Iranian Journal of Operations Research
Vol. 5, No. 2, 2014, pp. 26-51

Availability Optimization of A Series Repairable System with Multiple k-out-of-n Subsystems

A. Eshraghniaye Jahromi [1], Ali A. Yahyatabar Arabi [2,*]

We address the redundancy allocation of a series repairable system with multiple k-out-of-n subsystems with each subsystem following shut-off rule. The objective of the problem is to find the optimal number of repairmen and redundant components in each subsystem for optimization of steady-state availability subject to weight, cost and volume constraints. We propose a Particle Swarm Optimization (PSO) algorithm to solve the problem. The PSO algorithm is demonstrated to be more efficient as compared to a Simulated Annealing (SA) algorithm.

***Keywords**: Repairable component, Redundancy allocation, Repairman, Availability optimization.*

Manuscript was received on 25/07/2013, revised on 03/04/2015 and accepted for publication on 02/07/2015.

1. Introduction

With advancing technology, most industrial processes are involved in engineering systems. System designers are often concerned the performance of their systems, and most essential elements impacting the performance of a system are availability and reliability.

Considering the nature and application of a system, it is categorized into repairable or non-repairable. Repairability of a system being a prominent feature, one of the most important performance criteria in repairable systems is the availability of the system. To enhance the availability of a system, a usual approach is to improve the availability of subsystems and one way to achieve this is redundancy allocation. For redundancy allocation, optimal configuration plays a major role especially for repairable systems. In fact, maintenance of a repairable system is harder, i.e., more cost consuming, than that of a non-repairable system.

Redundancy allocation is a renowned approach and has been studied extensively in the areas of reliability and availability. Redundancy Allocation Problem (RAP) is to maximize reliability and availability of a system consisting of repairable or non-repairable component selection in the presence of some constraints such as weight, volume and cost.

Various models of RAP and solution methods have been proposed in the literature (See [2], [4] and [5]). RAP is applied to various structures such as series, parallel, parallel-series, series-parallel and k-out-of-n. A straightforward definition of k-out-of-n structure is that at least k components out of n parallel components need to work. Comprehensive overviews of k-out-of-n systems may be found in

[*] Iran, Tehran, Department of Industrial Engineering, Khaje Nasir Toosi University of Technology, Email: ayahyatabar@kntu.ac.ir.
[1] Iran, Tehran, Department of Industrial Engineering, Sharif University of Technology, Email: Eshragh@sharif.edu.
[2] Iran, Tehran, Department of Industrial Engineering, Khaje Nasir Toosi University of Technology, Email: ayahyatabar@kntu.ac.ir.

[9], [10]. Barron et al. [1] presented an analysis of k-out-of-n systems with some repairmen (the number of repairmen being fixed) and exponential lifetimes. Frostige et al. [3] studied on the availability of a k-out-of-n repairable system. They used Markov renewal processes for a k-out-of-n system in which components were repairable. A knowledge-based interactive decision support system was developed to set up and store component parameters during the design process of repairable series-parallel system in [13].

Khatab et al. [14] studied a k-out-of-n system with N categories. In their study, there are R specified repair facilities and repair priorities are specified between the N non-identical components. A multi-dimensional Markov model was used to evaluate the availability of the system. Krishnan et al. [15] used Laplace transform techniques to analyze the reliability of repairable consecutive-k-out-of-n systems with a sensing device and repairmen. A sensing device was applied to detect the failure of each component in the system in advance and completion of repair of the components.

A shut-off rule expresses a condition in which there are some non-failed components while the system is down. Two kinds of shut-off rule were investigated in [8]. Continuous operation (CO) describes a kind of shut-off rule that the system is down, but all non-failed components can still fail. Another kind of shut-off is called suspended animation (SA). In SA, there is no additional failure of non-failed components when the system is down. Availability of k-out-of-n system with respect to suspended animation was analyzed by D. Huffman et al. [6]. Moghaddass et al. [16] analyzed the reliability and availability of a repairable k-out-of-n system considering R specified repairmen subject to shut-off rules using a Markovian process. They presented new closed form solutions for important performance measures including steady-state availability, mean time to failure, and mean time to first system failure with respect to all shut-off rules.

Although the availability of k-out-of-n systems has been extensively investigated, we present a new model here. What is novel in our work is consideration of a new variable to enhance the availability of a system so-called optimal number of repairmen or repair facilities in each subsystem. In general, the classical model, RAP is used along with the number of repairmen which were neglected in the literature for a k-out-of-n system. The proposed model is applied frequently in practice. Important application can be found in Power industry, oil and gas industry, aviation industry, etc.

The system structure is depicted in Fig.1. The components are identical in each subsystem and all components are active and repairable. All subsystems follow SA as the shut-off rule.

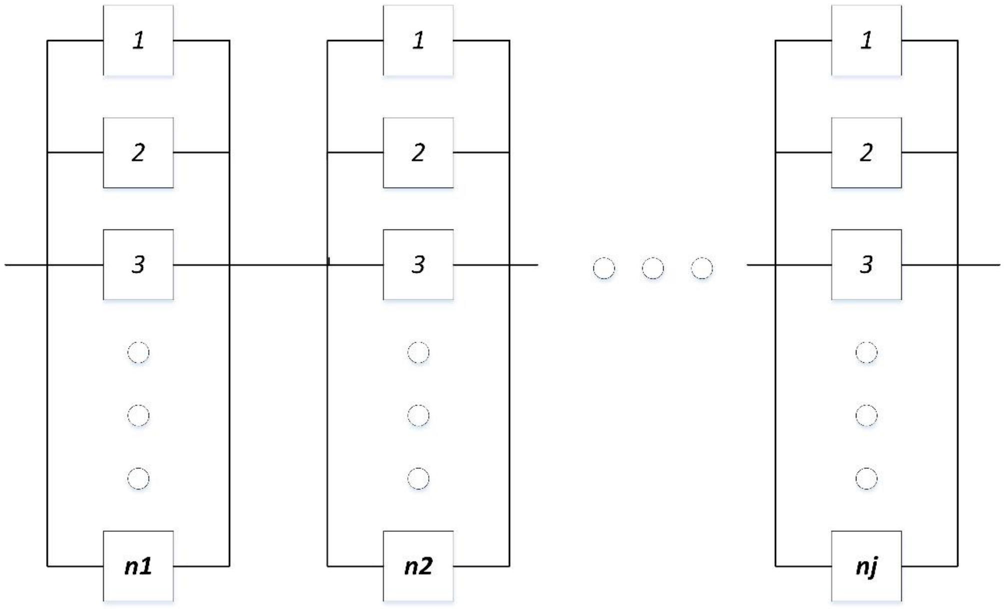

Figure 1. The system structure

Chern [17] illustrated that a simple redundancy allocation problem in series with linear constraints is NP-hard. As stated earlier, the main contribution of our study is to obtain the highest availability with an optimal number of repairmen and redundancy level in each subsystem as decision variables subject to cost, weight, and volume constraints. Due to the nonlinear structure of the objective function, there is not possibility to solve the problem by solver applications like GAMS or LINGO. Thus, a heuristic algorithm,exactly named Particle Swarm Optimization (PSO) algorithm, is proposed to solve the model.

Numerical experiments are made to show the efficiency of the proposed PSO algorithm in comparison with another heuristic algorithm called Simulated Annealing (SA). However, exact optimal solutions are required to show the closeness of the results obtained by heuristic algorithms. In this regard, we tried to get the exact optimal solution using a search method.

The reminder of our work is organized as follows. Section 2 considers problem formulation. In Section 3, the particle swarm optimization (PSO) algorithm is presented. The simulated annealing algorithm (SA) is described in Section 4. Section 5 presents adapted PSO and SA algorithm for the availability optimization. Parameters of the PSO and SA algorithms are set in Section 6. Numerical examples are presented and analyzed to prove the efficiency of the presented algorithm in Section 7. Finally, Section 8 gives the concluding remarks and provides directions for future research.

2. A Model for the system

2.1 System description

A series system with multiple k-out-of-n subsystems is discussed in here wherein each subsystem has repairable components. One type of variable is the number of repairmen; optimal number of repairmen would be allocated to each subsystem. However, the number of repairmen is costly and it should be considered as a variable subject to cost constraint. Another type of variable is redundancy level which directly affects all constraints, i.e., cost, volume and weight.

2.2 Notations

y: Number of subsystems in the system.
n_j : Number of allocated components in subsystem j.
λ_i: Failure rate of each subsystem when there are i failed components in each subsystem, $(1 \leq i \leq n_j - k_j)$.
μ_i : Repair rate of each subsystem when there are i ailed components in each subsystem, $(1 \leq i \leq n_j - k_j + 1)$.

r_j: Number of allocated repairmen in subsystem j.

$P_{i,j}(t)$: Probability that there are i failed components in subsystem j at time t, $(0 \leq i \leq n_j - k_j + 1)$.

$P_{i,j}$: Steady-state probability when there are i failed components in subsystem j, $(0 \leq i \leq n_j - k_j + 1)$.

$P'_{i,j}(t)$: First derivative of $P_{i,j}(t)$, $(0 \leq i \leq n_j - k_j + 1)$.

$A_{sub\ j}(t)$: Point availability of subsystem j at time t.

$A_{sub\ j}$: Steady-state availability of subsystem j.

A_s : Steady-state availability of system.

C : Total allowable cost of the system.

W : Weight upper limit of the system.

V : Volume upper limit of the system.

c_j: Cost of a component in subsystem j.

h_j : Cost of employing a repairman assigned to subsystem j.

w_j : Weight of a component in subsystem j.

v_j: Volume of a component in subsystem j.

2.3 Assumptions

- All subsystems have a k-out-of-n structure with active components.
- All components have identical independent distribution following exponential lifetime distribution.

- Only one repairman is allowed to be allocated to the repair of a failed component. The time to repair a failed component follows an identical independent exponential distribution, $1 \leq i \leq n_j - k_j + 1$.
- Subsystem j is called failed as soon as the number of failed components just reaches $n_j - k_j + 1$, and the system is considered failed as soon as one of the subsystems fails.
- All subsystems follow the suspended animation rule (a kind of shut-off rule).
- When a component of a subsystem fails, allocated repairman of the subsystem, if available, immediately begins; if not, the failed component must wait for the repairman. The repair is based on first-come, first-served.
- The probability that two or more components are restored or fail simultaneously in a small interval is not considered.

3. A Mathematical model

We consider an objective function associated with the maximization of the steady-state availability. The steady-state availability of the system (regarding Fig. 1) can be written as follows:

$$A_s = \prod_{j=1}^{y} A_{sub\ j} \quad , \tag{1}$$

With the availability of subsystem j being:

$$A_{sub\ j}(t) = 1 - P_{n_j-k_j+1,j}(t), \tag{2}$$

$$A_{sub\ j} = \lim_{t\to\infty} A_{sub\ j}(t) = 1 - 1 - \lim_{t\to\infty} P_{n_j-k_j+1,j}(t). \tag{3}$$

In accordance with the model description, steady-state availability of a k-out-of-n system is described using the Markov process to reach the availability of a subsystem [16]. The state transition diagram of a repairable k-out-of-n system is shown in Fig. 2. The numbers in the circle state the number of failed components; in fact, the status of the subsystem. The number $n - k + 1$ indicates that the system failed.

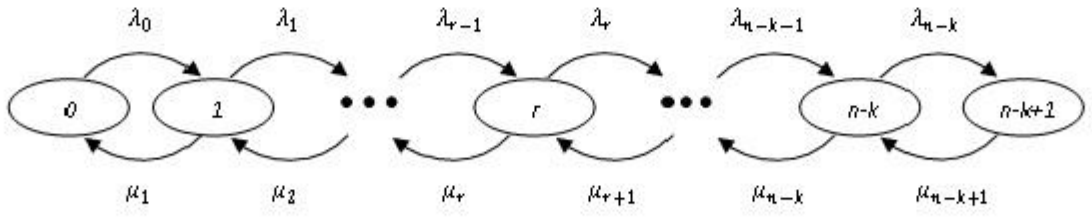

Figure 2. The state transition diagram of a repairable k-out-of-n system

The probability that there are i failed components in the subsystem j at time $t+\Delta t$ is shown by $P_{i,j}(t+\Delta t)$. To obtain this probability, the following four events are evaluated [18]:

1. During $(t, t+\Delta t)$, a transition happens; subsystem j is in status $i+1$ at time t and is in status *i* at time $t+\Delta t$.
2. During$(t, t+\Delta t)$, a transition happens; subsystem j is in status $i-1$ at time t and is in status *i* at time $t+\Delta t$.
3. During$(t, t+\Delta t)$, no change happens in the status of subsystem j.
4. During$(t, t+\Delta t)$, the status of subsystem j is transmitted by two or more.

For a Poisson process, the last event is negligible and close to zero. Based on Fig.2, the probability of the subsystem can be written as follows:

$$P_{i,j}(t+\Delta t) = P_{i+1,j}(t)(1-\lambda_{i+1}\Delta t)(\mu_{i+1}\Delta t) + P_{i-1,j}(t)(\lambda_{i-1}\Delta t)(1-\mu_{i-1}\Delta t) + P_{i,j}(t)(1-\lambda_i\Delta t)(1-\mu_i\Delta t) = P_{i,j}(t) - P_{i,j}(t)(\lambda_i+\mu_i)\Delta t + P_{i-1,j}(t)(\lambda_{i-1}\Delta t) + P_{i+1,j}(t)(\mu_{i+1}\Delta t). \quad (4)$$

Letting $\Delta t \to 0$,

$$P'_{i,j}(t) = -(\lambda_i+\mu_i)P_{i,j}(t) + \lambda_{i-1}P_{i-1,j}(t) + \mu_{i+1}P_{i+1,j}(t). \quad (5)$$

The steady-state distribution of subsystem j in discrepant status is expressed as follows:

$$P_{i,j} = \lim_{t\to\infty} P_{i,j}(t), \qquad \text{for } i = 0,1,\dots,n_j-k_j+1. \quad (6)$$

According to equations (5) and (6) we have

$$\lim_{t\to\infty} P'_{i,j}(t) = \lim_{t\to\infty}\left(-(\lambda_i+\mu_i)P_{i,j}(t) + \lambda_{i-1}P_{i-1,j}(t) + \mu_{i+1}P_{i+1,j}(t)\right),$$

$$-(\lambda_i+\mu_i)P_{i,j} + \lambda_{i-1}P_{i-1,j} + \mu_{i+1}P_{i+1,j} = 0, \quad (7)$$

$$\lambda_0 P_{0,j} + \mu_1 P_{1,j} = 0,$$

$$P_{1,j} = \frac{\lambda_0}{\mu_1}P_{0,j},$$

$$P_{2,j} = \frac{\lambda_0\lambda_1}{\mu_1\mu_2}P_{0,j},$$

.

.

.

$$P_{i,j} = \frac{\lambda_0\dots\lambda_{i-1}}{\mu_1\dots\mu_i}P_{0,j} \quad , \qquad \text{for } i = 1,\dots,n_j-k_j+1. \quad (8)$$

The total probability is given by

$$P_{0,j} + P_{1,j} + \cdots + P_{n_j-k_j+1,j} = 1. \tag{9}$$

Based on equations (8) and (9), we get

$$P_{0,j} = (1 + \sum_{i=1}^{n_j-k_j+1} \frac{\lambda_0 \dots \lambda_{i-1}}{\mu_1 \dots \mu_i})^{-1}. \tag{10}$$

According to equations (8) and (10), we have

$$P_{i,j} = \frac{\lambda_0 \dots \lambda_{i-1}}{\mu_1 \dots \mu_i}\left(1 + \sum_{i=1}^{n_j-k_j+1} \frac{\lambda_0 \dots \lambda_{i-1}}{\mu_1 \dots \mu_i}\right)^{-1}, \text{for } i = 1, \dots, n_j - k_j + 1 \tag{11}$$

The steady-state availability of subsystem j based on equations (3) and (11) is evaluated to be

$$A_{sub\ j} = 1 - P_{n_j-k_j+1,j} = 1 - \frac{\lambda_0 \dots \lambda_{n_j-k_j}}{\mu_1 \dots \mu_{n_j-k_j+1}}\left(1 + \sum_{i=1}^{n_j-k_j+1} \frac{\lambda_0 \dots \lambda_{i-1}}{\mu_1 \dots \mu_i}\right)^{-1}. \tag{12}$$

Therefore, the steady-state availability of the system based on equation (1) can be written as follows:

$$A_s = \prod_{j=1}^{y}\left(1 - \frac{\lambda_0 \dots \lambda_{n_j-k_j}}{\mu_1 \dots \mu_{n_j-k_j+1}}\left(1 + \sum_{i=1}^{n_j-k_j+1} \frac{\lambda_0 \dots \lambda_{i-1}}{\mu_1 \dots \mu_i}\right)^{-1}\right). \tag{13}$$

A subsystem j is considered as failed when the failed components reach $n_j - k_j + 1$, based on the shut-off rule; there are i failed components and $(n_j - i)$ working components in each subsystem j. We are going to use results from queuing theory for the failure rate of the subsystem j and repair rate of the subsystem j as follows:

$$\lambda_i = (n_j - i), 0 \le i \le n_j - k_j \tag{14}$$

$$\mu_i = \begin{cases} i\mu, & 0 \le i \le r_j \\ r_j\mu, & r_j \le i \le n_j - k_j + 1 \end{cases}. \tag{15}$$

Using the definitions above, the mathematical model of the system is formulated to be:

Maximize

$$A_s = \prod_{j=1}^{y}\left(1 - \left(\frac{n_j!}{(k_j-1)!r_j r_j^{n_j-k_j+1-r_j}}\left(\frac{\lambda}{\mu}\right)^{n_j-k_j+1}\right)\left(1 + \left(\sum_{i=1}^{r_j}\left(\binom{n_j}{i}\left(\frac{\lambda}{\mu}\right)^{i}\right) + \left\lfloor\frac{n_j+M}{r_j+k_j+M}\right\rfloor\left(\sum_{i=r_j+1}^{n_j-k_j+1}\left(\frac{n_j!\left(\frac{\lambda}{\mu}\right)^{i}}{(n-i)!r_j!r_j^{i-r_j}}\right)\right)\right)\right)^{-1}\right) \tag{16}$$

Subject to

$$\sum_{j=1}^{y} c_j n_j + \sum_{j=1}^{y} h_j r_j \le C \tag{17}$$

$$\sum_{j=1}^{y} w_j n_j \leq W \tag{18}$$

$$\sum_{j=1}^{y} v_j n_j \leq V \tag{19}$$

$$1 \leq r_j \leq (n_j - k_j + 1), \qquad \forall\, j = 1,2,\ldots,y \tag{20}$$

$$k_j \leq n_j, \qquad \forall\, j = 1,2,\ldots,y \tag{21}$$

With respect to equation (16), the objective is to determine the number of repairmen and the number of components in each subsystem to optimize the availability of the system according to Fig. 1. In equation (16), M is a large number. When $r_j = n_j - k_j + 1$, the floor value in the bracket is zero and when $r_j = n_j - k_j$, the floor value in the bracket is one. Constraints given by equations (17), (18) and (19) represent the available cost, weight and volume respectively. Constraint (20) indicate the upper and lower bounds of the repairmen in the subsystem j. Constraints (21) represent the k-out-of-n structure of subsystem j.

Considering equation (16), the model is a non-linear programming problem. Due to the complexity of the objective function, classical methods are not easily applicable. A main contribution of our work here is to apply heuristic methods such as particle swarm optimization (PSO) and simulated annealing (SA) algorithms.

4. Particle swarm optimization

Particle swarm optimization (PSO) is an optimization technique based on motion and intelligence of swarms. Kennedy and Eberhart [7] have developed this method using several particles as a group to find the best solution.

Each particle is a point in an *N*-dimensional space that adjusts its flight based on its flight experience and other particles 'experience. The best solution found by a particle is called personal best (p_{best}) and another important value in this algorithm is the best solution by all particles called global best (g_{best}). The main concept of PSO is the notion of velocity of each particle according to $g_{best}$and p_{best} used as a weighted velocity in each step as shown in Fig. 3.

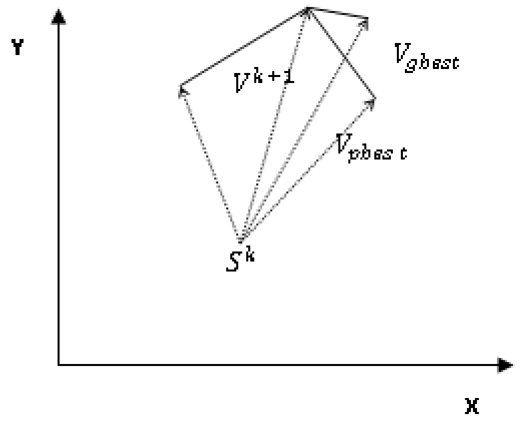

Figure 3. Particle swarm optimization algorithm applied to vectors.

There are several vectors in Fig. 3 that need to be explained: $s^{\,k}$ is the current search point, v^k is the current velocity, v_{pbest} is the velocity based on $pbest$, s^{k+1} is the corrected search point, v^k is the corrected velocity and v_{gbest} is the velocity based on g_{best}.

PSO commences with a stochastic group of particles and follows optimal points. Each particle is updated according to p_{best} and g_{best}. Each particle attempts to update its own position using the current position, current velocity, distance between current position and p_{best}, and distance between current position and g_{best} as follows:

$$v_i^{k+1} = \left(wv_i^k\right) + \left(c_1 \times rand_1(\ldots) \times \left(pbest_i - s_i^k\right)\right) + \left(c_2 \times rand_2(\ldots) \times \left(gbest_i - s_i^k\right)\right), \quad (22)$$

Where, v_i^k is the velocity of particle i at iteration k, w is an inertia weight [11], c_1 is the cognitive parameter, c_2 is the social parameter, $s_{\,i}^{\,k}$ is the current position of the particle i at iteration k, $rand_1$ and $rand_2$ are stochastic values between 0 and 1, p_{best_i} is p_{best} of particle i and g_{best} represents the best value in a group.

The inertia weight,w, is obtained by

$$w = w_{max} - \frac{(w_{max} - w_{min}) \times iter}{maxiter} \quad , \quad (23)$$

Where, $w_{\,max}$ is the primary weight, w_{min} is the final weight, $maxiter$ is the maximum number of iterations and $iter$ is the number of iterations up to now.

The last step is to update the positions as follows

$$s_i^{k+1} = s_i^k + v_i^{k+1} \quad . \quad (24)$$

The inertia weight w plays a main role in the local search and the global search. A large w helps in the local search, whereas a small w helps in the global search. Fig.4 shows the steps of a basic PSO algorithm of PSO.

The performance of a PSO algorithm depends on several elements:

- Size of the initial population
- c_1, the cognitive parameter and c_2, the social parameter
- w, the inertia weight; It decreases over the iterations
- v_{max}, the maximum velocity of each particle in each dimension
- A fitness function to help find the global optimal solution

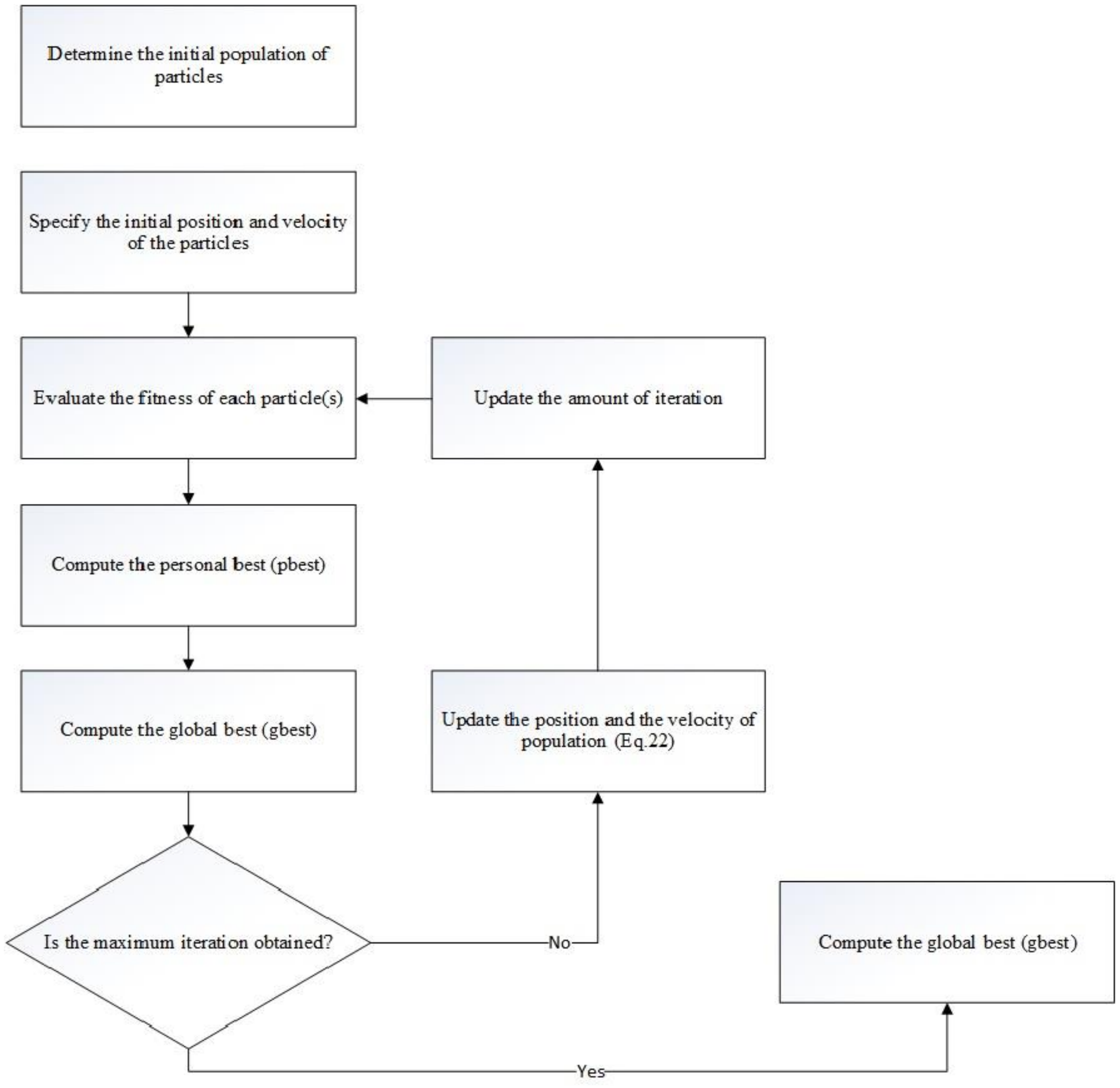
Determine the initial population of particles
Specify the initial position and velocity of the particles
Evaluate the fitness of each particle(s)
Update the amount of iteration
Compute the personal best (pbest)
Compute the global best (gbest)
Update the position and the velocity of population (Eq.22)
Is the maximum iteration obtained?
No
Compute the global best (gbest)
Yes

Figure 4. A flowchart for a basic Particle Swarm Algorithm

These parameters are specified based on the specific problem. The initial phase in the process is to adapt the general algorithm to the problem. By setting proper values for the parameters for the problem, the optimal solution is found conveniently.

5. Simulated annealing

Simulated annealing (SA) was proposed by Kirkpatrick et al. [19] in 1983 as an extension of the Metropolis algorithm [20]. Its name refers to the physical process of annealing in metallurgy. Achieving a minimum energy crystalline structure is a reason for SA technique, and it requires heating and slow cooling of materials. The SA algorithm follows this process with the aim of finding a good solution while providing the opportunity to escape from local optima. The opportunities to move away from local optima depends on the temperature. The more temperature provides the more opportunity. As the process ‘cools’, the focus is on finding an optimal solution, and so the probability of a jump to a new neighborhood is reduced.

The process starts with the highest temperature (T_0), which reduces after each iteration. After an initial solution is generated, a random search is conducted to move from the current solution to a neighborhood solution. The neighborhood range selection is done by the user, and it is very important for the success of an SA algorithm. A new solution with a better objective value will always be accepted. But, a solution with a worse objective value has an opportunity to be accepted based on a probability p, given by $p = e^{-\frac{\Delta}{T}}$, where Δ is the difference between the new solution and the current solution, and T is the current temperature.

The general structure of an SA algorithm for the minimization problem is illustrated as follows:

```
• Specify the SA control parameters, $T_0$, $Mitr$.
• Select the initial solution, $X_0$
• Set $T = T_0$, $X = X_0$, $X^* = X_0$, and $n = 1$
• Compute $f(X_0)$
• While the stop criterion is not met do
• While $n \leq Mitr$ do
      Generate solution $X_n$ in the neighbourhood of $X_0$
      Compute $\Delta = f(X_n) - f(X)$
  If $\Delta \leq 0$ then $X = X_n$
  Else generate a random number $r \in (0,1)$
  If $r \leq \left(p = e^{-\frac{\Delta}{T}}\right)$ then $X = X_n$
  $n = n + 1$
  End if
  End if
  If $f(X) < f(X^*)$ then $X^* = X_n$
  End if
• End while
• Update the temperature $T$
• End while
```

The used notations are:

X_0= initial solution,

X_n= current solution,

X^*= best solution,

$Mitr$= maximum number of iterations,

T_0= initial temperature,

n=repetition counter,

$f(x)$=value of the objective function

As most heuristic algorithms, SA algorithm begins with an initial solution generated randomly. With respect to the initial temperature, the objective value at the initial solution is computed and set as a current solution. This algorithm consists of two loops. In the outer loop, the temperature is updated by a function. The inner loop is iterated while n is less than $Mitr$. In each iteration, a neighborhood solution is generated; the difference between the objective value at a neighborhood solution and the objective value of current solution is calculated and saved into Δ. If Δ less than zero, i.e., if a movement to the neighbor of current solution decreases the objective function, then the current solution is set to be the neighborhood solution. If not, there is a chance for the neighborhood solution to be accepted with a probability $p = e^{-\frac{\Delta}{T}}$. This allows for escaping from local points. Note that the role of T as temperature in this probability. In each iteration, the updated temperature decreases the probability of accepting a worse solution.

Some parameters of SA play main roles in this algorithm. T_0 should be selected properly. A function is used to reduce the temperature. One usually uses $T_i = \varphi T_{i-1}$ where φ, typically between 0 and 100, is considered as cooling rate. A stopping criterion is defined to stop the algorithm and is usually specified as the total number of solutions or pre-specified value of final temperature, say T_f.

6. Applying PSO and SA to Availability Optimization

To optimize the steady-state availability of the system, the codes of the proposed PSO and SA algorithms were written in MATLAB software environment; see the pseudo code for the proposed PSO and SA algorithms are represented in tables 1 and 2.

The fitness function of the problem was represented according to equations (16) to (19) as follows:

$$Fitness\ function = \frac{A_S}{1-\left(\min\left(0,\left(C-\sum_{j=1}^{y}(c_j n_j + h_j r_j)\right)\right)+\min\left(0,\left(W-\sum_{j=1}^{y} w_j n_j\right)\right)+\min\left(0,\left(V-\sum_{j=1}^{y} v_j n_j\right)\right)\right)} \quad .(25)$$

This fitness function was constructed based on [12] and was used to guarantee the feasible solution generated from the proposed heuristic algorithms. Before demonstrating the performance of the proposed heuristic algorithms, we set the parameters of the algorithms.

Table 1. A pseudo code of our proposed PSO algorithm

```
1. Determine $y$, $maxiter$ (maximum iteration), $partnum$ (the size of primary
   population), $k_j$, $c_j$, $w_j$, $v_j$, $h_j$, $C$, $W$, $V$, $\lambda$ , $\mu$ and $interval\ n_j$ (maximum number of
   components in subsystem $j$).
2. Determine the parameters of PSO, i.e., $w_{max}$, $w_{min}$, $v_{max}$, $c_1$ and $c_2$ .
3. Compute the $interval\ r$ as follows:
                    $interval\ r_j = interval\ n_j - k_j + 1$.
4. For $i$ =1 to $partnum$ do ( generation of the initial population)
   4.1. For $j$ =1 to $y$ do
        4.1.1. Select a random number between $k_j$ and $interval\ n_j$, and store it in
               $n(i,j)$.
        4.1.2. Select a random number between 1 and $n(i,j)-k_j+1$ and store it in $r(i,j)$.
    4.2. End for
    4.3. Set $Pbest(i)=0$ and $fpbest(i)=0$.
5. End for
6. Set BA (best availability) = 0 and BP (best point) = 0.
7. For $l$=1 to maxiter do
    7.1. Update the inertia weight by using equation (23).
    7.2. For $i$=1 to partnum do
        7.2.1. Compute $fitness\ (n(i),r(i))$.
        7.2.2. If $fpbest(i) \le fitness\ (n(i),r(i))$ then $pbest(i)=(n(i),r(i))$, $fpbest\ (i) = fitness\ (n(i),r(i))$.
        7.2.3. Update the velocity of each point by using equation (22).
        7.2.4. Update the position of each point by using equation (24).
    7.3. End for
    7.4. Compute the greatest $fpbest\ (i)$ and place it into $fgbest(l)$ and set $gbest(l)=pbest(i)$.
    7.5. If $Best \le fgbest(l)$ then set BA= $fgbest(l)$ and BP =$gbest(l)$.
8. End for
9. Write BA and BP.
```

Table 2. A pseudo code of our proposed SA algorithm

```
1. Determine $y$ , $Mitr$ (maximum number of iteration), $c_j, w_j, v_j, h_j, C, W, V, \lambda, \mu$ and $n_{\max}$ (maximum number of components in each subsystem).
2. Determine parameters of SA, i.e., $T_0$, $T_f$ and $\varphi$.
3. Generate random points $x_0 = [n_1, n_2, ..., n_y, r_1, r_2, ..., r_y]$ as an initial solution.
4. Compute the fitness function of initial solution, $f(x_0)$.
5. While $T \succ T_f$ do
    5.1. For i=1 to $Mitr$ do
        5.1.1. Generate neighborhood values of initial solution, say $x = [n_1^1, n_2^1, ..., n_y^1, r_1^1, r_2^1, ..., r_y^1]$.
        5.1.2. Calculate the fitness function of neighborhood solution, $f(x)$.
        5.1.3. If $f(x) \succ f(x_0)$ then set $x_0 = x$ and $f(x) = f(x_0)$
        5.1.4. Else generate a random number, $r \in (0,1)$.
            5.1.4.1 If $r \le e^{-\frac{[f(x)-f(x_0)]}{T}}$ then set $x_0 = x$ and $f(x_0) = f(x)$.
        5.1.5. End if
        5.1.6. End for
```

```
6. Set $T = \varphi T$ .
7. End while
8. Write $x_0 = [n_1, n_2, ..., n_y, r_1, r_2, ..., r_y]$ and $f(x_0)$ .
```

7. Parameter setting

As mentioned in Section 3, several parameters such as v_0, w_{min}, w_{max}, c_1 and c_2 affect the performance of PSO algorithm and parameters φ , T_0, and T_f affect the performance of the SA algorithm. Good solutions can be obtained by proper selection of these parameters. We used Design of Experiments (DoE) to determine the values of PSO and SA parameters maximizing the availability function. Three levels of low, medium, and high were considered for each parameter as follows: v_0: (0.02, 2, 20), c_1 and c_2: (2, 10, 120), w_{min}: (0.1, 2, 30), and w_{max}: (0.5, 3, 50) for PSO and φ: (0.45, 0.75, 0.95), T_0: (10, 100, 1000), and T_f: (0.001, 0.1, 1) for SA. These levels were selected based on many runs and various settings. To determine a good combination of these values, we ran 3^4 experiments for the PSO algorithm and 3^3 for the SA algorithm, but to reduce the number of experiments we utilized the Taguchi method. The number of experiments were decreased to 9 by using L_9 orthogonal array. Actually, 9 discrepant experiments with three replications were used. Availability was used as a measure to set the parameters. The parameter setting process was carried out based on the second problem instance, having four subsystems. The results are presented in tables 3 and 4.

Table 3. Experimental results of using L_9 for PSO parameters

Experiment No.	v_0	c_1 and c_2	w_{min}	w_{max}	A_s	Average of A_s	S/N
1	0.02	2	0.1	0.5	0.656 0.644 0.648	0.650	−3.741
2	0.02	10	2	3	0.500 0.443 0.490	0.478	−6.445
3	0.02	120	30	50	0.371 0.367 0.338	0.359	−8.917
4	2	2	2	50	0.420 0.469 0.413	0.434	−7.275
5	2	10	30	0.5	0.308 0.296 0.302	0.302	−10.393
6	2	120	0.1	3	0.365 0.361 0.345	0.357	−8.943
7	20	2	30	3	0.386 0.412	0.403	−7.899

					0.411		
8	20	10	0.1	50	0.369 0.354 0.376	0.366	−8.723
9	20	120	2	0.5	0.492 0.521 0.523	0.512	−5.819

The average values of availability function for each parameter at each level was calculated and the results are summarized in tables 5 and 6. The quality characteristic analyzed in this study was "the-bigger-the-better", i.e., maximization of availability function. Fig.5 illustrates the main effect plot for each average availability function.

In the Taguchi method, the Signal to Noise (S/N) ratio is used to evaluate the sensitivity of the analyzed quality to error in the experiment. The greater S/N ratio indicating smaller variance around the target value, the higher value of S/N ratio is noted.

Table 4. Experimental results of using L_9 for SA parameters

Experiment No.	φ	T_0	T_f	A_s	Average of A_s	S/N
1	0.45	10	0.001	0.495 0.485 0.462	0.481	−14.64
2	0.45	100	0.1	0.512 0.506 0.523	0.514	−13.30
3	0.45	1000	1	0.501 0.499 0.508	0.503	−13.73
4	0.75	10	0.1	0.475 0.456 0.455	0.462	−15.44
5	0.75	100	1	0.519 0.526 0.502	0.516	−13.23
6	0.75	1000	0.001	0.526 0.531 0.489	0.515	−13.28
7	0.95	10	1	0.486 0.448 0.498	0.478	−14.81
8	0.95	100	0.001	0.590 0.591 0.578	0.586	−10.66
9	0.95	1000	0.1	0.512 0.523	0.510	−13.45

				0.495		

Table 5. Level average for main effects associated with PSO

Parameters	Low	Medium	High
v_0	0.495	0.364	0.427
c_1 and c_2	0.496	0.382	0.409
w_{min}	0.458	0.475	0.354
w_{max}	0.488	0.412	0.386

Table 6. Level average for main effects associated with SA

Parameters	Low	Medium	High
φ	0.499	0.498	0.525
T_0	0.473	0.539	0.509
T_f	0.527	0.495	0.499

To calculate the S/N ratio, Mean Square Deviation (MSD) for “the-bigger-the-better” quality characteristics were obtained using the following equation:

$$MSD = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{A_{s,i}^2} \quad , \tag{26}$$

$$S/N = 10\log_{10}(MSD) \quad , \tag{27}$$

where $A_{s,i}$ is the availability function value for *i*th experiment. The S/N ratios for all the experiments were calculated as represented in tables 3 and 4. As seen in tables 3 and 4, the experiment number 1 for PSO and the experiment number 8 for SA yield the largest ratios. In parallel, we can determine the best level of each parameter by the main effects plot of availability function as shown in Fig. 5 and Fig. 6.

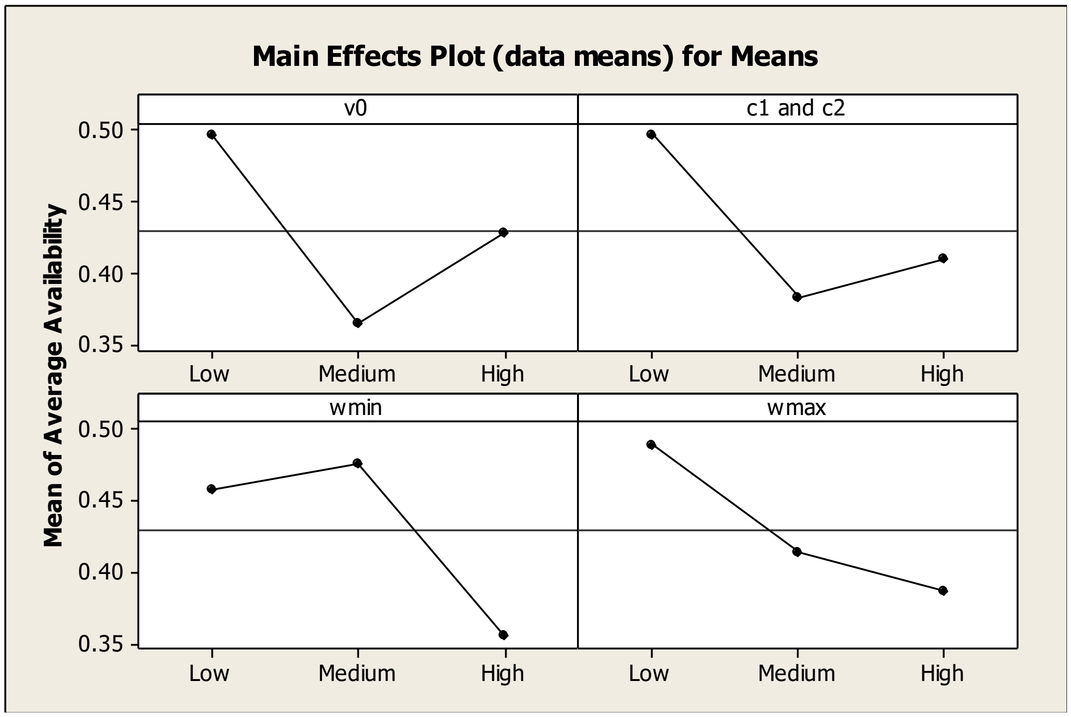

Figure 5. Main effects plot on availability function for parameters of PSO

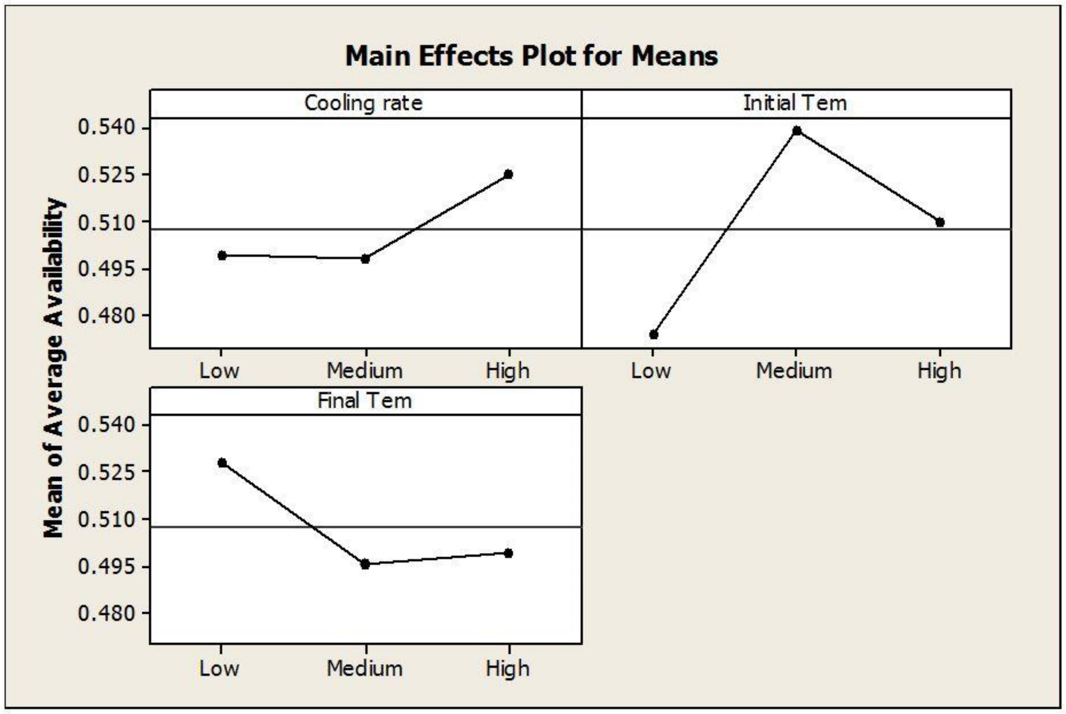

Figure 6. Main effects plot on availability function for parameters of SA

As seen in Fig. 5, the low level of v_0, the low levels of c_1 and c_2, the low and medium level of w_{min}, and the low level of w_{max} can be considered as the best parameters levels for PSO and considering the value of *S/N* (i.e., −3.741) in Table 3, the combination of the first experiment can be chosen. It can be seen in Fig.6 that the high level of cooling rate, the medium level of initial temperature, and the low level of final temperature have the most effect on the SA algorithm and considering the *S/N* value (−10.6643) in Table 4, the combination in experiment 8 given an effective parameter levels.

Table 7. The result of ANOVA associated to PSO

Parameters	DOF	Sum of Squares	Variance	F	$F_{0.05,2,18}$	Contribution (%)
v_0	2	0.076	0.038	115.83	3.55	28.06
c_1 and c_2	2	0.062	0.031	95.18		23.01
w_{min}	2	0.075	0.038	114.25		27.67
w_{max}	2	0.049	0.024	74.90		18.06
Error	18	0.005	0.0003			3.17

The ANalysis Of VAriance (ANOVA) provides statistical results illustrating the significant factors. We used ANOVA to identify the effective factors. Tables 7 and 8 show the results of ANOVA. As seen in Table 7, all four factors for PSO show the same contribution, and Table 8 shows that T_0 is more effective than the other factors.

Table 8. The result of ANOVA associated to SA

Parameters	DOF	Sum of Squares	Variance	F	$F_{0.05,2,18}$	Contribution (%)
φ	2	0.003	0.002	6.79	3.55	10.29
T_0	2	0.018	0.009	31.29		53.81
T_f	2	0.005	0.002	9.20		14.56
Error	18	0.005	0.0003			21.31

8. Numerical experiments

Seven problem instances were randomly generated to compare the performance of the proposed PSO and SA algorithms. These seven problems cover discrepant subsystems of various sizes, i.e., 3, 4, 5, 6, 7, 8, and 9 subsystems. All instances were coded in MATLAB on the Intel Core 2, CPU 2.66 and 2.67 GHz PC. Parameters of the problems including various k-out-of-n subsystems are shown in Table 9.

Table 9. Input parameters of the algorithms

J	*y*	***k***	c_j	w_j	v_j	h_j	***C***	***W***	***V***
1	3	(2,1,2)	(5,4,7)	(0.3,0.4,0.2)	(1.1,1.3,2.1)	(1,1,2)	480	45	40
2	4	(2,1,2,3)	(5,4,7,6)	(0.3,0.4,0.27,0.7)	(1.1,1.3,2.1,3.1)	(1,1,2,2)	261	35	55
3	5	(4,2,3,2,1)	(18,13,12,24,22)	(2.12,3.11,2.76,2.34,2.18)	(64.34,72.12,72.87,77.11,72.18)	(63,55,52,52,53)	42267	49.04	31897
4	6	(2,3,3,2,2,3)	(45,65,42,34,36,33)	(11.2,10.21,12.22,13.14,11.45,10.23)	(20.43,22.32,21.17,27.33,22.28,24.21)	(113,155,212,42,63,134)	38672	271.66	11897
5	7	(3,2,4,2,3,2,2)	(21,34,24,19,23,26,43)	(6,5,8,5,8,9,10)	(23.13,29.8,31,42,53.2,71.6,15.4)	(111,132,123,108,132,131,102)	52454	157	10456
6	8	(4,2,3,2,4,3,1,2)	(18,16,13,18,13,12,24,22)	(12,15,11,22,11,26,34,41)	(46.31,52.31,21.34,32.44,42.22,47.23,66.14,71.12)	(41,33,38,51,29,52,53,12)	76540	475	65897
7	9	(2,1,3,4,2,2,3,2,1)	(98,42,75,32,69,43,55,87,38)	(21,11,14,25,23,24,31,22,32)	(43,32,17,33,28,21,31,42,39)	(322,243,423,142,93,265,112,213,87)	68423	2897	734

Due to the complexity of the objective function of our model, the exact optimal solution could not be obtained by codes such as GAMS and LINDO. We had to find a way to obtain the exact optimal solution. The only way was to check all points in the solution space one by one. We coded the point-by-point searching method in MATLAB. However checking all points was not logical for large problems. We noted the elapsed time by the three proposed methods, PSO, SA and Exact optimal method to demonstrate that our heuristic methods are more efficient than the point-by-point search method.

Availability function is the main measure to the compare the two heuristic algorithms. To show the effectiveness of the proposed PSO algorithm, the closeness of the solution to the exact optimal solution was considered. Because of the stochastic nature of the proposed PSO and SA algorithms, 100 independent runs were carried out for all the instances and the maximum availability value was considered in all instances. After many runs and various settings of population size for our scenarios, the population size 25 was considered for both heuristic algorithms.

Parameters of the PSO and SA algorithms were considered based on the mentioned parameter setting approach and the results are shown in tables 10 and 11.

Table 10. Input parameters of the proposed PSO algorithm

w_{max}	w_{min}	c_1	c_2	v_0
0.5	0.1	2	2	0.02

Table 11. Input parameters of the proposed SA algorithm

φ	T_0	T_f
0.95	100	0.001

The seven instances were solved by the three mentioned methods and the results are summarized in table 12.

As shown in table 12, it is not proper to investigate all the points in the feasible space to find the optimal point. Desiring more availability results in a larger feasible space. It is seen that the first example with three subsystems takes 2.33 hours for 92% availability, whereas the second example with four subsystems takes 24.681 hours only for 66% availability. It is also seen that for five subsystems with only 4% availability, it has taken 6.45 hours and so on. Therefore an efficient straightforward heuristic algorithm can speed up the process. The elapsed times for PSO and SA algorithms presented in table 12 shows that the two heuristic algorithms need much shorter time in comparison with the exact optimal solution approach. For instance, for the first instance, the three subsystems, the elapsed time for the exact optimal solution is 2.33 hr, while the PSO algorithm needs 0.0058 hr and the SA algorithm needs 0.0023 hr.

As mentioned earlier, availability function is the main measure to compare PSO and SA algorithms with the exact optimal solution, but the elapsed times show the efficiency of the heuristic algorithms. The elapsed times for the two heuristic algorithms are so close to each other, but in comparison with the results obtained by the proposed exact method.

It is quite clear from table 12 that the maximum availability of the system corresponding to all problems is so close to the exact optimal solution.

Fitness convergences of the proposed PSO and SA algorithm for all problems are illustrated in Fig. 7. It is clear that the proposed PSO instances and SA instances strongly converged to their optimal solutions.

The statistical results of 100 runs for seven instances are presented in table 13. Fig. 8 and Fig.9 respectively denote the difference between the average availability and standard deviation of availabilities of PSO and SA algorithms. We can interpret Fig. 8 from two viewpoints.

Table 12. Results

J	Exact Optimal Solution			PSO Solution			SA Solution		
	(n_1, n_2,…, r_1, r_2,…)	Availability	Elapsed Time	(n_1, n_2,…, r_1, r_2,…)	Availability	Elapsed Time	(n_1, n_2,…, r_1, r_2,…)	Availability	Elapsed Time
1	(2,6,9,11,6,8)	0.9234	2.33 hr	(10,5,6,7,5,5)	0.923	0.005 hr	(9,8,9,8,8,8)	0.915	0.0023 hr

2	(8,5,7,8, 7,5,6,6)	0.6570	24.681 hr	(9,17,4, 4,3,8,3,2)	0.656	0.001 hr	(2,1,2,3, 1,1,1,1)	0.591	0.0012 hr
3	(7,3,4,4, 2,4,2,2,3 ,2)	0.0483	6.45 hr	(7,3,4,4, 2,4,2,2,3 ,2)	0.048	0.007 hr	(7,2,5,4, 2,4,1,3,3 ,2)	0.037	0.0083 hr
4	(3,5,5,3, 3,5,2,3,3 ,2,2,3)	0.0187	31.73 hr	(3,5,5,3, 3,5,2,3,3 ,2,2,3)	0.018	0.009 hr	(4,5,5,2, 3,5, 3,3,3,1,2 ,3)	0.014	0.125 hr
5	(4,3,5,3, 3,2,2,2,2 ,2,2,1,1, 1)	0.0004	3.12 hr	(3,3,4,4, 3,2,3, 1,2,1,3,1 ,1,2)	0.0003	0.013 hr	(4,2,4,2, 4,2,3, 2,1,1,1,2 ,1,2)	0.0002	0.0143
6	(5,3,4,3, 6,3,1,2,2 ,2,2,2,3, 1,1,1)	0.0003	5.59 hr	(8,8,8,8, 8,8,5,7, 5,7,6,7,5 ,6,5,6)	0.0002	0.021 hr	(9,9,9,8, 9,9,1,7, 6,8,7,7,6 ,7,1,6)	0.0002	0.0216 hr
7	(2,2,3,4, 2,4,3,2,2 ,1,2,1,1, 1,3,1,1,2)	0.0004	9.26 hr	(8,6,8,8, 8,8,8,8,5 ,7,6,6,5, 7,7,6,7,5)	0.0002	0.025 hr	(7,5,7,7, 7,7,7,7,5 ,6,5,5,4, 6,6,5,6,5)	0.0001	0.0297 hr

The first consideration is the closeness of the proposed PSO results to the exact optimal solutions, and another view is the lower differences of the PSO results with the exact optimal solution in contrast to the SA results with the exact optimal solutions. In terms of small standard deviations presented in table 13 and the so trivial standard deviation of results of PSO as compared to SA, the proposed PSO algorithm is sufficiently efficient.

Table 13. Statistical results of 100 runs for PSO and SA algorithms

Problem instance	Heuristic method	Maximum	Minimum	Mean	Standard Deviation
1	PSO	0.923473232	0.923379913	0.923404471	0.000045
	SA	0.915524481	0.856265432	0.896693436	0.021628
2	PSO	0.657091464	0.656884645	0.657700453	0.000102
	SA	0.591386928	0.549785457	0.582157771	0.013435
3	PSO	0.048307605	0.039235201	0.045872123	0.003152
	SA	0.037886510	0.026544502	0.028227788	0.003839
4	PSO	0.018721927	0.015454514	0.018357371	0.000577
	SA	0.014000875	0.009765326	0.010716106	0.001477
5	PSO	0.000324939	0.000284686	0.000299213	0.000013
	SA	0.000272210	0.000114856	0.000191981	0.000049
6	PSO	0.000286257	0.000198745	0.000263533	0.000023
	SA	0.000212707	0.000098564	0.000119304	0.000034
7	PSO	0.000228049	0.000112893	0.000205959	0.000022
	SA	0.000164702	0.000074236	0.000092356	0.000023

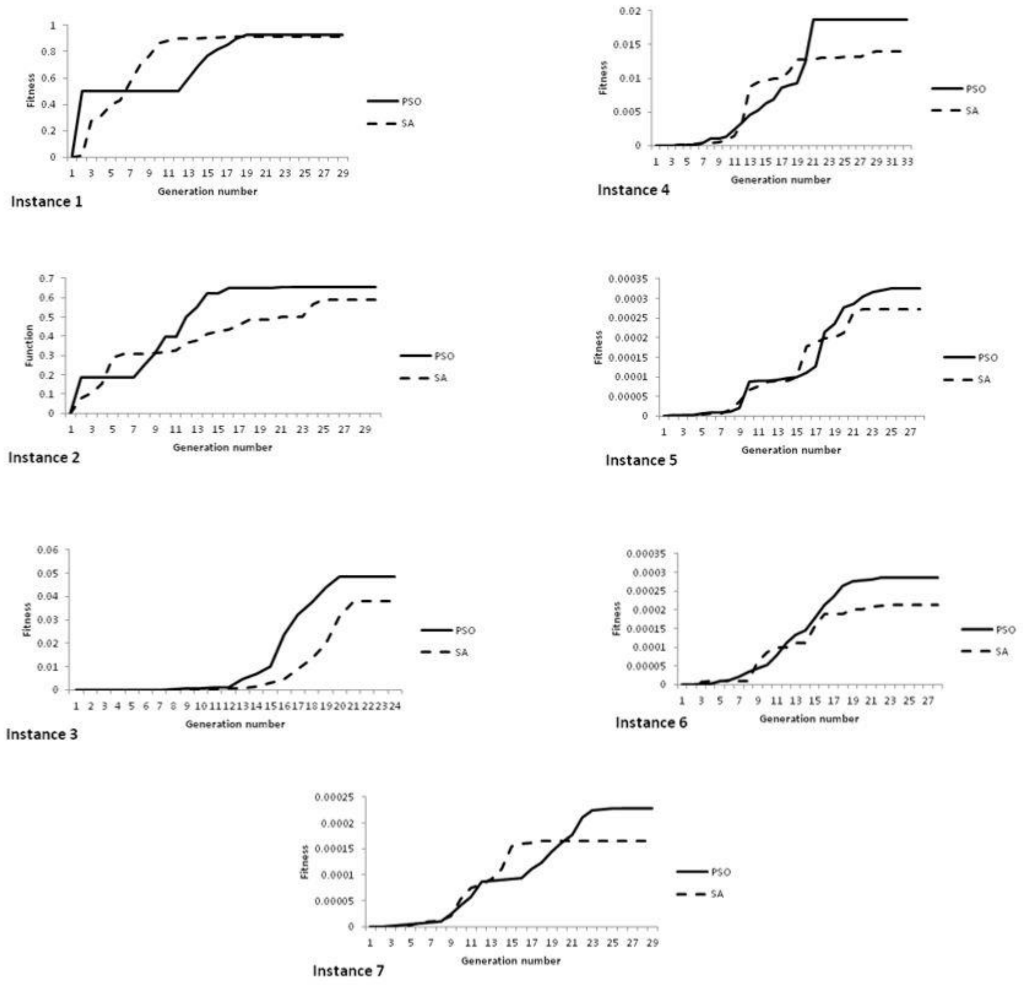

Figure 7. Fitness convergence of the proposed PSO and SA algorithms

Considering the presented results we demonstrated the effectiveness of our heuristic algorithms we also realized the efficiency of the PSO algorithm as compared to a SA algorithm by the differences between their availability functions. Finally, the convergence of the heuristic algorithms and the trivial deviation of availability in 100 runs showed the efficiency of the PSO algorithm compared to the SA algorithm.

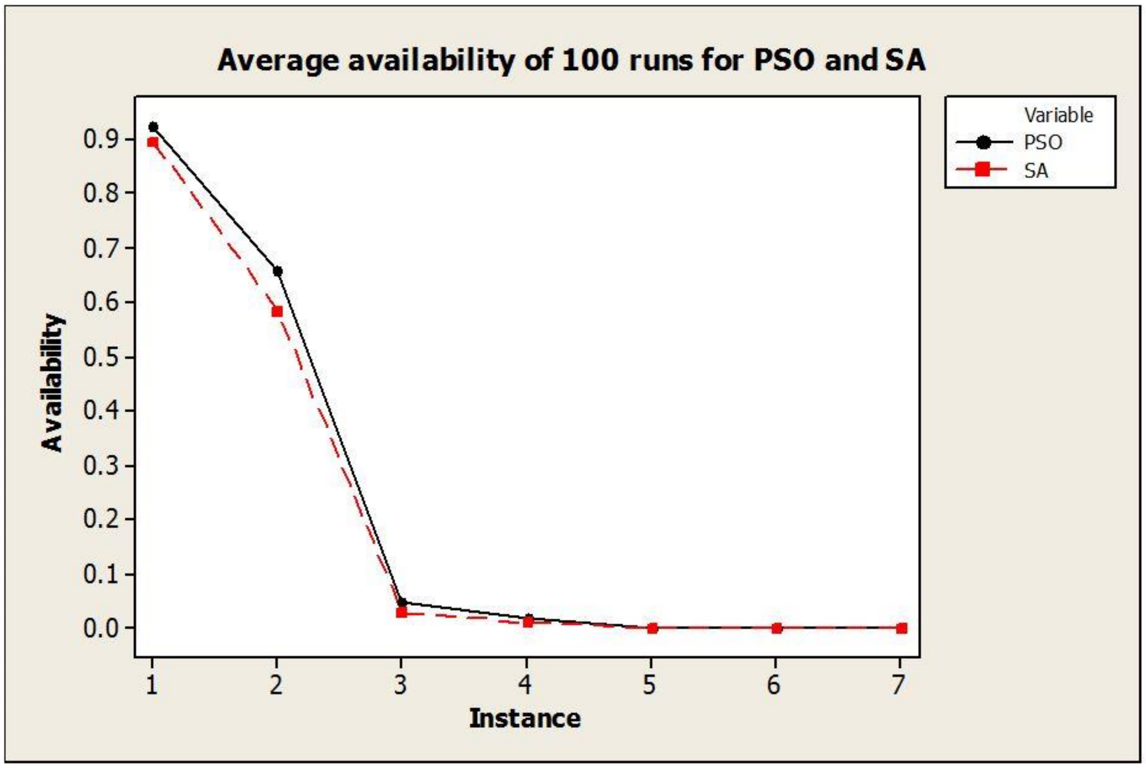

Figure 8: Average availability of 100 runs for PSO and SA algorithms

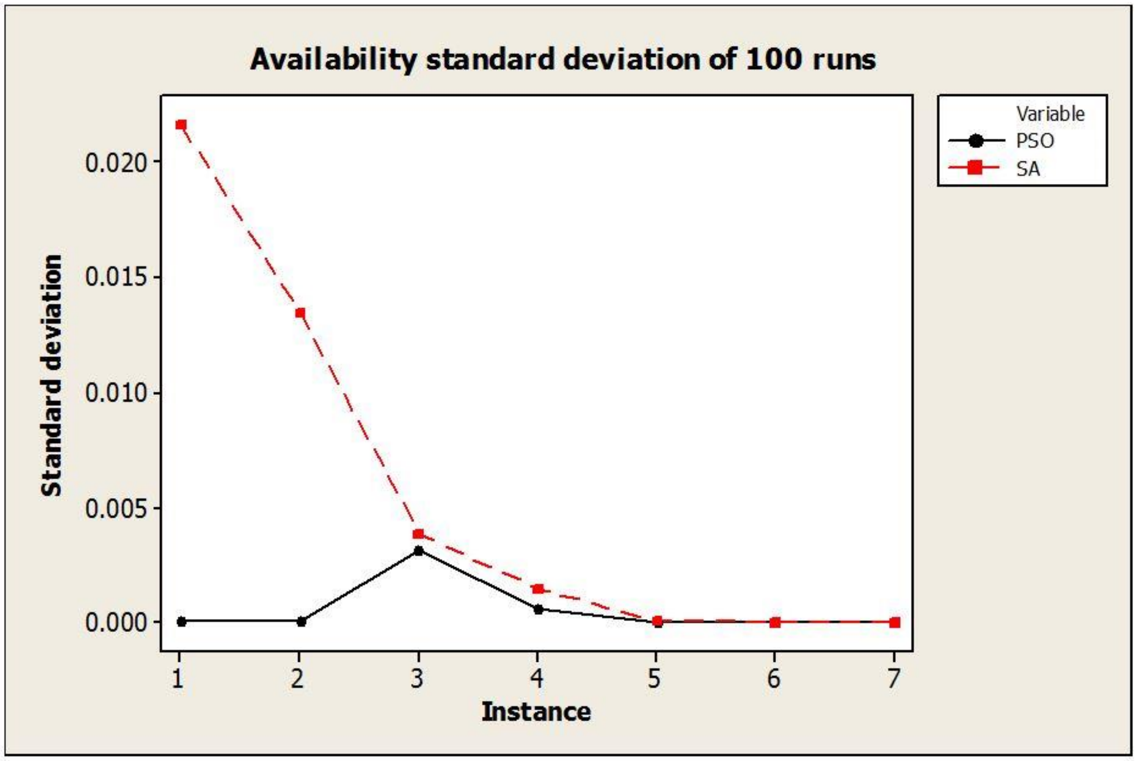

Figure 9. Standard deviation of availability for 100 runs

9. Conclusion and future research

Availability optimization of a system structure consisting of a series with multiple k-out-of-n subsystems were investigated. The main contribution of our work was to consider the number of repairmen as a variable. The structure was modeled and seven various instances were solved. Results indicated two main points: the first one is that point-by-point base method is not appropriate, when we can use a heuristic method quite effectively. Two well-known heuristic algorithms, i.e., PSO and SA algorithms were used to solve the problems and it was shown that PSO is more efficient than SA for our problems.

Because of different applications of this structure with different numbers of subsystems, the proposed PSO was illustrated to be efficient. The convergence of the proposed algorithm and the numerical results of 100 independent runs have been shown on seven instances. The results showed small standard deviations. Some assumptions of the problem could be relaxed. The assumption of all components in being active can be replaced by other states; cold-standby or warm-standby. Two-state problem was considered in our work, multi-state problem can be considered for future investigation. The problem can be solved by other heuristic and meta-heuristic methods and compared with our proposed algorithms. Also, our proposed approach can be used to solve problems with other distributions such as Weibull distribution or Gamma distribution.

Reference

[1] Barron, Y., Frostig, E. and Levikson, B. (2006), Analysis of r out of n systems with several repairmen, exponential life times and phase type repair times: an algorithmic approach, *European Journal of Operations Research*, 169(1), 202-225.

[2] Bulfin, R.L. and Liu, C.Y. (1985), Optimal allocation of redundant components for large systems, *IEEE Transactions on Reliability*, 34, 241–247.

[3] Frostig, E. and Levikson, B. (2002), On the availability of an r out of n repairable systems, *Naval Research Logistics*, 49 (5), 483-498.

[4] Fyffe, D.E., Hines, W.W. and Lee, N.K. (1968), System reliability allocation and a computational algorithm, *IEEE Transactions on Reliability*, 17, 74–79.

[5] Gen, M. and Yun, Y.S. (2006), Soft computing approach for reliability optimization: state-of-the-art survey, *Reliability Engineering and System Safety*, 91(9), 1008-1026.

[6] Huffman, D., Bergman, R., Amari, S.V. and Zuo, M.J. (2008), Availability analysis of systems with suspended animation, *Conference on Reliability Maintenance, Las Vegas, USA*, 283-288.

[7] Kennedy, J. and Eberhart, R.C. (1997), A discrete binary version of the particle swarm algorithm, *International Conference on Systems, Man, and Cybernetics, Piscataway, USA*, 4104-4109.

[8] Khalil, Z.S. (1985), Availability of series systems with various shut-off rules, *IEEE Transaction on Reliability*, 34 (2), 187-189.

[9] Kuo, W., Prasad, V.R., Tillman, F.A. and Hawang, C. (2001) , Optimal Reliability Design Fundamental and Application, Cambridge University Press, UK.

[10] Kuo, W. and Zuo, M.J., (2003), Optimal Reliability Modelling - Principles and Applications, John Wiley & Sons, New York, USA.

[11] Shi, Y. and Eberhart, R. (1998), Parameter selection in particle swarm optimization, Proceedings of Evolutionary Programming, pp 591-600 (1998).

[12] Ta-Cheng Chen., (2006), IAs based approach for reliability redundancy allocation problems, *Applied Mathematics and Computation*, 182 (2), 1556-1567.

[13] Ying-Shen Juang, Shui-Shun Lin b. and Hsing-Pei Kao., (2008), A knowledge management system for series-parallel availability optimization and design, *Expert System with Applications*, 34 (1), 181-193.

[14] Khatab, A., Nahas, N. and Nourelfath, M., (2009), Availability of k-out-of-n:G systems with non-identical components subject to repair priorities, *Reliability Engineering and System Safety*, 94 (2), 142-151.

[15] Krishnan, R., Somasundaram, S., (2011), Reliability analysis of repairable consecutive-k-out-of-n: G systems with sensor and repairmen, *International Journal of Quality and Reliability Management*, 28 (8), 894-908.

[16] Moghaddass, R., Zuo, M.J., Qu, J., (2011), Reliability and availability analysis of a repairable k-out-of-n : G System with R repairmen subject to shut-off rules, *IEEE Transactions on Reliability*, 60 (3),658-666.

[17] Chern MS, (1992), On The computational complexity of reliability redundancy allocation in a series system. *Operations Research Letters* 11:309-315.

[18] W. Kuo and M. J. Zuo, (2003), Optimal Reliability Modeling: Principles and Applications. New York: John Wiley & Sons.

[19] Kirkpatrick S., Gelatt CD, Vecchi MP, (1983) Optimization by simulated annealing. *Science*, 220, 671-680.

[20] Metropolis N, Rosenbluth MN, Teller AH, (1953), Equation of state calculations by fast Computing Machines, *Journal of Chemical Physics*, 21,1087-1093.