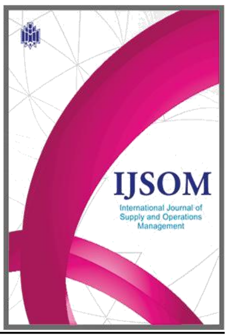
International Journal of Supply and Operations Management

IJSOM

August 2018, Volume 5, Issue 3, pp. 218-233
ISSN-Print: 2383-1359
ISSN-Online: 2383-2525
www.ijsom.com



An Adjusted Water Cycle Algorithm for Solving Reliability-redundancy Allocation Problems with Cold-standby Components

Mohammad N. Juybari [a], Mostafa Abouei Ardakan*, [a], Hamed Davari-Ardakani [a]

[a] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*

Abstract
One of the most practical methods for improving system reliability is making a tradeoff between components reliability and redundancy levels, which is known as reliability-redundancy allocation problem (RRAP). The RRAP aims to maximize the overall system reliability by creating a balance between the component reliabilities and the number of redundant components in each subsystem. In the RRAP, the redundant components are performed in a predetermined order under a redundancy strategy. In this paper, a cold standby redundancy strategy is considered for the redundant components. Besides, a penalty guided water cycle algorithm is adjusted for solving the problem. The proposed algorithm is implemented on two famous benchmark problems to evaluate the performance of the proposed approach. The obtained numerical results reveal the superiority of the proposed solution method over all previous studies.

Keywords: Reliability-redundancy allocation problem; Cold-standby strategy; Reliability optimization; Water cycle algorithm.

1. Introduction

Most of the catastrophic incidents during the last century might imply that failures and their consequences are due to poor reliability of their scales. The explosion of two NASA's space shuttles Challenger and Columbia in 1986 and 2003, respectively, are obvious examples that failure of a system can have widespread effect on the total expected mission of the system. Another example of the role of reliability in structural design is the biggest nuclear disasters in the world, like the explosion of four operating nuclear reactors at the Chernobyl site in 1986 (Elsayed, 2012). These examples signify that reliability plays a critical role in industries and has far reaching effect on consumers of services. The formal definition of reliability is the probability that a service or product will operate without failures for a definite time interval under the specified operating stresses. In other words, reliability is the system's success measurement in supporting its function perfectly during its specified period of time or its design life (Elsayed, 2012).

Reliability optimization problem (ROP) is an important topic that has attracted the interest of many academic and applied engineering studies. Two different categories of ROP are: (a) Redundancy Allocation Problem (RAP) and (b) Reliability-Redundancy Allocation Problem (RRAP). In the RAP, the component choices and their characteristics such as reliability, cost and weight are predetermined. The goal of the RAP is to find the optimal number of redundant components in each subsystem in order to maximize the overall system reliability whereas in the RRAP, the component reliability is a design variable and its characteristics are computed as increasing nonlinear functions of component reliability (Ardakan & Hamadani, 2014a). The RRAP is formulated as a Mixed Integer Nonlinear Programming (MINLP) problem. Therefore, solving such a complex problem is very difficult especially on a large scale. In this paper the RRAP is considered.

Corresponding author email address: Abouei@khu.ac.ir

A redundancy strategy determines the way of using redundant components in a subsystem. Generally, there are two traditional redundancy strategies called active and standby. In the active strategy, all the redundant components start their operation from time zero. There are three alternatives to the standby strategy, namely cold, warm and hot. In the cold standby strategy, one component is in active mode and the rest of the components are kept idle and protected from operational stresses. In this strategy, when the active component fails, the first redundant component is activated and starts its mission as a new one. But in the warm standby strategy, although just one component is needed, all redundant ones are somewhat affected by the operational stresses. Finally, in the hot standby strategy, the component failure is not influenced by whether it was in operating mode or in idle model. As an extension to redundancy strategies, Ardakan and Hamadani (2014b) introduced a new redundancy strategy for adding redundant components to subsystems in the RAP, which is called ‘mixed’ strategy. This strategy is a comprehensive model of both active and standby strategies. Abouei Ardakan et al. (2016) implemented the mixed strategy in the RRAP, which is more complicated than the RAP. Peiravi et al. (2017) formulated a general form of the mixed strategy and introduced “k-mixed”. This novel redundancy strategy utilizes concepts of k-out-of-n and mix strategy, simultaneously. They implemented k-mixed in the RAP.

Generally, in standby strategies, a switching system is needed for replacing the failed component by a new one (Tavakkoli-Moghaddam, Safari, & Sassani, 2008). System designers consider two possible scenarios for switching mechanisms. In the first suggested scenario (S1), the failure detector scans the system performance in order to find any failure in the active components and replace them by a redundant one. In this scenario, the switch reliability is considered a non-increasing function of time ($\rho_i(t)$). On the other hand, in the second scenario (S2), the probability of switch failure is considered a constant value $\rho_i(t)$.. In this case, the switch failure happens only when it is used (Coit, 2001).

Kim (2018) studied optimal reliability design for a RAP system with mixed components and imperfect switching policy. A Structured Markov chain was used to calculate each subsystem’s reliability. Most of the researchers formulated the RRAP only by considering the active redundancy strategy. Both exact and meta-heuristic methods were applied to the problem. Exact optimization methods including dynamic programming (Kuo, 2001) and branch and bond (Kuo, Lin, Xu, & Zhang, 1987) have been used to solve the problem. Hikita et al. (1992) developed a dynamic programming approach to solve the RRAP by a single constrained surrogate method. However, most of traditional methods failed to solve the problem in a reasonable time, due to high-dimension of the RRAP. Hence, researchers applied meta-heuristic methods to solve the reliability optimization problem (Wang & Li, 2012). In the field, many meta-heuristic and nature inspired algorithms such as genetic algorithm (Ardakan, Hamadani, & Alinaghian, 2015; Ramirez-Marquez, Coit, & Konak, 2004; Tavakkoli-Moghaddam et al., 2008), artificial neural networks (Habib, Alsieidi, & Youssef, 2009), particle swarm optimization (dos Santos Coelho, 2009), ant colony optimization (Liang & Smith, 2004; Nahas & Nourelfath, 2005) have been developed for solving different RRAP problems.

For solving the RRAP with the active redundant strategy, a two-phase approach based on an Immune Algorithm (IA) has been developed by Hsieh and You (2011) . They implemented the IA in the first phase and developed a new procedure to improve the final solution in the second phase. For the same problem, a hybrid algorithm which is a combination of Harmony search and Differential Evolution algorithm was introduced by Wang and Li (2012). Yeh and Hsieh (2012) developed a modified Artificial Bee Colony (ABC) for the RRAP problems. Results showed that the proposed approach leads to better solutions compared to those of other meta-heuristic algorithms. A new version of Harmony search (HS) called EGHS was proposed by Zou et al. (2011) to solve the RRAP problem. The concept of EGHS is inspired by mixing the HS algorithm and the concept of swarm intelligence in the particle swarm optimization (PSO) algorithm. In another study, for solving the RRAP problems, a novel particle swarm optimization algorithm called IPSO was introduced by Wu et al. (2011).

Afonso et al. (2013) used an Imperialist Competitive Algorithm (ICA) for the RRAP problems. Valian and Valian (2013) modified the Cuckoo Search (CS) algorithm for the RRAPs and tested the performance of the proposed algorithm in different benchmark problems. Valian et al. (2013) improved the convergence rate and accuracy of the CS and compared its performance in RRAP problems with the results of other studies. A new version of the Stochastic Fractal Search (SFS) algorithm called PSFS was represented by M.A.Mellal and E.Zio (2016) to solve the reliability optimization problems, including the RRAP with active components. The idea of PSFS was inspired by adding a penalty function to the SFS algorithm. In most practical optimization problems, handling of conflicting objectives is a serious concern. Ardakan and Rezvan (2017) formulated the RRAP as a bi-objective problem by considering system cost and total reliability as two opposite goals.

It should be noted that all the above-mentioned studies considered the RRAP with the active redundancy strategy. For the first time, Ardakan and Hamadani (2014a) developed the RRAP with a standby redundancy strategy and solved the problem by using a penalty guided GA. The outcomes of their study confirmed that the standby redundancy strategy outperforms the active one.

In the present study, the RRAP is considered with the cold-standby strategy for redundant components. A recently introduced algorithm called Water Cycle Algorithm (WCA) (Eskandar et al., 2012) is adjusted and implemented on the problem. The rest of the paper is organized as follows. In Section 2, the formulation of the RRAP with the cold standby strategy and two benchmark problems are presented. Section 3 presents the WCA for solving the proposed non-linear problems. The efficiency of the WCA in the RRAP is presented in Section 4. Finally, Section 5 contains the conclusions.

Notations:

t:	total considered time
m:	total number of subsystems
λ_i :	exponential distribution factor which indicates the failure rate of components in subsystem i; $1 \le i \le m$
$\rho_i(t)$, ρ_i :	Switch reliabilities at given time t for the first and second switching scenarios, respectively.
n:	**n**=(n_1,n_2,...,n_m) is a row vector of redundancy level representing the number of components allocated to subsystem i.
r:	**r**=(r_1,r_2,...,r_m) is a row vector of the reliability for each component in subsystem i; $1 \le i \le m$
R'	total reliability of system without considering penalty function
$\tilde{R}_i(t)$	estimated lower bound of reliability for subsystem i at time t; $1 \le i \le m$
$R_{c,i}(t_m)$:	cold-standby reliability at mission time (t_m) calculated for subsystem i; $1 \le i \le m$
R_s:	total reliability of system
M:	total number of considered resource constraints
$g_j(\mathbf{r},\mathbf{n})$:	The j^{th} constraint of the mathematical formulation. $1 \le j \le M$
$f(\mathbf{r},\mathbf{n})$:	the objective function of the mathematical model
l:	**l**=(l_1,l_2,…,l_M) is a vector that implies the resource bounds
$f_i^x(t)$:	the probability density function of the x^{th} component failure time in subsystem i at time t; $1 \le i \le m$
R_i and $R_i(n_i,t)$:	reliability of subsystem i at time t involving n components; $1 \le i \le m$
V, C, and W:	The upper bound of the sum of the subsystems' product of volume and weight, the upper bound on the total cost of the entire system, and the upper bound on the total weight of the system, respectively.
w_i, v_i, and c_i:	Weight, volume, and cost of each component in subsystem i, respectively.
α_i , β_i :	physical features of subsystem i; $1 \le i \le m$
Z^+:	discrete space of positive integers
Raindrop:	solutions that are generated by the water cycle algorithm
Npop	the quantity of raindrops to be generated in the initializing step of the algorithm
NS	intensity of flow
d	distance between individuals
$X_{individual}$	flow direction of a particular individual
ω_i :	penalty parameter greater than one
$\tau(r_i, n_i)$:	penalty term
UB and *LB*:	Upper and lower bound of the variable, respectively
BP:	the position of the best point among the considered group of involved points
Nvar	Number of variables in a single solution
max_iteration:	maximum number of iterations

1. Formulation of the reliability-redundancy allocation problem

Improving the system reliability is the main objective of reliability optimization problems. The RRAP is useful for systems that have high reliability requirements (dos Santos Coelho, 2009). The RRAP is considered with the aim of maximizing system reliability subject to some non-linear constraints such as weight, cost and volume as follows:

$$Maximize \quad R_s = f(\mathbf{r},\mathbf{n}) \tag{1}$$

$$\text{subject to} \quad g_j(\mathbf{r},\mathbf{n}) \le l_j \tag{2}$$

$$0 \le r_i \le 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \le i \le m$$

where R_S indicates the reliability of the system; g_j represents the jth constraint; $\mathbf{r} = (r_1, r_2, \ldots, r_m)$ is the vector that indicates the component reliabilities of the system. $\mathbf{n} = (n_1, n_2, \ldots, n_m)$ is a vector that shows the redundancy level of subsystems; r_i and n_i indicate the reliability of and the number of components in the ith subsystem, respectively. $f(\mathbf{r},\mathbf{n})$ is the objective function of the problem i.e. the system reliability; l_j indicates the maximum level of resource. j and m is the total number of subsystems. The aim of the model is to maximize the total reliability of the system by determining the redundancy level and the components' reliability in each subsystem while considering the resource limitations. The RRAP model is categorized as the constrained MINLP optimization problem (Ardakan & Hamadani, 2014a).

2.1. System reliability with the cold standby redundancy strategy

Coit (2001) presents a general formulation of a subsystem reliability with the cold standby strategy that is appropriate for any distribution of component time-to-failure, as in Eq. (3).

$$R_i(t) = r_i(t) + \sum_{x=1}^{n_i-1} \int_0^t r_i(t-u)\, f_i^{(x)}(u)\, du \tag{3}$$

where $r_i(t)$ indicates the component reliability at time t in subsystem i; n_i shows the number of components in the ith subsystem and $f_i^{(x)}(t)$ is the probability density function for the x_{th} failure arrival for subsystem i.

For determining the reliability of the subsystem with imperfect switching, Coit (2001) represented Eqs. (4) and (5) for two scenarios as follows:

Scenario 1 (S1): Continuous scanning and switching:

$$R_i(t) = r_i(t) + \sum_{x=1}^{n_i-1} \int_0^t \rho_i(u)\, r_i(t-u)\, f_i^{(x)}(u)\, du \tag{4}$$

Scenario 2 (S2): Switch activation only in response to failures:

$$R_i(t) = r_i(t) + \sum_{x=1}^{n_i-1} \rho_i^{\,x} \int_0^t r_i(t-u)\, f_i^{(x)}(u)\, du \tag{5}$$

where $\rho_i(t)$ and ρ_i are switch reliabilities at time t for the first and second scenarios, respectively. This paper considers the first switching scenario (*S1*). Coit (2001), also determined an appropriate lower bound on subsystem reliability, $\tilde{R}_i(t)$, as follows:

$$\tilde{R}_i(t) = r_i(t) + \rho_i(t) \sum_{x=1}^{n_i-1} \int_0^t r_i(t-u)\, f_i^{(x)}(u)\, du \tag{6}$$

As it is clear, for all u<=t, $\rho_i(t) \le \rho_i(u)$, therefore, Eq. (6) is an estimation for Eq. (4). Coit (2001), considered the time-to-failure of components as the Erlang distribution. In the RRAP, all non-linear constraints were extended by assuming the exponential time-to-failure (Dhingra, 1992). Hence, in this study, Eq. (6) is developed based on the exponential distribution and put into practice. In this case, Eq. (6) can be derived by applying the Poisson process. Thus, the occurrences of subsystem failures are strictly less than n_i failures (Coit, 2001), and Eq. (7), reformulates the Eq. (6) as follows:

$$\tilde{R}_i(t) = r_i(t) + \rho_i(t) \sum_{x=1}^{n_i-1} \frac{e^{-\lambda_i t} (\lambda_i t)^x}{x!} \tag{7}$$

where λ_i is the component failure rate in the exponential distribution and t represents the total mission time of the system.

In the present study, the calculation of subsystem reliability is based on Eq. (7). Therefore, instead of using r_i as a decision variable, the failure rate (λ_i) is considered the decision variable. As a result, there are two decision variables in this study as λ_i and n_i. We can obtain r_i based on λ_i. In the present work, two famous benchmark problems are considered. In the next subsections, these benchmarks are introduced.

2.2. Test problem 1 (*P1*): Series system

The first test problem includes five subsystems as series components which form a mixed-integer non-linear programming problem as follows:

$$Maximize\ f(\mathbf{r},\mathbf{n}) = \prod_{i=1}^{m} R_i(n_i,t) \tag{8}$$

Subject to

$$g_1(\mathbf{r},\mathbf{n}) = \sum_{i=1}^{m} w_i.v_i^2.n_i^2 \le V \tag{9}$$

$$g_2(\mathbf{r},\mathbf{n}) = \sum_{i=1}^{m} \alpha_i \cdot \left(-\frac{1000}{\ln r_i}\right)^{\beta_i} \cdot \left[n_i + e^{0.25 n_i}\right] \le C \tag{10}$$

$$g_3(\mathbf{r},\mathbf{n}) = \sum_{i=1}^{m} w_i\ .\ n_i\ .\ e^{0.25 n_i} \le W \tag{11}$$

$$0 \le r_i \le 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \le i \le m \tag{12}$$

This problem was also considered by (Afonso et al., 2013; Ardakan & Hamadani, 2014a; Y-C Hsieh & You, 2011; Mellal & Zio, 2016; Valian et al., 2013; Valian & Valian, 2013; Yeh & Hsieh, 2011). A schematic view of the Series system is depicted in Fig. 1.

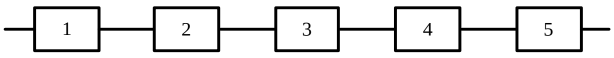


Figure 1. A schematic view of the Series system (*P1*)

Here, the objective function, Eq. (8), contains the redundancy level of subsystem *i* and the reliability of the component applied to gain the maximum system reliability at time *t*. As mentioned before, the reliability is calculated using Eq. (7). Volume, cost and weight limitations are considered in Eqs. (9- 11). A combination of weight, redundancy allocation and volume is given by Eq. (9). Eq. (10) and Eq. (11) are cost and weight limitations, respectively (Afonso et al., 2013). In this formulation, V is the upper limit of the sum of the subsystem's product of volume and weight, W represents the upper limit on the weight of the system and C denotes the upper limit on the cost of the whole system. Also, weight of each component in subsystem *i* is denoted by w_i, volume of each component in subsystem *i* is symbolized by v_i and finally, c_i is the cost of each component in subsystem *i*. Physical features of the *i*th subsystem are implied by β_i and α_i. Discrete space of positive integers is represented by Z^+.

2.3. Test problem 2 (P2): Series-Parallel system

Same as the first test problem, the second one includes five subsystems and is a mixed-integer nonlinear programming problem as follows:

$$Maximize\ f(\mathbf{r},\mathbf{n}) = 1-(1-R_1R_2)(1-(R_3+R_4-R_3R_4)R_5) \tag{13}$$

Subject to

$$g_1(\mathbf{r},\mathbf{n}) \le V,\quad g_2(\mathbf{r},\mathbf{n}) \le C,\quad g_3(\mathbf{r},\mathbf{n}) \le W$$
$$0 \le r_i \le 1, \quad r_i \in \Re, \quad n_i \in Z^+, \quad 1 \le i \le m$$

where, the calculation of R_i is based on Eq. (7), and all variable conditions are the same as those mentioned in *P1*. We can find the same test problem in (Afonso et al., 2013; Ardakan & Hamadani, 2014a, 2014b; Mellal & Zio, 2016; Valian et al., 2013; Valian & Valian, 2013; Wang & Li, 2012). Fig. 2 depicts this Series-Parallel system schematically.

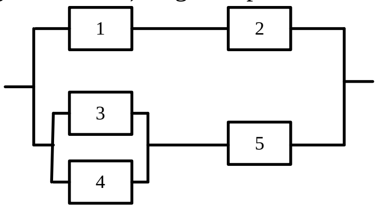


Figure 2. A schematic view of the Series-Parallel system (*P2*)

3. Solution method: Water Cycle Algorithm

Water Cycle Algorithm, as one of the most powerful evolutionary algorithms, was first introduced by Eskandar et al., 2012. The idea of Water Cycle Algorithm (WCA) was inspired by water circulation in nature. Fig. 3 shows a simple diagram of the hydrologic cycle based on the observations in nature. By melting the glaciers in the mountains, streams are formed and move downhill. By joining some streams through their journey downhill, the rivers are made. Finally, the rivers end up in a sea. During this mechanism, water in rivers and lakes is evaporated and comes back to the earth as raindrops.

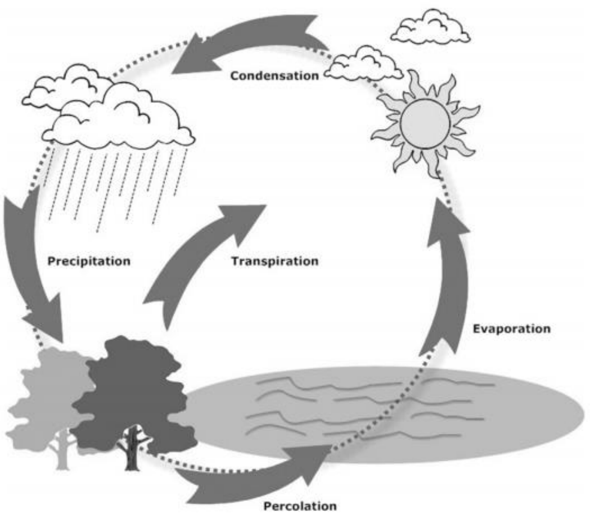


Figure 3. A schematic diagram of water cycle in nature (Eskandar et al., 2012)

Same as other metaheuristic algorithms, creating the initial population is the first step of this algorithm. Here, the initial population is called raindrops. The best raindrop is considered sea. Then, several good individuals (raindrops) are chosen as rivers and the rest of individuals are implied as streams. Each river absorbs streams based on their magnitude and flows to the sea (Eskandar et al., 2012).

3.1. Creating the first population

In the GA and PSO, the values of a single population were formed as an array called "Chromosome" and "Particle Position", respectively. Accordingly, in this algorithm, this array is called "Raindrop" for each individual. The test problems applied in this paper contain 10 variables. Hence, the particular raindrop for these problems is a 1×10 vector as follows:

$$Raindrop = [\lambda_1 \quad \lambda_2 \quad \lambda_3 \quad \lambda_4 \quad \lambda_5 \quad n_1 \quad n_2 \quad n_3 \quad n_4 \quad n_5] \tag{14}$$

Where λ_i is failure rate of the component and n_i indicates the redundancy in each subsystem. To start the optimization process, a population with N_{pop} size is needed. We can obtain the population of raindrops by representing a matrix of size N_{pop} × 10 as Fig. 4.

$$\text{Population of raindrops} = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} & \lambda_{1,4} & \lambda_{1,5} & n_{1,1} & n_{1,2} & n_{1,3} & n_{1,4} & n_{1,5} \\ \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} & \lambda_{2,4} & \lambda_{2,5} & n_{2,1} & n_{2,2} & n_{2,3} & n_{2,4} & n_{2,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \lambda_{Npop,1} & \lambda_{Npop,2} & \lambda_{Npop,3} & \lambda_{Npop,4} & \lambda_{Npop,5} & n_{Npop,1} & n_{Npop,2} & n_{Npop,3} & n_{Npop,4} & n_{Npop,5} \end{bmatrix}$$

Figure 4. The population representation

The objective function calculates the reliability of each raindrop as follows:

$$R'_i = f(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}, \lambda_{i4}, \lambda_{i5}, n_{i1}, n_{i2}, n_{i3}, n_{i4}, n_{i5}) \quad i = 1, 2, 3, ..., Npop \tag{15}$$

In the searching process of the algorithm, streams and rivers may violate the considered constraints of the problem. In this paper, a penalty function is added to the objective function in order to remove the infeasible solutions. By applying this penalty, the feasibility of the final solution is guaranteed. The formulation of the modified fitness function is as follows:

$$\text{fitness function} = -f(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}, \lambda_{i4}, \lambda_{i5}, n_{i1}, n_{i2}, n_{i3}, n_{i4}, n_{i5}) + \tau(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}, \lambda_{i4}, \lambda_{i5}, n_{i1}, n_{i2}, n_{i3}, n_{i4}, n_{i5}) \quad i \in Npop \tag{16}$$

The water cycle algorithm is originally designed to consider the minimum value of cost function as the best solution. For this reason, a sign (-) is added before the objective function in order to find the maximum optimum. In this case, if one of the streams or rivers uses more resources than the upper bound of the resource limitations, the term $\tau(\lambda_{i1}, \lambda_{i2}, \lambda_{i3}, \lambda_{i4}, \lambda_{i5}, n_{i1}, n_{i2}, n_{i3}, n_{i4}, n_{i5})$ adds a proper penalty to the fitness function. The penalty function is calculated as follows:

$$\tau(r_1, r_2, ..., r_m, n_1, n_2, ..., n_m) = \sum_{i=1}^{3} \omega_i . \max(0, \frac{g_j(\mathbf{r},\mathbf{n})}{l_j} - 1) \tag{17}$$

In the present study, the violation of a constraint such as $g_j(\mathbf{r},\mathbf{n}) \le l_j$ is formulated as $\frac{g_j(\mathbf{r},\mathbf{n})}{l_j} - 1$ and ω_i is a number greater than 1, which is used to remove the infeasible streams or rivers. The feasibility of the final solution (i.e. sea) is ensured by this approach.

For the first population, the maximum values of raindrops are considered as rivers and a sea. *Nsr* shows the number of individuals which are chosen as sea and rivers. It is formulated in Eq. (18). The number of remained raindrops is calculated by using Eq. (19).

$$N_{sr} = \textit{Number of Rivers} + 1 \tag{18}$$
$$N_{Raindrops} = N_{pop} - N_{sr} \tag{19}$$

By using Eq. (20), the number of streams which are assigned to the specific rivers and sea is calculated.

$$NS_n = round\left\{ \left| \frac{\mathrm{R}'(n)}{\sum_{i=1}^{Nsr} \mathrm{R}'(i)} \right| \times N_{Raindrops} \right\}, \quad n = 1, 2, ..., N_{sr} \tag{20}$$

where NS_n is the number of streams which flow to the rivers or sea based on the intensity of the flow (Eskandar et al., 2012).

3.2. Stream flow

By joining the streams to each other, new rivers are created, and some streams can flow directly to the sea, which is the best individual. A schematic view of stream's flow is illustrated in Fig. 5.

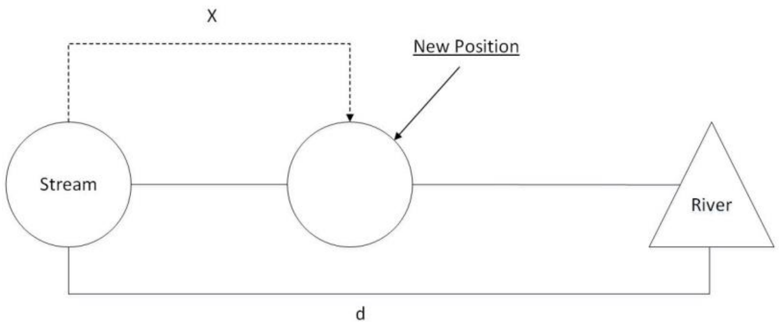


Figure 5. Simplified diagram of the stream's flow to a specific river

The stream needs a route to find its way towards a specified river. Therefore, a flow direction helps the stream to reach a new position as follows:

$$X \in (0, C \times d), \quad 1 < C < 2 \tag{21}$$

Where C is a constant value between 1 and 2, and d is the current distance between the stream and river. The value of X is the flow direction.

This concept is also used in the process of flowing rivers to sea. Hence, after obtaining a new route, the new positions of stream and river are derived as following equations:

$$X^{i+1}_{stream} = X^{i}_{Stream} + rand \times C \times (X^{i}_{River} - X^{i}_{Stream}) \tag{22}$$

$$X^{i+1}_{River} = X^{i}_{River} + rand \times C \times (X^{i}_{Sea} - X^{i}_{River}) \quad (23)$$

Where a uniformly distributed random number in interval [0,1] is represented by *rand*. The positions of the stream and the river are exchanged if the solution of the stream is better than the river. Such exchange may happen for a river and a sea (Eskandar et al., 2012).

3.3. Evaporation condition

In order to prevent the algorithm from rapid convergence, evaporation is considered in the algorithm procedure. As depicted in Fig. 3, water which is evaporated from rivers and lakes finally converts into clouds and raindrops. The rains create new streams which flow to rivers or directly to the sea. By inspiring this concept, the proposed method avoids being trapped in local optimum solutions. If the distance between a specified river and a sea is less than a constant number (i.e. d_{max}), then the river joins the sea. In this situation, evaporation and raining processes occur. The following pseudo-code shows the condition of evaporation and raining processes (Eskandar et al., 2012).

Algorithm 1. Pseudo-code of evaporation and raining processes

```
1: If $|X^i_{Sea} - X^i_{River}| < d_{max}$
2:    Start evaporation and raining processes
3: End if
```

Figure 6. Pseudo-code of evaporation and raining condition

The value of d_{max} is decreased iteratively, as Eq. (24):

$$d^{i+1}_{max} = d^{i}_{max} - \frac{d^{i}_{max}}{max\,iteration} \quad (24)$$

3.4. Raining process

After applying the evaporation process, the raining process begins. In this state, new raindrops form streams in various locations. The following equation specifies the locations of new streams (Eskandar et al., 2012):

$$X^{new}_{Stream} = LB + rand \times (UB - LB) \quad (25)$$

Where *LB* and *UB* are lower and upper bounds of the defined problem, respectively. Again, the best raindrop joins the sea while the rest of the raindrops form streams and rivers.

In order to improve the convergence rate of the algorithm, Eq. (26) is applied to those streams which straightly flow to the sea. Furthermore, to enhance the exploration near the sea region (the best solution), this equation is formulated and used as follows:

$$X^{new}_{Stream} = X_{Sea} + \sqrt{\mu} \times randn(1, N_{var}) \quad (26)$$

Where parameter μ controls the range of searching area near the sea and *randn* stands for a normally distributed random number. In this problem, the number of variables (N_{var}) is 10. As well as the RRAP is a constrained problem, Eq. (26) improves the computational performance of the algorithm to search more in the feasible region.

3.5. Steps of the adjusted WCA

The pseudo-code of the penalty guided water cycle algorithm, applied to solving the RRAP is presented in algorithm 2, followed by the flowchart as in Fig. 8 (Eskandar et al., 2012).

Algorithm 2. Pseudo-code of the adjusted water cycle algorithm

```
1: Set the initial parameters of WCA: Npop, Nsr, max_iteration and dmax
2: generate initial population by Eqs. (14), (18) and (19)
3: When the number of iterations is less than max_iteration,:
4:     evaluate the fitness function of raindrops by Eq. (16)
5:     calculate intensity of flow by Eq. (20)
6:     flow the streams to the rivers by Eq. (22)
7:     flow the rivers to the sea by Eq. (23)
8:     replace the position of a stream with a river which gives a better solution.
```

```
9:     replace the position of a river with a sea which gives a better solution.
10:     check the evaporation condition based on algorithm 1.
11:    if the evaporation condition is met, then create clouds and perform the raining process by Eqs. (25) and (26)
12:     update the value of $d_{max}$ by Eq. (24)
13: End while
14: Print the final obtained results
```

Figure 7. Pseudo-code of the adjusted water cycle algorithm

4. Experimental results

In this section, the experimental results obtained from implementing the proposed WCA are presented for the two benchmark problems described in Section 2. The proposed water cycle algorithm was coded in MATLAB software on an Intel Core i7 with 4 GB of RAM on a personal computer. Taguchi method, one of the most applicable designs of experiments, is employed to tune the initial WCA factors including *Npop*, *Nsr*, *max_iteration* and *dmax*. In this paper, each factor is considered a 3-level parameter. Table 1 shows the input values of the Taguchi method, utilized for the proposed WCA.

Table 1. Input values of the Taguchi method

Factor Level		1	2	3
1	*Npop*	500	1000	1500
2	*Nsr*	4	5	6
3	*max_iteration*	30	40	50
4	*dmax*	1e-15	1e-10	1e-5

Main effects for means of each factor indicate the influence of the parameter on the total reliability at different levels, as plotted in Fig. 9. The highest value represents the best level for each factor.

As a result, the following parameters are set in the proposed WCA: Npop=1500, Nsr=5, max_iteration= 30 and dmax=1e-10. Same as other studies (Ardakan & Hamadani, 2014; Coit, 2001; Tavakkoli-Moghaddam et al., 2008), the switch is assumed to be imperfect and its reliability is considered to be 0.99. In order to obtain desired results of the proposed algorithm, each benchmark problem was solved independently five times. Table 2 and 3 present the input parameters of the benchmark P1 (Series system) and P2 (Series-Parallel system), respectively.

Table 2. Input parameters of benchmark *P1* (Series system)

Stage	$10^5.\alpha_i$	β_i	$w_i.v_i^2$	w_i	*V*	*C*	*W*
1	2.330	1.5	1	7	110	175	200
2	1.450	1.5	2	8			
3	0.541	1.5	3	8			
4	8.050	1.5	4	6			
5	1.950	1.5	2	9			

Table 3. Input parameters of benchmark P2 (Series-Parallel system)

Stage	$10^5.\alpha_i$	β_i	$w_i.v_i^2$	w_i	*V*	*C*	*W*
1	2.500	1.5	2	3.5	180	175	100
2	1.450	1.5	4	4.0			
3	0.541	1.5	5	4.0			
4	0.541	1.5	8	3.5			
5	2.100	1.5	4	4.5			

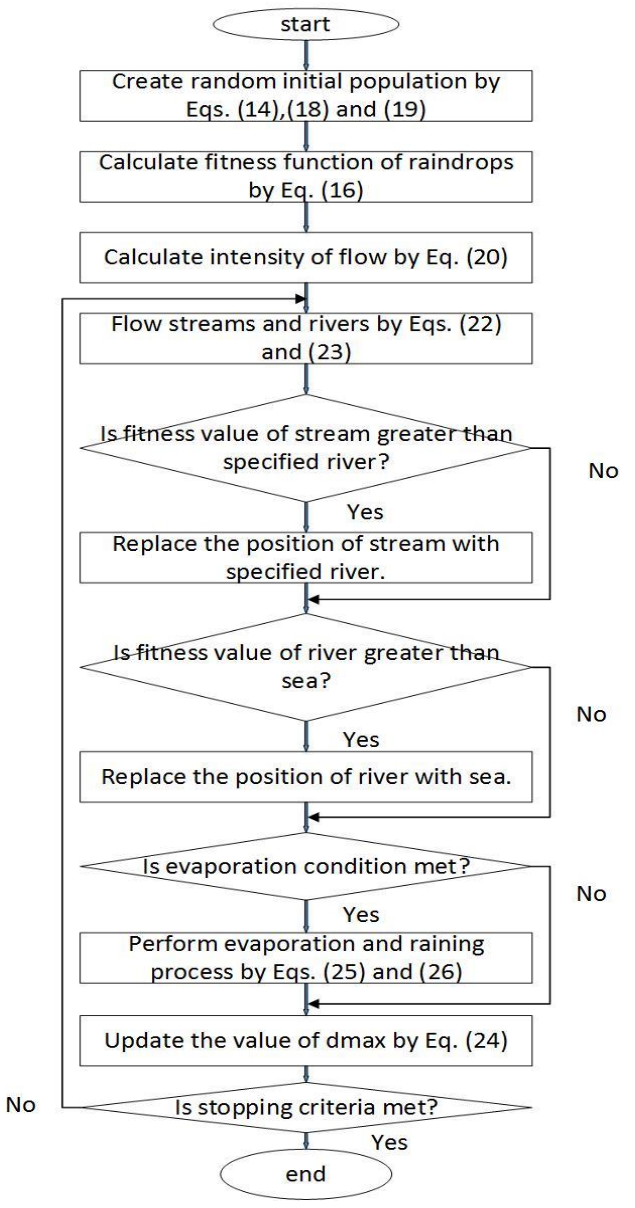

start
Create random initial population by Eqs. (14),(18) and (19)
Calculate fitness function of raindrops by Eq. (16)
Calculate intensity of flow by Eq. (20)
Flow streams and rivers by Eqs. (22) and (23)
Is fitness value of stream greater than specified river?
No
Yes
Replace the position of stream with specified river.
Is fitness value of river greater than sea?
No
Yes
Replace the position of river with sea.
Is evaporation condition met?
No
Yes
Perform evaporation and raining process by Eqs. (25) and (26)
Update the value of dmax by Eq. (24)
No
Is stopping criteria met?
Yes
end

Figure 8. Flowchart diagram of the adjusted water cycle algorithm

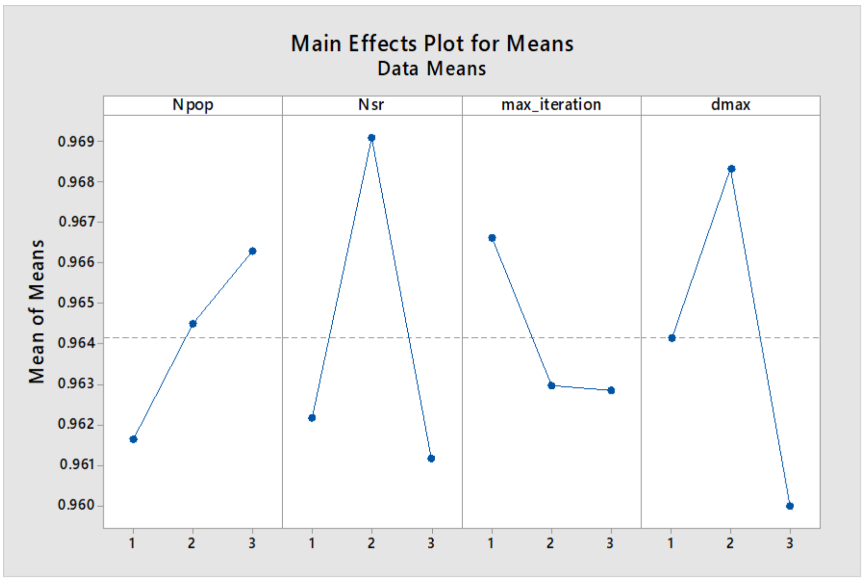


Figure 9. Main effects plot for means

The simulation results of all five runs of the proposed WCA for the test problems *P1* and *P2* are represented in Table 4 and 5, respectively. Also Tables 4 and 5 report the STD of the fitness function in different five runs.

Table 4. Different runs of WCA for test problem *P1* (Series system)

Parameter	Run #1	Run #2	Run #3	Run #4	Run #5	STD
R_s	0.96957819	0.96957873	0.96817799	0.96957812	**0.96957858**	0.00056
λ_i	0.00026535	0.00026720	0.00026993	0.00026757	0.00026718	
	0.00011931	0.00011941	0.00023774	0.00011953	0.00011930	
	0.00008923	0.00008857	0.00008920	0.00008836	0.00008884	
	0.00036702	0.00036627	0.00036952	0.00036674	0.00036644	
	0.00025285	0.00025355	0.00013288	0.00025259	0.00025298	
n	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 3, 2, 3, 2)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	
r	0.76693891	0.76552336	0.76343354	0.76523618	0.76553481	
	0.88752974	0.88743978	0.78841410	0.88733991	0.88754511	
	0.91463445	0.91523597	0.91466407	0.91543219	0.91499220	
	0.69279863	0.69331569	0.69107159	0.69299238	0.69319857	
	0.77658645	0.77604143	0.87557043	0.77678395	0.77648630	
Slack (g1)	27	27	27	27	27	
Slack (g2)	0.00005083	0.00000051	0.00000002	0.00000003	0.00021772	
Slack (g3)	7.51891824	7.51891824	10.5724757	7.51891824	7.51891824	

Table 5. Different runs of WCA for test problem *P2* (Series-Parallel system)

Parameter	**Run #1**	**Run #2**	**Run #3**	**Run #4**	**Run #5**	**STD**
R_s	**0.99998828**	0.99998828	0.99998827	0.99998828	0.99998828	4.0e-9
λ_i	0.00019138	0.00019160	0.00019276	0.00019114	0.00019202	
	0.00016465	0.00016306	0.00016602	0.00016391	0.00016510	
	0.00010657	0.00009639	0.00009682	0.00009606	0.00009603	
	0.00009648	0.00010742	0.00010513	0.00010734	0.00010598	
	0.00014830	0.00014885	0.00014694	0.00014887	0.00014806	
n	(3, 3, 1, 2, 3)	(3, 3, 2, 1, 3)	(3, 3, 2, 1, 3)	(3, 3, 2, 1, 3)	(3, 3, 2, 1, 3)	
r	0.82582078	0.82564109	0.82467802	0.82601610	0.82528969	
	0.84818976	0.84954379	0.84702574	0.84881753	0.84781158	
	0.89891322	0.90811131	0.90772127	0.90840989	0.90843699	
	0.90802848	0.89814457	0.90020768	0.89821940	0.89943930	
	0.86217186	0.86169644	0.86334976	0.86167768	0.86237954	
Slack (g1)	53	62	62	62	62	
Slack (g2)	0.00013618	0.00057513	0.00119624	0.00000193	0.00005565	
Slack (g3)	7.11084884	6.10414028	6.10414028	6.10414028	6.10414028	

Where λ_i shows the failure rate of components used in subsystem *i* when the time-to-failure of components are exponentially distributed. n_i represents the total number of components in subsystem *i*, and r_i is the reliability of each component in subsystem *i* at mission time (1000 hour) which is calculated based on λ_i. Finally, the slack parameters imply unused resources for each constraint. The best obtained structure for each benchmark is illustrated in Fig. 10 and 11.

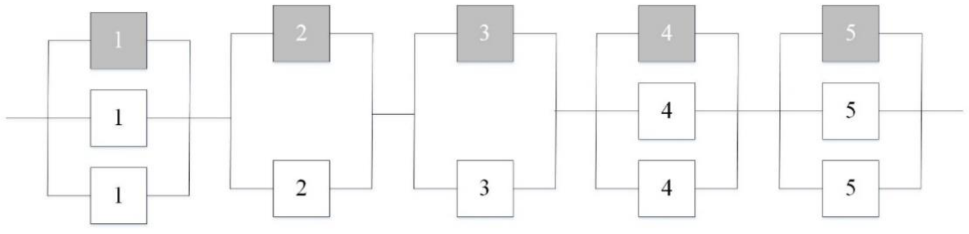


Figure 10. The best structure of Series system (*P1*)

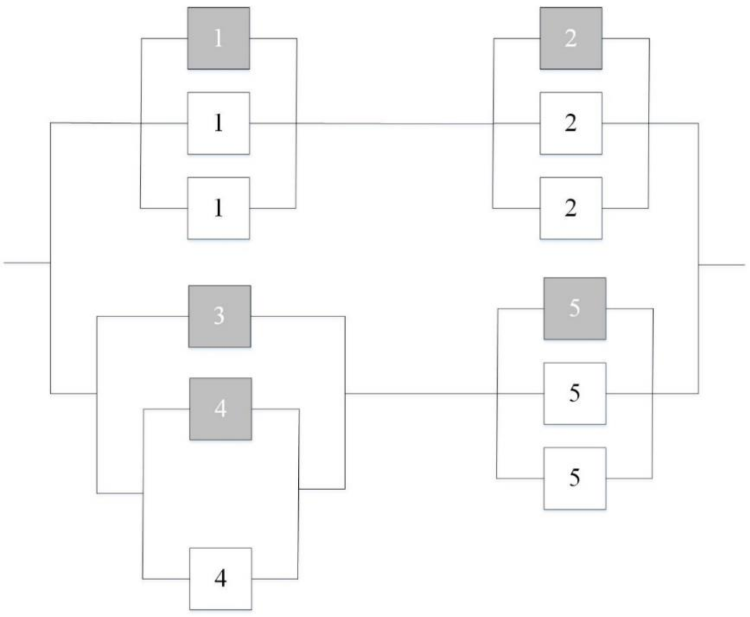


Figure 11. The best structure of Series-Parallel system (*P2*)

The improving trends of WCA on the RRAP for benchmarks P1 and P2 are illustrated in Fig. 12 and 13, respectively. Accordingly, the tuned algorithm tries to meet the optimum value in the first 30 iterations, which indicates the fast convergence rate of the adjusted WCA.

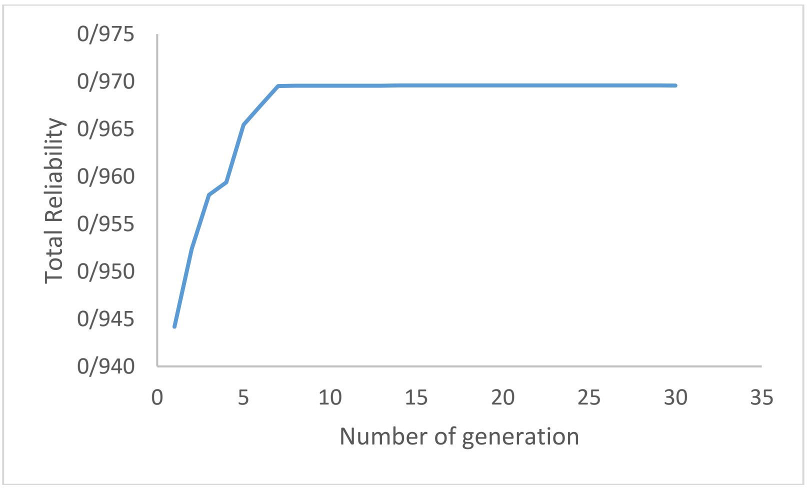


Figure 12. Plot of the adjusted water cycle algorithm in solving test problem *1* (Series system)

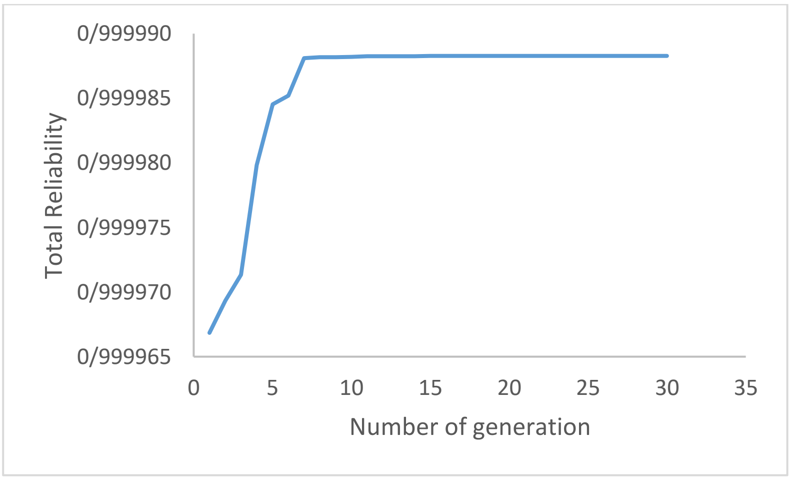


Figure 13. Plot of the adjusted water cycle algorithm in solving test problem 2 (Series-Parallel system)

The best obtained result among five different runs is selected to be compared with those reported in the literature. The results of comparing the obtained solutions with those reported in previous studies are reported in Tables 6 and 7. The Maximum Possible Improvement (*MPI*) index is used in order to measure the impact of the improvements achieved by using the adjusted WCA. The *MPI* is calculated as follows (Ardakan & Rezvan, 2017; dos Santos Coelho, 2009; Mellal & Zio, 2016; Yeh & Hsieh, 2011):

$$MPI(\%) = [R_s(New\,Approach) - R_s(Other)]/[1 - R_s(Other)] \tag{27}$$

Where R_s (*New Approach*) is the total reliability obtained from the proposed solution method in this study, and R_s *(Other)* shows the total reliability reported in other similar RRAP studies.

Based on the MPI values reported in Tables 6 and 7, it is clear that the proposed WCA outperforms all those in the previous studies. The advantage of the standby strategy over the active one is also revealed. More importantly, it can be seen that compared to a recent study by Ardakan & Hamadani (2014a), the proposed water cycle algorithm obtained better solutions in the standby structure. More specifically, in comparison with the solutions reported in Ardakan & Hamadani (2014a) for Series and Series-Parallel benchmarks, the MPI index values of the study show improvements by 0.0032% and 0.2638%, respectively. These remarkable improvements reveal the superior performance and robustness of the adjusted water cycle algorithm in dealing with the RRAP problems.

Table 6. Comparison of best results obtained by the adjusted water cycle algorithm with those mentioned in the literature for Series system (P1)

	Active strategy								**Cold-standby strategy**	
Parameter	(Yi-Chih Hsieh, Chen, & Bricker, 1998)	(Chen, 2006)	(Y-C Hsieh & You, 2011)	(Wu et al., 2011)	(Yeh & Hsieh, 2011)	(Valian & Valian, 2013)	(Afonso et al., 2013)	(Mellal & Zio, 2016)	(Ardakan & Hamadani, 2014a)	This study
Rs	0.93157800	0.93167800	0.93168234	0.93168000	0.93168200	0.93168239	0.93167939	0.93168239	0.96957758	**0.96957858**
n	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)	(3, 2, 2, 3, 3)
r	0.77942700	0.77926600	0.77946230	0.78037307	0.77939900	0.77941694	0.77987400	0.77939888	0.76459335	0.76553481
	0.86948200	0.87251300	0.87188346	0.87178343	0.87183700	0.87183328	0.87205700	0.87183701	0.88752892	0.88754511
	0.90267400	0.90263400	0.90280088	0.90240890	0.90288500	0.90288508	0.90342600	0.90288536	0.91539527	0.91499220
	0.71403800	0.71064800	0.71135017	0.71147356	0.71140300	0.71139387	0.71096000	0.71140252	0.69350544	0.69319857
	0.78689600	0.78840600	0.78786159	0.78738760	0.78780000	0.78780371	0.78690200	0.78779948	0.77603145	0.77648630
MPI(%)	55.5385	55.4734	55.4706	55.4722	55.4709	55.4706	55.4725	55.4706	0.0032	---
Slack (g1)	27	27	27	27	27	27	27	27	27	27
Slack (g2)	0.12145400	0.00155900	0.00000053	0.00010100	-0.0002184	0.00000027	0.00009900	0.00000000	0.00002478	0.00021772
Slack (g3)	7.51891800	7.51891800	7.51891800	7.51891800	7.51891820	7.51891824	7.51891800	7.51891824	7.51891824	7.51891824

5. Concluding remarks

In this paper, one of the major problems in the field of reliability optimization problem called reliability-redundancy allocation problem (RRAP) was considered. The RRAP aims to maximize the system reliability by selecting appropriate components reliability and suitable redundancy level, subject to some resource limitations. Unlike most recent studies which used the active redundancy strategy, in this research the cold standby strategy was chosen for redundant components. For solving the cold standby RRAPs, the present study used an adjusted water cycle algorithm. In order to check the validity of the proposed solution method, its performance was compared with those algorithms reported in other similar studies. For future work, implementing heterogeneous components in the benchmark problems can be an interesting idea. Furthermore, the proposed algorithm might be applied to handle other reliability optimization problems.

Table 7. Comparison of best results obtained by the adjusted water cycle algorithm with those mentioned in the literature for Series-Parallel system (*P2*)

	Active strategy								**Cold-standby strategy**	
Parameter	(Chen, 2006)	(Yeh & Hsieh, 2011)	(Wu et al., 2011)	(Y-C Hsieh & You, 2011)	(Valian & Valian, 2013)	(Wang & Li, 2012)	(Afonso et al., 2013)	(Mellal & Zio, 2016)	(Ardakan & Hamadani, 2014a)	This study
Rs	0.9999765800	0.9999773100	0.9999766400	0.9999766490	0.9999766490	0.9999766500	0.9999766100	0.9999766491	0.9999882490	**0.99998828**
n	(2,2,2,2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2, 2, 2, 2, 4)	(2,2,2,2,4)	(3,3,2,1,3)	(3,3,1, 2,3)
r	0.81248500	0.81974570	0.81918526	0.81959156	0.81992709	0.81959600	0.82201264	0.81965939	0.82484673	0.82582078
	0.84315500	0.84500800	0.84366421	0.84495107	0.84526766	0.84500000	0.84365640	0.84498085	0.84281657	0.84818976
	0.89738500	0.89545810	0.89472992	0.89542855	0.89549155	0.89551400	0.89129092	0.89550643	0.90817308	0.89891322
	0.89451600	0.90090320	0.89537628	0.89552234	0.89544069	0.89551900	0.89869886	0.89550645	0.89869900	0.90802848
	0.87059000	0.86840690	0.86912724	0.86849023	0.86831878	0.86845600	0.86824939	0.86844769	0.86546301	0.86217186
MPI (%)	49.9573	48.34729	49.8288	49.8094	49.8094	49.8072	49.8931	49.8092	0.2638	---
Slack (g1)	40	40	40	40	40	40	40	40	62	53
Slack (g2)	0.00262700	-1.46952200	0.00056100	0.00000000	0.00001610	0.00000700	0.00039600	0.00000000	0.00006415	0.00013618
Slack (g3)	1.60928900	1.60928900	1.60928900	1.60928900	1.60928900	1.60928900	1.60928900	1.60928897	6.10414028	7.11084884

References

Abouei Ardakan, M., and Rezvan, M. T. (2018). Multi-objective optimization of reliability–redundancy allocation problem with cold-standby strategy using NSGA-II. *Reliability Engineering & System Safety*, Vol. 172, pp. 225–238.

Abouei Ardakan, M., Sima, M., Zeinal Hamadani, A., and Coit, D. W. (2016). A novel strategy for redundant components in reliability--redundancy allocation problems. *IIE Transactions*, Vol. 48(11), pp. 1043–1057.

Afonso, L. D., Mariani, V. C., and dos Santos Coelho, L. (2013). Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability-redundancy optimization. *Expert Systems with Applications*, Vol. 40(9), pp. 3794–3802.

Ardakan, M. A., and Hamadani, A. Z. (2014a). Reliability–redundancy allocation problem with cold-standby redundancy strategy. *Simulation Modelling Practice and Theory*, Vol. 42, pp. 107–118.

Ardakan, M. A., and Hamadani, A. Z. (2014b). Reliability optimization of series–parallel systems with mixed redundancy strategy in subsystems. *Reliability Engineering & System Safety*, Vol. 130, pp. 132–139.

Ardakan, M. A., Hamadani, A. Z., and Alinaghian, M. (2015). Optimizing bi-objective redundancy allocation problem with a mixed redundancy strategy. *ISA Transactions*, Vol. 55, pp. 116–128.

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

COIT, D. W. (2001). Cold-standby redundancy optimization for nonrepairable systems. *IIE Transactions*, Vol. 33(6), pp. 471–478.

Dhingra, A. K. (1992). Optimal apportionment of reliability and redundancy in series systems under multiple objectives. *IEEE Transactions on Reliability*, Vol. 41(4), pp. 576–582.

dos Santos Coelho, L. (2009). An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications. *Reliability Engineering & System Safety*, Vol. 94(4), pp. 830–837.

Elsayed, E. A. (2012). *Reliability engineering* (Vol. 88). John Wiley & Sons.

Eskandar, H., Sadollah, A., Bahreininejad, A., and Hamdi, M. (2012). Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, Vol. 110–111, 151–166.

Habib, A., Alsieidi, R., and Youssef, G. (2009). Reliability analysis of a consecutive r-out-of-n: F system based on neural networks. *Chaos, Solitons & Fractals*, Vol. 39(2), pp. 610–624.

Hikita, M., Nakagawa, Y., Nakashima, K., and Narihisa, H. (1992). Reliability optimization of systems by a surrogate-constraints algorithm. *IEEE Transactions on Reliability*, Vol. 41(3), pp. 473–480.

Hsieh, T.-J., and Yeh, W.-C. (2012). Penalty guided bees search for redundancy allocation problems with a mix of components in series–parallel systems. *Computers & Operations Research*, Vol. 39(11), pp. 2688–2704.

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

Hsieh, Y.-C., and You, P.-S. (2011). An effective immune based two-phase approach for the optimal reliability–redundancy allocation problem. *Applied Mathematics and Computation*, Vol. 218(4), pp. 1297–1307.

Kim, H. (2018). Maximization of system reliability with the consideration of component sequencing. *Reliability Engineering & System Safety*, Vol. 170(Supplement C), pp. 64–72.

Kuo, W. (2001). *Optimal reliability design: fundamentals and applications*. Cambridge university press.

Kuo, W., Lin, H.-H., Xu, Z., & Zhang, W. (1987). Reliability optimization with the Lagrange-multiplier and branch-and-bound technique. *IEEE Transactions on Reliability*, Vol. 36(5), pp. 624–630.

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

Mellal, M. A., and Zio, E. (2016). A penalty guided stochastic fractal search approach for system reliability optimization. *Reliability Engineering & System Safety*, Vol. *152*(Supplement C), pp. 213–227.

Nahas, N., and Nourelfath, M. (2005). Ant system for reliability optimization of a series system with multiple-choice and budget constraints. *Reliability Engineering & System Safety*, Vol. 87(1), pp. 1–12.

Peiravi, A., Karbasian, M., and Abouei Ardakan, M. (2017). K-mixed strategy: A new redundancy strategy for reliability problems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 1748006X17736166. Article in press.

Ramirez-Marquez, J. E., Coit, D. W., and Konak, A. (2004). Redundancy allocation for series-parallel systems using a max-min approach. *Iie Transactions*, Vol. 36(9), pp. 891–898.

Tavakkoli-Moghaddam, R., Safari, J., and Sassani, F. (2008). Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm. *Reliability Engineering & System Safety*, Vol. 93(4), pp. 550–556.

Valian, E., Tavakoli, S., Mohanna, S., and Haghi, A. (2013). Improved cuckoo search for reliability optimization problems. *Computers & Industrial Engineering*, Vol. 64(1), pp. 459–468.

Valian, E., and Valian, E. (2013). A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems. *Engineering Optimization*, Vol. 45(11), pp. 1273–1286.

Wang, L., and Li, L. (2012). A coevolutionary differential evolution with harmony search for reliability–redundancy optimization. *Expert Systems with Applications*, Vol. 39(5), pp. 5271–5278.

Wu, P., Gao, L., Zou, D., and Li, S. (2011). An improved particle swarm optimization algorithm for reliability problems. *ISA Transactions*, Vol. 50(1), pp. 71–81.

Yeh, W.-C., and Hsieh, T.-J. (2011). Solving reliability redundancy allocation problems using an artificial bee colony algorithm. *Computers & Operations Research*, Vol. 38(11), pp. 1465–1473.

Zou, D., Gao, L., Li, S., and Wu, J. (2011). An effective global harmony search algorithm for reliability problems. *Expert Systems with Applications*, Vol. 38(4), pp. 4642–4648.