrm{C}) = 107, \quad (31)$$
$$C_U(\mathrm{B}) = 150 - C_L(\mathrm{A}) - C_L(\mathrm{C}) = 101, \quad (32)$$
$$C_U(\mathrm{C}) = 150 - C_L(\mathrm{A}) - C_L(\mathrm{B}) = 102, \quad (33)$$
$$C_U(\mathrm{A}_1) = U(A) - C_L(\mathrm{A}_2) - C_L(\mathrm{A}_3) = 88, \quad (34)$$
$$C_U(\mathrm{A}_2) = U(\mathrm{A}) - C(T(\mathrm{A}_1)) - C_L(\mathrm{A}_3) = 90, \quad (35)$$
$$C_U(\mathrm{A}_3) = U(\mathrm{A}) - C(T(\mathrm{A}_1)) - C_L(\mathrm{A}_3) = 89, \quad (36)$$
$$C_U(\mathrm{B}_1) = U(\mathrm{B}) - C_L(\mathrm{B}_2) = 90, \quad (37)$$
$$C_U(\mathrm{B}_2) = U(\mathrm{B}) - C_L(\mathrm{B}_1) = 91, \quad (38)$$
$$C_U(\mathrm{C}_1) = U(\mathrm{C}) - C_L(\mathrm{C}_2) = 91, \quad (39)$$
$$C_U(\mathrm{C}_2) = U(\mathrm{C}) - C_L(\mathrm{C}_1) = 91. \quad (40)$$

Besides,

$$26 \cdot 3 + 2^3 = 86 < 107 < 26 \cdot 4 + 2^4 = 120, \quad (41)$$

we have

$$U(\mathrm{A}) = 3. \quad (42)$$

In the same way,

$$U(\mathrm{S}) = 2, U(\mathrm{C}) = 4, U(\mathrm{B}) = U(\mathrm{A}_1) = U(\mathrm{A}_2) = U(\mathrm{A}_3) = U(\mathrm{B}_1)$$
$$= U(\mathrm{B}_2) = U(\mathrm{C}_1) = U(\mathrm{C}_2) = 3. \quad (43)$$

The above result is summarized in Table 2.

4.6. Initial population

The initial population is generated randomly such that Eq. (15) is satisfied without transferring each particle into binary bits.

Table 2
The result for finding the upper-bound of each dimension in Fig. 1.

i	Component	c_i	λ_i	$C_L(x_i)$	$C_U(x_i)$	$U(x_i)$	$C(U(x_i))$
0	S	72	2	70	150	2	148
1	A	26	2	27	107	3	86
2	B	19	3	21	101	3	84
3	C	21	2	22	102	4	100
4	A_1	5	3	8	88	3	42
5	A_2	6	4	10	90	3	82
6	A_3	5	4	9	89	3	79
7	B_1	6	4	10	90	3	82
8	B_2	7	4	11	91	3	85
9	C_1	8	3	11	91	3	51
10	C_2	7	4	11	91	3	85

4.7. Second stage in the proposed 2DPSO: the proposed DPSO

The underlying principle of the traditional PSO is that the next position of each particle is a compromise of its current position, the best position in its history so far, and the best position among all the existing particles. From Eq. (5), it is very easy to efficiently decide the next positions for the problems with continuous variables, but it is not trivial and well-defined for the problems with discrete variables and sequencing problems. To overcome the drawback of PSO for discrete variables, a novel method to implement PSO procedure is proposed based on the following equation, after c_w, c_p, and c_g are given

$$x_{ij}^t = \begin{cases} x_{ij}^{t-1} & \text{if } \rho_{ij}^t \in [0, C_w) \\ p_{ij}^{t-1} & \text{if } \rho_{ij}^t \in [C_w, C_p) \\ g_i & \text{if } \rho_{ij}^t \in [C_p, C_g) \\ x & \text{if} \rho_{ij}^t \in [C_g + x \cdot \Delta_j, (x+1) \cdot \Delta_j) \end{cases}, \quad (44)$$

where

$$C_w = c_w, \quad (45)$$
$$C_p = C_w + c_p, \quad (46)$$
$$C_g = C_p + c_g, \quad (47)$$

$$\Delta_i = \frac{1 - C_g}{U(x_i) - L(x_i) + 1}. \quad (48)$$

For example, let $X_4^3 = (1,2,3,4,5)$, $P_4^3 = (6,7,3,2,4)$, $G_3 = (4,3,5,3,2)$, $c_w = .1$, $c_p = .25$, $c_g = .35$, $L(X) = (0,1,2,1,0)$, $U(X) = (9,10,6,6,5)$, and $\rho = (.91,.45,.28,.05,.87)$. Then,

$$C_w = c_w = .1, \quad C_p = C_w + c_p = .35, \quad C_g = C_p + c_g = .70, \quad (49)$$
$$C_1 = (1 - .70)/(U_i - L_i + 1) = (1 - .70)/(9 - 0 + 1)$$
$$= .03 = C_2, \quad C_3 = .06, \quad C_4 = C_5 = .05. \quad (50)$$

Since

$$C_g + 7 \cdot .03 = .91 = \rho_1 < C_g + 8 \cdot .03, \quad (51)$$
$$C_p < \rho_2 = .45 < C_g, \quad (52)$$
$$C_w < \rho_3 = .28 < C_p, \quad (53)$$
$$\rho_4 = .05 < C_w, \quad (54)$$
$$C_g + 5 \cdot .03 < \rho_5 = .87 < C_g + 6 \cdot .03, \quad (55)$$

we have $X_4^4 = (7,3,3,4,5)$.

Updating the velocity and positions is the most important part of PSO. It plays an important role in exchanging information among the particles. It leads to an effective combination of partial solutions in other particles and speeds up the search procedure early in the generation. In the traditional PSO, each particle needs to use two more equations (i.e. Eqs. (5), (7)–(9)), generate three random numbers (e.g. w, r_1, and r_2), five multiplications, and three summations to move to its next position. Thus, the time complexity is $O(8nm)$ for the traditional PSO. However, there is no need to use the velocity (i.e. Eq. (5)), and one random number, two multiplications, and one comparison are needed in the proposed 2DPSO after c_w, c_p, and c_g are given. Therefore, the proposed 2DPSO is more efficient than other PSOs.

5. A numerical example

A numerical experiment is provided to the small-scale MMRAP system (Fig. 1) and the corresponding numerical parameters and data are presented in Table 1 for evaluating the performance and confirming the validity of 2DPSO. This benchmark MMRAP example is taken from Yun & Kim, 2004 with 11 basic items and the system cost constraint $g_r(x_i) = c_i x_i + \lambda_i^{x_i}$ ranges from 150 to 340 in steps of 10.

This example was implemented using the proposed 2DPSO, BPSO, and the best-known method (GA) and the results of the experiments were compared. The proposed 2DPSO and BPSO for MMRAP were implemented in *C* programming language on an Intel Pentium 2.6 GHz PC with 256 MB memory. For fair comparison, the number of iterations (120), the number of runs (10), and the populations (120) for the proposed 2DPSO and BPSO are taken directly from Yun & Kim, 2004. In addition, $c_g = 0.5$, $c_p = 0.3$, and $c_w = 0.1$ are adopted in the proposed 2DPSO, and $c_1 = c_2 = 2$ and $V_{max} = 4$ are used in the BPSO.

The results are summarized in Tables 3 and 4 and Figs. 7–10 contain the summary of the results. The shaded box shows the best result. The complete GA structure proposed in the best-known method for the MMRAP has limitations in penalty function and in handling the solutions with unfeasible structures. Yun and Kim (2004) present only the summary of results, while CPU time, the number of best solutions reached in 10 runs and the average numbers of the earliest finding of the best solutions are all unavailable. The solutions obtained using 2DPSO are in agreement with those obtained using GA. However, for Budget (190) in the benchmark MMRAP example, the proposed 2DPSO has achieved better results (184) with less than 3% of GA cost (188). Furthermore, 2DPSO is superior to BPSO in all categories and at all problem instances.

Table 3 shows that the proposed 2DPSO uses only one-fourth of CPU time in average compared to BPSO and always keeps it around 0.033 irrespective of the budget limitation. Also, 2DPSO has lower standard deviations (STDEV) in obtained reliabilities which can be verified from Figs. 8 and 10. Therefore, the proposed 2DPSO is more robust than BPSO.

In Table 4, the comparisons between 2DPSO and BPSO over 10 runs are divided into three categories: maximum, minimum, and average system reliabilities (denoted by R_{max}, R_{min}, and R_{avg}, respectively). These comparisons are also measured using the percentage maximum possible improvement (%MPI) and plotted in Fig. 9. %MPI is the percentage that the best feasible solution improves upon another, i.e. %MPI = 1 − (the result obtained from 2DPSO)/(the corresponding result obtained from BPSO).

Table 4
The comparison of computational results.

Budget	2DPSO			BPSO		
	R_{max}	R_{min}	R_{avg}	R_{max}	R_{min}	R_{avg}
150	0.8057	0.8057	0.8057	0.8057	0.7193	0.7836
160	0.8316	0.8309	0.8313	0.8309	0.8031	0.8201
170	0.8576	0.8511	0.8540	0.8550	0.8210	0.8430
180	0.8773	0.8680	0.8733	0.8680	0.8316	0.8524
190	0.8920	0.8799	0.8891	0.8796	0.8621	0.8753
200	0.9136	0.8974	0.9060	0.9040	0.8736	0.8859
210	0.9319	0.9157	0.9228	0.9319	0.8826	0.9044
220	0.9457	0.9301	0.9346	0.9333	0.8876	0.9076
230	0.9535	0.9400	0.9516	0.9364	0.9077	0.9217
240	0.9587	0.9492	0.9565	0.9511	0.9219	0.9336
250	0.9641	0.9596	0.9619	0.9566	0.9412	0.9485
260	0.9694	0.9631	0.9671	0.9600	0.9198	0.9482
270	0.9739	0.9711	0.9727	0.9643	0.9482	0.9586
280	0.9773	0.9741	0.9764	0.9711	0.9560	0.9620
290	0.9808	0.9774	0.9797	0.9777	0.9564	0.9692
300	0.9835	0.9814	0.9824	0.9808	0.9652	0.9754
310	0.9861	0.9840	0.9852	0.9830	0.9620	0.9759
320	0.9888	0.9866	0.9882	0.9853	0.9764	0.9814
330	0.9903	0.9883	0.9898	0.9875	0.9798	0.9848
340	0.9918	0.9897	0.9904	0.9866	0.9798	0.9841
Avg.	0.9387	0.9322	0.9360	0.9324	0.9048	0.9208

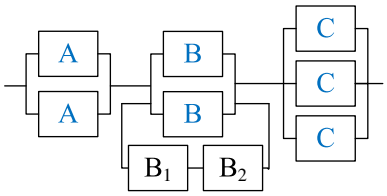

Fig. 7. The structure w.r.t. $(S, A, B, C, A_1, A_2, A_3, B_1, B_2, C_1, C_2) = (0, 2, 2, 3, 0, 0, 0, 1, 1, 0, 0)$ in Fig. 1.

An interesting observation is that %MPI is lower upon increasing the budget constraint in Fig. 9. This may result from the slowly increasing objective function as the budget is close to 340, as shown in Fig. 10.

Table 3
Comparison of the proposed 2DPSO result with the best available one.

Budget	GA[a]			2DPSO						BPSO					
	Solution	Reliability	Cost	Reliability	Cost	Solution	CPU secs.	$N^{\#}$	N^{*}	Reliability	Cost	Solution	CPU secs.	$N^{\#}$	N^{*}
150	01111111111	0.805693	143	0.805693	143	01111111111	0.035	31.1	8	0.805693	144	02110001111	0.171	0.0	1
160	01111121111	0.831629	160	0.831629	160	01111121111	0.032	39.8	4	0.830871	158	02200000022	0.156	37.0	3
170	01101121122	0.857618	170	0.857618	170	01101121122	0.037	44.1	4	0.855042	167	02120001111	0.156	78.0	1
180	00112221112	0.877267	180	0.877267	180	00112221112	0.035	51.0	3	0.868025	179	02200001122	0.164	69.0	2
190	00222220011	0.891977	188	0.891977	184	00122221111	0.033	63.0	2	0.879595	184	03100001122	0.157	59.5	2
200	00022222211	0.913644	199	0.913644	199	00022222211	0.033	32.3	3	0.904006	198	01021122211	0.157	49.5	2
210	00222221111	0.931862	209	0.931862	209	00222221111	0.036	51.7	2	0.931863	209	00222221111	0.172	33.0	1
220	02221111111	0.945659	219	0.945659	219	02221111111	0.036	64.5	4	0.933252	219	01012122222	0.172	17.0	1
230	02211111122	0.953456	229	0.953456	229	02211111122	0.038	70.2	4	0.936392	230	03120001221	0.157	44.0	1
240	02212111122	0.958738	240	0.958738	240	02212111122	0.033	41.8	2	0.951117	236	03220001121	0.164	39.5	2
250	01212221122	0.964087	247	0.964087	247	01212221122	0.037	55.0	2	0.956575	241	03220001112	0.157	52.0	1
260	01112222222	0.969355	259	0.969355	259	01112222222	0.033	49.3	2	0.959973	259	04230001100	0.156	83.0	1
270	01222221122	0.973863	270	0.973863	270	01222221122	0.033	46.2	2	0.964271	269	03210002222	0.156	118.0	1
280	02222121122	0.977262	280	0.977262	278	02222121122	0.034	53.6	3	0.971054	275	04220001112	0.172	38.0	1
290	03131112211	0.980817	286	0.980817	286	03131112211	0.037	51.3	2	0.956415	278	04040002200	0.156	103.0	1
300	02222121222	0.983537	299	0.983537	299	02222121222	0.037	57.0	3	0.980817	293	04130002211	0.156	60.0	1
310	02232122211	0.986107	309	0.986107	309	02232122211	0.037	50.0	4	0.983045	301	04120002222	0.171	86.0	1
320	03221112222	0.988792	319	0.988792	319	03221112222	0.036	42.6	5	0.985299	318	04330001111	0.172	79.0	1
330	03222112222	0.990266	330	0.990266	330	03222112222	0.036	51.1	5	0.987537	326	04320001122	0.161	71.7	3
340	02222222222	0.991760	335	0.991760	335	02222222222	0.035	42.0	2	0.986551	325	04240002200	0.156	61.0	1
Avg.			335	0.991760	335	02222222222	0.035	42.0	2	0.986551	325	04240002200	0.156	61.0	1

N^{*}, the number of best solutions reached in 10 runs.
$N^{\#}$, the average generation numbers of the best solution reached for the first time.
[a] The CPU secs., N^{*}, and $N^{\#}$ are all unknown.

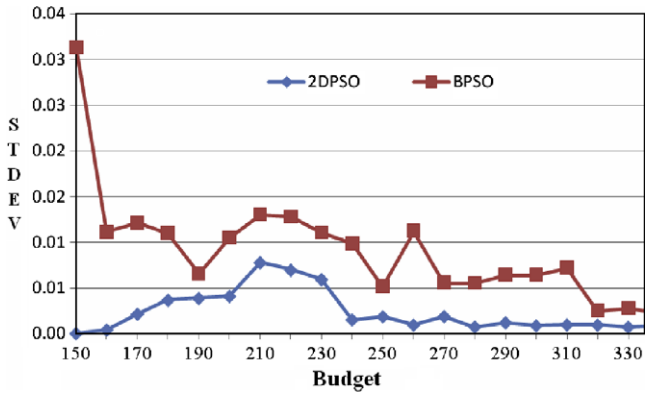

Fig. 8. The STDEV trend for 2DPSO and BPSO, respectively.

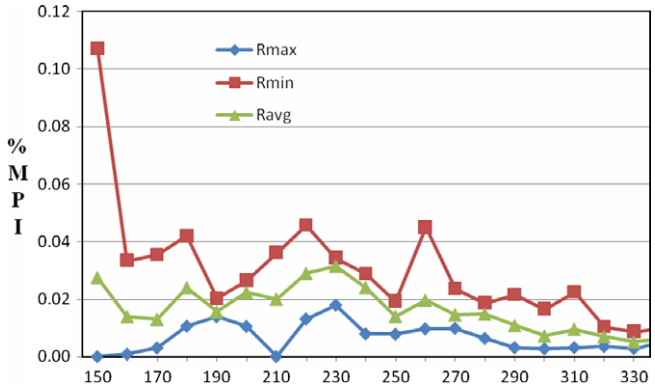

Fig. 9. The %MPI for R_{max}, R_{min}, and R_{avg}, respectively.

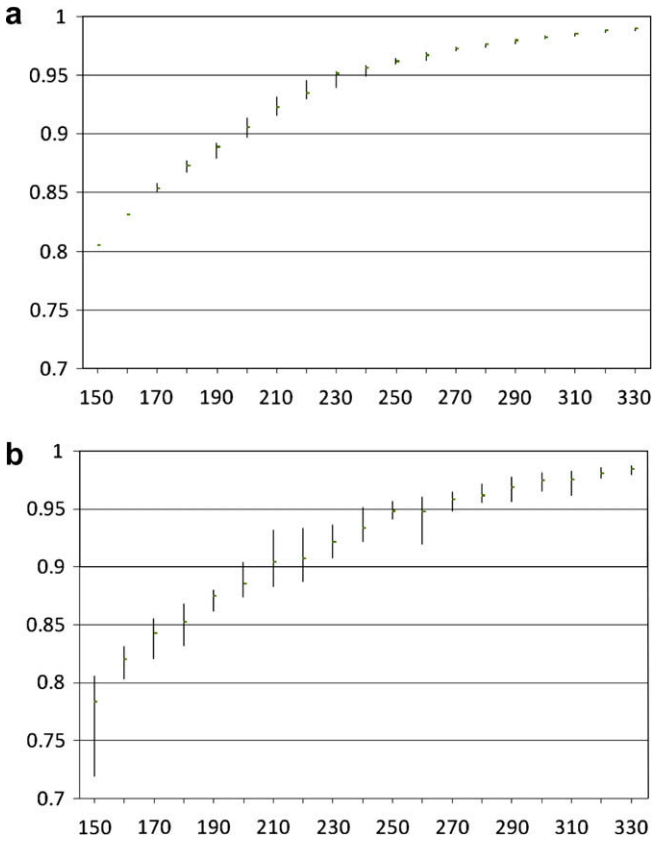

Fig. 10. Range of performance for 10 runs with mean represented by horizontal dash (a) 2DPSO and (b) BPSO.

In summary, the above proposed 2DPSO outperforms BPSO in all cases including the running time, solution quality, robustness, number of times it finds the best solutions, and the earliest number of finding the best solution. Further, it also improves one of the solutions obtained from the best-known method (implemented using GA) in MMRAP by almost 3%.

6. Conclusions

Network reliability theory has been applied extensively in many real-world systems. In practical applications, reliability is most often assigned as informal compromise between functional quality and manufacturing cost. With the increase in the competition in today's market place, small savings in cost or small increase in reliability may determine the success of a product. RAP is an efficient method to improve reliability.

In this study, a new discrete version of PSO called 2DPSO is proposed to solve MMRAP in which all available items for redundancy can be simultaneously chosen. A novel, simpler, and more efficient mechanism is employed to move each particle to its next position in the proposed 2DPSO. From numerical experiments, the results obtained clearly show the competitiveness of the proposed 2DPSO, which is able to outperform the best-known results and the traditional discrete PSO (BPSO) for the benchmark in both the running time and solution quality (Yun & Kim, 2004; Yun et al., 2007).

With the success of developing a good approximation by the proposed 2DPSO, MMRAP can be analyzed, sensitivity analysis can be performed, and the solution can be obtained in an effective way with better quality. It can also help us gain better understanding of the structure of the problem through analyzing MMRAP.

In summary, the results presented in this work are encouraging and promising for the application of the proposed 2DPSO to the MMRAP, and hence to the other discrete problems. For future studying, this work would be extended to apply the proposed 2DPSO to solve other combinatorial optimization discrete problems.

Acknowledgement

This research was supported in part by the National Science Council of Taiwan, ROC, under Grant NSC 90-2218-E-035-006.

References

Agarwal, M., & Gupta, R. (2005). Penalty function approach in heuristic algorithms for constrained. *IEEE Transaction Reliability, 54*(3), 549–558.

Allahverdi, A., & Al-Anzi, F. S. (2006). A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. *Computers and Operations Research, 33*, 1056–1080.

Bland, J. A. (1998). Structural design optimization with reliability constraints using tabu search. *Engineering Optimization, 30*, 55–74.

Chen, T. C. (2006). IAs based approach for reliability redundancy allocation problems. *Applied Mathematics and Computation, 182*(2), 1556–1567.

Chern, M. S. (1992). On the computational complexity of reliability redundancy allocation in a series system. *Operations Research Letters, 11*, 309–315.

Eberhart, R.C., & Shi, Y., (2001). Particle swarm optimization: Developments, application and resources. In *Proceedings of the 2001 congress on evolutionary computation, Seoul, South Korea* (Vol. 1, pp. 81–86).

Ha, C., & Kuo, W. (2006). Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems. *European Journal of Operational Research, 171*, 24–38.

Hsieh, Y. C. (2002). A linear approximation for redundant reliability problems with multiple component choices. *Computers and Industrial Engineering, 44*, 91–103.

Hsieh, Y. C., Chen, T. C., & Bricker, D. L. (1998). Genetic algorithms for reliability design problems. *Microelectronics Reliability, 38*(10), 1599–1605.

Kennedy, J., & Eberhard, R.C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks, Piscataway, NJ, USA* (pp. 1942–1948).

Kennedy, J., Eberhard, R. C., & Shi, Y. (2001). *Swarm intelligence*. San Francisco, CA: Morgan Kaufmann.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm, systems, man, cybernetics, computational cybernetics and simulation. *IEEE International Conference, 5*(12–15), 4104–4108.

Kim, H. G., Bae, C. O., & Park, D. J. (2006). Reliability-redundancy optimization using simulated annealing algorithms. *Journal of Quality in Maintenance Engineering, 12*(4), 354–363.

Kuo, W., & Wan, R. (2007). Recent advances in optimal reliability allocation. *IEEE Transactions on Systems, Man and Cybernetics, Part A, 37*(2), 143–156.

Kuo, W., & Prasad, V. R. (2000). An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability, 49*(2), 176–187.

Kuo, W., Prasad, V. R., Tilman, F. A., & Hwang, C. L. (2001). *Optimal reliability design: Fundamentals and applications*. UK: Cambridge University Press.

Liang, Y. C., & Chen, Y. C. (2007). Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm. *Reliability Engineering and System Safety, 92*, 323–331.

Liang, Y. H., & Smith, A. E. (2004). An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Transactions on Reliability, 53*(3), 417–423.

Moore, J., & Chapman, R. (1999). *Application of particle swarm to multiobjective optimization*. Department of Computer Science and Software Engineering, Auburn University.

Onishi, J., Kimura, S., James, Ross J. W., & Nakagawa, Y. (2007). Solving the redundancy allocation problem with a mix of components using the improved surrogate constraint method. *IEEE Transactions on Reliability, 56*(1), 94–101.

Ravi, V., Reddy, P. J., & Zimmermann, Hans-Jürgen (2000). Fuzzy global optimization of complex system reliability. *IEEE Transactions on Fuzzy Systems, 8*(3), 241–248.

Sousa, T. F., Silva, A. P., & Silva, A. F. (2004). Particle swarm based data mining algorithms for classification tasks. *Parallel Computing, 30*(5–6), 767–783.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters, 85*, 317–325.

Yeh, W. C. (2003). A MCS-RSM approach for the network reliability to minimize the total cost. *International Journal of Advanced Manufacturing Technology, 22*(9–10), 681–688.

Yun, W. Y., & Kim, J. W. (2004). Multi-level redundancy optimization in series systems. *Computers and Industrial Engineering, 46*, 337–346.

Yun, W. Y., Song, Y. M., & Kim, H. G. (2007). Multiple multi-level redundancy allocation in series systems. *Reliability Engineering and System Safety, 92*, 308–313.