An adaptive cuckoo optimization algorithm for system design optimization under failure dependencies

Mohamed Arezki Mellal[a,*], Enrico Zio[b,c]

[a]*LMSS, Faculty of Engineering Sciences (FSI), M'Hamed Bougara University, Boumerdes, Algeria*

mellal.mohamed@univ-boumerdes.dz, mellal.mohamed@gmail.com

[b]*Mines ParisTech, PSL Research University, CRC, Sophia Antipolis, France*

[c]*Energy Department, Politecnico di Milano, Milano, Italy*

Abstract

This paper presents an algorithm for optimal redundancy and repair teams allocation with respect to minimum system cost and a system availability constraint. Four scenarios are considered for the failures occurring in the subsystems of the system: independence, linear dependence, weak dependence, and strong dependence. An adaptive cuckoo optimization algorithm (AA-COA) is developed for solving the nonlinear integer optimization problem of allocation. A series-parallel system with six subsystems is considered as a case study for demonstration purposes. The results obtained highlight the good performance of the developed algorithm.

Keywords: Redundancy allocation; Repair allocation; System availability; Failure dependencies; Adaptive cuckoo optimization algorithm; Series-parallel system.

Notations

A_s	system availability.
A_0	system availability constraint value.
A_i	availability of subsystem i.
m	number of subsystems in the system.

N	maximum number of allowable components in each subsystem.
n_i	number of components in subsystem i.
r_i	number of repair teams for subsystem i.
r	$=(r_1, r_2,\ldots, r_m)$, vector of repair teams for the system.
n	$=(n_1, n_2,\ldots, n_m)$, vector of redundancy allocated in the system.
λ_i	inherent failure rate of the component in subsystem i.
μ_i	repair rate of the component in subsystem i.
C_s	system cost.
C_i^c	unit cost of components in subsystem i.
C_i^r	unit cost of repair teams in subsystem i.
j	number of working components in subsystem i.

1. Introduction

Dependability of modern systems is optimized at the design stage, in order to satisfy the customer requirements [1]. The dependability design optimization of a system considers all five concepts building up dependability, i.e., reliability, availability, maintainability, safety, and cost (RAMS+C) [2]. The main focus, then, depends on the target of the designer and the design problem is formulated as an optimization problem. Evolutionary optimization methods have been shown to provide very good results in solving RAMS+C optimization problems. In [3], a particle-based simplified swarm optimization (PSSO) has been implemented to maximize the system reliability. In [4], a novel genetic algorithm has been developed for the optimal redundancy allocation problem in multi-state systems. The objective was to maximize the system availability under the cost constraint. A penalty guided stochastic fractal search (PSFS) has been proposed in [5] to maximize the system reliability or minimize the system cost subject to a system reliability constraint. Various evolutionary optimization methods have been applied in [6,7] to solve single objective system reliability optimization problems.

Some authors developed genetic algorithms for optimizing multiple targets of RAMS+C [2,8–11]. In [12], a new adaptive particle swarm optimization has been developed for system reliability-redundancy allocation, considering reliability, cost, volume and weight within a multi-objective problem solved by using the weighted-sum method. The series-parallel connection is one of the most used system configurations in industry. The optimization of such a system has been investigated considering various issues [13–17]. In [13], the Pareto front for system reliability and cost has been found using harmony search and particle swarm optimization. The cold-standby strategy has been considered in [14] and the problem has been solved using a penalty-guided fractal search algorithm. In [15], optimization models of selective maintenance strategies for multi-state cases have been proposed. The authors used a genetic algorithm to solve the problems. A new multi-objective redundancy allocation strategy considering different types of redundant components have been proposed in [16]. The system cost and availability under failure dependencies has been investigated in [17] using a genetic algorithm. Availability is the relevant performance indicator for repairable systems [18,19] and failure dependencies can impact the system availability [17,20–25].

The aim of the present work is to propose an adaptive cuckoo optimization algorithm (AA-COA) for minimizing system cost subject to a system availability constraint. Four scenarios are considered for the system availability constraint, i.e., independence, weak dependence, linear dependence and strong dependence of the failure behaviors of the subsystems. The approach developed is tested on a series-parallel system.

The structure of the paper is as follows. Section 2 presents the optimization. Section 3 describes the proposed solution approach. A numerical case study involving six subsystems connected in series-parallel is presented in Section 4. Section 5 provides a discussion on the results obtained and on the applicability of the approach proposed. Finally, the last section draws some conclusions and perspectives.

2. Problem description

Let us consider a system consisting of *m* subsystems connected in series-parallel [17]. The failure behaviors of these subsystems are dependent. For simplicity of the analysis, but with no loss of generality in the optimization problem, it is assumed that the components of the subsystems can only be in two states, working or failed, the repair rate of each component in each subsystem is constant, each repair team can repair only one component at a time, and the failure rates of the operating components increase with the number of failed components. The goal of the design optimization problem is to minimize the system cost:

$$\text{Minimize} \quad C_s(n,r) = \sum_{i=1}^{m} \left(n_i C_i^c + r_i C_i^r \right) \tag{1}$$

subject to

$$\begin{gathered} A_s(n,r) \geq A_0 \\ n_i, r_i \in \mathbb{Z}^+ \\ r_i \leq n_i \\ i = 1, 2, \ldots, m \end{gathered} \tag{2}$$

where $n=(n_1,n_2,\ldots,n_m)$ and $r=(r_1,r_2,\ldots,r_m)$ are the vectors of redundancy and repair teams allocation for the system, respectively; *m* is the number of subsystems in the system, C_i^c and C_i^r are the unit cost of components and unit cost of repair teams in subsystem *i*, respectively; C_s is the system cost; A_s is the system availability; and A_0 is the system availability constraint value.

The mathematical expression of the system availability A_s is written according to the dependence function. The dependence refers to the interactions between the failures of the components in the system. A graphical parameter modeling these interactions defines the level of dependence [20,26]. Four main cases (classes of dependence) can be considered [17]:

Case 1: Independence

$$A_s = \prod_{i=1}^{m} \left\{ 1 - \left[1 + \sum_{j=1}^{n_i - r_i} \frac{r_i^{\,j}}{j!} \left(\frac{\mu_i}{\lambda_i} \right)^j + \sum_{j=n_i - r_i + 1}^{n_i} \frac{r_i^{\,n_i - r_i} r_i!}{j!(n_i - j)!} \left(\frac{\mu_i}{\lambda_i} \right)^j \right]^{-1} \right\} \tag{3}$$

Case 2: Linear dependence

$$A_s = \prod_{i=1}^{m} \left\{ 1 - \left[1 + \sum_{j=1}^{n_i - r_i} r_i^{\,j} \left(\frac{\mu_i}{\lambda_i} \right)^j + \sum_{j=n_i - r_i + 1}^{n_i} \frac{r_i^{\,n_i - r_i} r_i!}{(n_i - j)!} \left(\frac{\mu_i}{\lambda_i} \right)^j \right]^{-1} \right\} \tag{4}$$

Cases 3 and 4: Weak dependence ($0 < l < 1$) and strong dependence ($l > 1$)

$$A_s = \prod_{i=1}^{m} \left\{ 1 - \left[1 + \sum_{j=1}^{n_i - r_i} r_i^{\,j} (j!)^{l-1} \left(\frac{\mu_i}{\lambda_i} \right)^j + \sum_{j=n_i - r_i + 1}^{n_i} \frac{r_i^{\,n_i - r_i} r_i!(j!)^{l-1}}{(n_i - j)!} \left(\frac{\mu_i}{\lambda_i} \right)^j \right]^{-1} \right\} \tag{5}$$

where j is the number of working components, μ_i is the repair rate of a component in subsystem i and λ_i is the failure rate of a component in subsystem i. It can be seen that the above metrics are nonlinear and include integer decision variables. Also, the number of repair teams allocated to subsystem i is taken to be less than or equal to the number of components ($r_i \leq n_i$).

3. Adaptive cuckoo optimization algorithm

The basic cuckoo optimization algorithm (COA) has been developed by Rajabioun [27]. It is inspired by the lifestyle of the cuckoo bird in laying eggs and in migrating. The cuckoo is capable of laying eggs only in nests of other bird species, called host nests. The eggs of the host nests are mimicked by the cuckoos to increase the discretion. Sometimes, the birds of the host nests recognize the cuckoos' eggs and destroy it. Furthermore, some growing cuckoos' chicks are thrown out from the nests by the host birds or starve, because they are bigger than the other birds and need more food. When the cuckoos' chicks become mature, they move away to a better living place. The original cuckoo optimization algorithm is based on egg laying radius and *k*-means clustering for creating potential solutions for the problem to be optimized.

The cuckoo optimization algorithm has proven its effectiveness in solving several engineering optimization problems, such as optimal controller design [27], energy production cost minimization [28,29], optimal data clustering [30], optimal machining parameters [31–33], optimal job scheduling [34], and optimal replacement strategy of obsolete industrial components [35,36]. The main disadvantage of COA is the difficulty of handling integer variables and strongly nonlinear constraints. In this respect, the basic COA cannot be implemented for system cost optimization with availability constraint. For this reason, in this paper the algorithm is modified and adapted to an adaptive cuckoo optimization algorithm (AA-COA). The main steps of AA-COA are described as follows.

Step 1: Generate a random area of cuckoos.

An area consisting of *M* peer habitats is considered at each iteration as follows:

$$Area = \begin{cases} Habitat_1 = \begin{cases} Nest_a = \{n_1, n_2, ..., n_m\} \\ Nest_b = \{r_1, r_2, ..., r_m\} \end{cases} \\ Habitat_2 = \begin{cases} Nest_a = \{n_1, n_2, ..., n_m\} \\ Nest_b = \{r_1, r_2, ..., r_m\} \end{cases} \\ \vdots \\ Habitat_M = \begin{cases} Nest_a = \{n_1, n_2, ..., n_m\} \\ Nest_b = \{r_1, r_2, ..., r_m\} \end{cases} \end{cases} \quad (6)$$

where $Nest_a$ is the vector containing the numbers of redundant components and $Nest_b$ is the vector of the numbers of repair teams. One cuckoo only is considered in each habitat and one egg only in each nest.

At first, the random nest of redundant components (*Nest*$_a$) in each habitat is generated, $n_i \in \{1,\ldots,N\}$, where N is the maximum number of redundant components in each subsystem. The real numbers are rounded to the nearest integer values.

Then, a random nest of repair teams (*Nest*$_b$) $r_1 \in \{1,\ldots,n_1\}$; $r_2 \in \{1,\ldots,n_2\};\ldots;$ $r_m \in \{1,\ldots,n_m\}$ is generated, accounting for the constraint $r_i \leq n_i$.

Step 2: Evaluate the system cost and handle the system availability constraint.

The numbers of redundant components and repair teams generated in Step 1 are introduced into Eqs. (1) and (2). The constrained problem is transformed to an unconstrained one by using a penalized function [37]:

$$P(n,r) = -w \cdot \text{Min}\{0, A_s(n,r) - A_0\} \tag{7}$$

where w is the penalty value allowing to handle the degree of constraint violation. The feasible constraint values are reset as zero. In this paper, this value changes (adaptive value) during successive iterations; if in all the previous iterations the best solution was infeasible, then, the penalty value is increased, if in all the previous iterations the best solution was feasible, then, w is decreased; otherwise, it is kept constant.

Therefore, the penalized function is written as follows:

$$\tilde{C}_s(n,r) = C_s(n,r) - w \cdot \text{Min}\{0, A_s(n,r) - A_0\} \tag{8}$$

Step 3: Select the best habitat (solution) and destroy worst habitats.

The habitat with a minimum system cost is selected and saved. For the remainder habitats, it is assumed that the eggs have been recognized by the host birds and are destroyed. Thus, these habitats are considered worst. On the other hand, it is assumed that one cuckoo's chick only will survive and the others starve.

Step 4: Migrate the cuckoo

The cuckoo’s chick becomes mature and migrates for mating. Then, the best solution saved in Step 2 is introduced in the next iteration (new area) for improving the fitness. It should be noted that in the next iteration the number of new random habitats is (M–1).

Step 5: Repeat Steps 1 to 4 until the number of iterations is reached, than, the minimum system cost with the best vectors of redundant components and repair teams are displayed.

Algorithm below shows the pseudo-code of the implemented AA-COA, and Figure 1 shows its flowchart.

Algorithm 1 – Pseudo-code of the implemented AA-COA.

```
1: Input the parameters: $A_0$, $N$, $w$, $M$, $N_{Iter}$.
2: While $z \leq N_{Iter}$
3: Generate a random area according to Eq. (6).
4: Evaluate the system cost (each habitat) and constraint handling according to
   Eq. (8).
5: Increase, decrease or keep constant the value of $w$.
6: Select the best habitat and destroy the eggs of the worst habitats.
7: Migrate the cuckoo by introducing the best solution in the next iteration.
8: End While
9: Display the minimum system cost and the vectors of the numbers of
   redundant components and repair teams.
```

Insert Figure 1 – Flowchart of the proposed AA-COA.

4. Numerical case study

The system considered (Figure 2) consists of 6 subsystems (m=6) connected in series-parallel [17]. Therefore, the optimization problem involves 12 integer decisions variables (6 for the numbers of redundant components and 6 for the numbers of repair teams to allocate to the 6 subsystems). Three values of system availability constraint (A_0) are considered: 0.90, 0.95, and 0.99. The maximum number of allowable redundant components in each subsystem is 15 (N=15). The adopted values of l in Eq. (5) are 0.5 for weak dependence and 1.5 for strong dependence. Table 1 summarizes the relevant data of the system.

Insert Figure 2 – Series-parallel system.

Insert Table 1 – Data of the system.

5. Results and discussion

The adaptive cuckoo optimization algorithm has been programmed using MATLAB 2015 and run with 20 habitats and over 200 iterations (4000 function evaluations) on a PC Intel Pentium G620 (2.60 GHz, 4 GB of RAM, Sandy Bridge, 3Mo Cache, Windows 7, 32 bits). The total computation time for the optimization search has been of 33.41s in case of independence, 50.06s in case of linear dependence, 3911s in case of weak dependence, and 4822s in case of strong dependence.

Insert Table 2 – Optimal solutions when A_0=0.90.

Insert Table 3 – Optimal solutions when A_0=0.95.

Insert Table 4 – Optimal solutions when A_0=0.99.

Tables 2–4 summarize the results obtained by GA in [17] and by the AA-COA proposed in this work, for A_0=0.90, A_0=0.95 and A_0=0.99, respectively. The best results are bolded in the Table. In Table 2, the system costs (in arbitrary units) when A_0=0.90 are 1355 for the independence case, 1225 for the linear dependence, and 160 for the strong dependence. These results are similar between the GA and the AA-COA. However, the system cost in weak dependence is found to be 1285 by the GA, and 1235 by the AA-COA.

Table 3 reports the optimal solutions when A_0=0.95 and it can be observed that the results of AA-COA improve those obtained by the GA. The system costs are GA: 1615, AA-COA: 1355 in case of independence, GA: 1410, AA-COA: 1390 in case of weak dependence, GA: 1275, AA-COA: 1270 in case of linear dependence, and GA: 1185, AA-COA: 1175 in case of strong dependence.

When the system availability constraint value is 0.99 (see Table 4), the system costs obtained by GA and AA-COA are as follows: 2135 and 2125 in case of independence, 1810 and 1770 in case of weak dependence, 1590 and 1565 in case of linear dependence, 1410 and 1405 in case of strong dependence. In all cases, AA-COA outperforms the GA solution found in [17].

The comparison of the results clearly shows that the newly proposed AA-COA performs well, providing better results than GA. Moreover, GA used 25000 function evaluations in the search, whereas AA-COA only 4000.

6. Conclusions

In this paper, a novel optimization algorithm has been presented for minimizing system cost subject to an availability constraint and in case of dependencies in the failure behaviors of the subsystems. Specifically, an adaptive cuckoo optimization algorithm (AA-COA) has been developed. Four scenarios of failure dependencies have been considered: independence, weak dependence, linear dependence and strong dependence.

A series-parallel system consisting of six subsystems has been worked out using the proposed algorithm under three values of the system availability constraint. The results obtained demonstrate the effectiveness of the optimization algorithm.

Further work will focus on the development of a multi-objective AA-COA, for multi-objective RAMS+C problems and the consideration of uncertainties.

References

[1] Habchi G, Barthod C. An overall methodology for reliability prediction of mechatronic systems design with industrial application. Reliab Eng Syst Saf 2016;155:236–54. doi:10.1016/j.ress.2016.06.013.

[2] Torres-Echeverría AC, Martorell S, Thompson HA. Design optimization of a safety-instrumented system based on RAMS+C addressing IEC 61508 requirements and diverse redundancy. Reliab Eng Syst Saf 2009;94:162–79. doi:10.1016/j.ress.2008.02.010.

[3] Huang C-L. A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems. Reliab Eng Syst Saf 2015;142:221–30. doi:http://dx.doi.org/10.1016/j.ress.2015.06.002.

[4] Sun MX, Li YF, Zio E. A novel genetic algorithm developed on a reduced search space for optimal redundancy allocation in multi-state series-parallel systems. ESREL 2015 Eur. Saf. Reliab. Conf., Zurich, Switzerland: n.d.

[5] Mellal MA, Zio E. A penalty guided stochastic fractal search approach for system reliability optimization. Reliab Eng Syst Saf 2016;152:213–227.

[6] Mellal MA, Zio E. System reliability-redundancy allocation by evolutionary computation. 2017 2nd Int. Conf. Syst. Reliab. Saf., Milan, Italy: IEEE; 2017, p. 15–9.

[7] Mellal MA, Williams EJ. Large scale reliability-redundancy allocation optimization problem using three soft computing methods. Model. Simul. based Anal. Reliab. Eng., CRC Press Francis & Taylor; 2018, p. 199–214.

[8] Marseguerra M, Zio E, Martorell S. Basics of genetic algorithms optimization for RAMS applications. Reliab Eng Syst Saf 2006;91:977–91. doi:10.1016/j.ress.2005.11.046.

[9] Martorell S, Villanueva JF, Carlos S, Nebot Y, Sánchez A, Pitarch JL, et al. RAMS+C informed decision-making with application to multi-objective optimization of technical specifications and maintenance using genetic algorithms. Reliab Eng Syst Saf 2005;87:65–75. doi:10.1016/j.ress.2004.04.009.

[10] Konak A, Coit DW, Smith AE. Multi-objective optimization using genetic algorithms: A tutorial. Reliab Eng Syst Saf 2006;91:992–1007. doi:10.1016/j.ress.2005.11.018.

[11] Chebouba BN, Mellal MA, Adjerid S. Multi-objective system reliability Optimization in a power plant. 3rd Int. Conf. Electr. Sci. Technol. Maghreb, Algiers, Algeria: 2018.

[12] Mellal MA, Zio E. An adaptive particle swarm optimization method for multi-objective

system reliability optimization. J Risk Reliab 2019. doi:10.1177/1748006X19852814.

[13] Zhao J, Si S, Cai Z, Su M, Wang W. Multiobjective optimization of reliability-redundancy allocation problems for serial parallel-series systems based on importance measure. J Risk Reliab 2019. doi:10.1177/1748006X19844785.

[14] Juybari MN, Abouei Ardakan M, Davari-Ardakani H. A penalty-guided fractal search algorithm for reliability–redundancy allocation problems with cold-standby strategy. Proc Inst Mech Eng Part O J Risk Reliab 2019. doi:10.1177/1748006X19825707.

[15] Dao CD, Zuo MJ, Pandey M. Selective maintenance for multi-state series-parallel systems under economic dependence. Reliab Eng Syst Saf 2014. doi:10.1016/j.ress.2013.09.003.

[16] Safari J. Multi-objective reliability optimization of series-parallel systems with a choice of redundancy strategies. Reliab Eng Syst Saf 2012;108:10–20. doi:10.1016/j.ress.2012.06.001.

[17] Hu L, Yue D, Li J. Availability analysis and design optimization for a repairable series-parallel system with failure dependencies. Int J Innov Comput Inf Control 2012;8:6693–705. doi:10.4156/aiss.vol3.

[18] Neil M, Marquez D. Availability modelling of repairable systems using Bayesian networks. Eng Appl Artif Intell 2012;25:698–704. doi:10.1016/j.engappai.2010.06.003.

[19] Mellal MA, Zio E. Availability optimization of parallel-series system by evolutionary computation. 3rd Int. Conf. Syst. Reliab. Saf., Barcelona, Spain: 2018.

[20] Yu H, Chu C, Châtelet E, Yalaoui F. Reliability optimization of a redundant system with failure dependencies. Reliab Eng Syst Saf 2007;92:1627–34. doi:10.1016/j.ress.2006.09.015.

[21] Yuan L. Reliability analysis for a k-out-of-n:G system with redundant dependency and repairmen having multiple vacations. Appl Math Comput 2012;218:11959–69. doi:10.1016/j.amc.2012.06.006.

[22] Yu H, Chu C, Châtelet É. Availability optimization of a redundant system through dependency modeling. Appl Math Model 2014;38:4574–85. doi:10.1016/j.apm.2014.03.006.

[23] Habib M, Chehade H, Yalaoui F, Chebbo N, Jarkass I. Availability optimization of a redundant dependent system using genetic algorithm. IFAC-PapersOnLine 2016;49:733–8.

[24] Siju KC, Kumar M. System reliability estimation and cost analysis of series-parallel

systems in the presence of repair dependence function. Int J Reliab Saf 2016;10:48–71.

[25] Lin YH, Li YF, Zio E. Component importance measures for components with multiple dependent competing degradation processes and subject to maintenance. IEEE Trans Reliab 2016;65:547–57. doi:10.1109/TR.2015.2500684.

[26] Habib M, Yalaoui F, Chehade H, Jarkass I, Chebbo N. Multi-objective design optimisation of repairable k-out-of-n subsystems in series with redundant dependency. Int J Prod Res 2017. doi:10.1080/00207543.2017.1346319.

[27] Rajabioun R. Cuckoo optimization algorithm. Appl Soft Comput 2011;11:5508–5518.

[28] Mehdinejad M, Mohammadi-Ivatloo B, Dadashzadeh-Bonab R. Energy production cost minimization in a combined heat and power generation systems using cuckoo optimization algorithm. Energy Effic 2016. doi:10.1007/s12053-016-9439-6.

[29] Mellal MA, Williams EJ. Cuckoo optimization algorithm with penalty function for combined heat and power economic dispatch problem. Energy 2015;93:1711–8. doi:10.1016/j.energy.2015.10.006.

[30] Amiri E, Mahmoudi S. Efficient protocol for data clustering by fuzzy cuckoo optimization algorithm. Appl Soft Comput 2016;41:15–21.

[31] Mellal MA, Williams EJ. Parameter optimization of advanced machining processes using cuckoo optimization algorithm and hoopoe heuristic. J Intell Manuf 2016;27:927–42.

[32] Mellal MA, Williams EJ. Total production time minimization of a multi-pass milling process via cuckoo optimization algorithm. Int J Adv Manuf Technol 2016;87:747–754. doi:10.1007/s00170-016-8498-3.

[33] Mellal MA, Williams EJ. Cuckoo optimization algorithm for unit production cost in multi-pass turning operations. Int J Adv Manuf Technol 2015;76:647–56. doi:10.1007/s00170-014-6309-2.

[34] Rabiee M, Sajedi H. Job scheduling in grid computing with cuckoo optimization algorithm. Int J Comput Appl 2013;62:38–44.

[35] Mellal MA, Adjerid S, Williams EJ, Benazzouz D. Optimal replacement policy for obsolete components using cuckoo optimization algorithm based-approach: Dependability context. J Sci Ind Res (India) 2012;71:715–21.

[36] Mellal MA, Adjerid S, Williams EJ. Optimal selection of obsolete tools in manufacturing systems using cuckoo optimization algorithm. Chem Eng Trans 2013;33:355–60. doi:10.3303/CET1333060.

[37] Chootinan P, Chen A. Constraint handling in genetic algorithms using a gradient-based repair method. Comput Oper Res 2006;33:2263–81. doi:10.1016/j.cor.2005.02.002.

Figure captions

Figure 1 – Flowchart of the proposed AA-COA.

Figure 2 – Series-parallel system.

Table captions

Table 1 – Data of the system.

Table 2 – Optimal solutions when A_0=0.90.

Table 3 – Optimal solutions when A_0=0.95.

Table 4 – Optimal solutions when A_0=0.99.

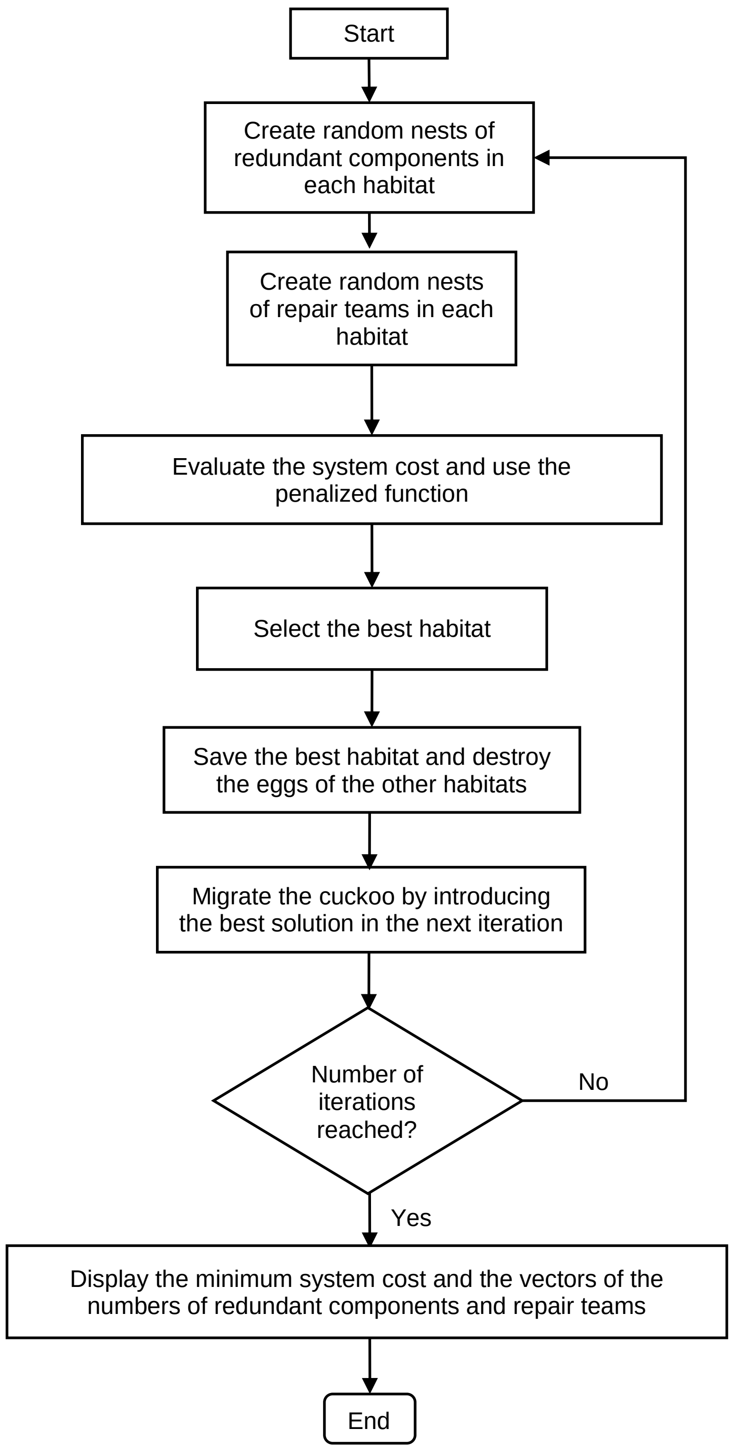
Start
Create random nests of redundant components in each habitat
Create random nests of repair teams in each habitat
Evaluate the system cost and use the penalized function
Select the best habitat
Save the best habitat and destroy the eggs of the other habitats
Migrate the cuckoo by introducing the best solution in the next iteration
Number of iterations reached?
No
Yes
Display the minimum system cost and the vectors of the numbers of redundant components and repair teams
End

Figure 1 – Flowchart of the proposed AA-COA.

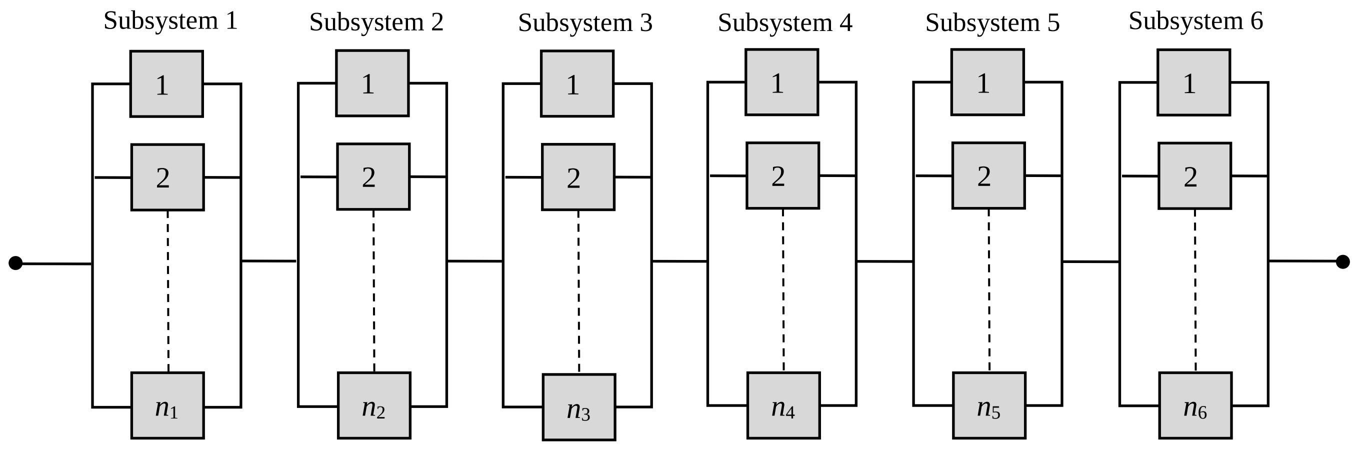

Figure 2 – Series-parallel system.

Table 1 – Data of the system.

Subsystem *i*	λ_i	μ_i	C_i^c	C_i^r
1	0.03	0.10	40	15
2	0.04	0.13	50	20
3	0.05	0.14	30	10
4	0.06	0.20	70	30
5	0.07	0.18	65	25
6	0.09	0.27	80	35

Table 2 – Optimal solutions when A_0=0.90.

Subsystem *i*	Independence (n_i,r_i)		Weak dependence (n_i,r_i)		Linear dependence (n_i,r_i)		Strong dependence (n_i,r_i)	
	GA [15]	AA-COA	GA [15]	AA-COA	GA [15]	AA-COA	GA [15]	AA-COA
1	(3,3)	(3,3)	(3,1)	(3,2)	(3,2)	(3,2)	(3,1)	(3,1)
2	(3,2)	(3,2)	(3,2)	(3,2)	(3,2)	(3,2)	(2,2)	(2,2)
3	(4,3)	(4,3)	(3,2)	(4,2)	(3,2)	(3,2)	(3,2)	(3,2)
4	(3,2)	(3,2)	(3,2)	(2,2)	(2,2)	(2,2)	(2,2)	(2,2)
5	(3,3)	(3,3)	(3,3)	(3,2)	(3,2)	(3,2)	(3,2)	(3,2)
6	(3,2)	(3,2)	(3,2)	(3,2)	(2,2)	(2,2)	(2,2)	(2,2)
C_s	1355	1355	1285	**1235**	1125	1125	1060	1060
A_s	0.9025	0.9025	0.9051	0.9031	0.9020	0.9020	0.9031	0.9031

Table 3 – Optimal solutions when A_0=0.95.

Subsystem i	Independence (n_i,r_i)		Weak dependence (n_i,r_i)		Linear dependence (n_i,r_i)		Strong dependence (n_i,r_i)	
	GA [15]	AA-COA	GA [15]	AA-COA	GA [15]	AA-COA	GA [15]	AA-COA
1	(4,3)	(4,3)	(3,3)	(4,2)	(3,2)	(3,2)	(3,2)	(3,1)
2	(3,3)	(4,3)	(3,2)	(3,3)	(3,3)	(3,2)	(3,2)	(3,1)
3	(4,3)	(4,3)	(4,2)	(4,2)	(3,3)	(3,2)	(3,2)	(3,2)
4	(4,3)	(3,3)	(3,3)	(3,2)	(3,1)	(3,1)	(3,1)	(3,1)
5	(4,3)	(4,3)	(3,3)	(3,3)	(3,2)	(3,3)	(3,1)	(3,2)
6	(3,3)	(3,3)	(3,3)	(3,2)	(3,2)	(3,2)	(3,1)	(3,1)
C_s	1615	**1595**	1410	**1390**	1275	**1270**	1185	**1175**
A_s	0.9502	0.9506	0.9504	0.9502	0.9514	0.9503	0.9528	0.9526

Table 4 – Optimal solutions when A_0=0.99.

Subsystem i	Independence (n_i,r_i)		Weak dependence (n_i,r_i)		Linear dependence (n_i,r_i)		Strong dependence (n_i,r_i)	
	GA [15]	AA-COA	GA [15]	AA-COA	GA [15]	AA-COA	GA [15]	AA-COA
1	(5,3)	(5,4)	(4,4)	(4,4)	(4,2)	(4,3)	(3,3)	(4,1)
2	(5,4)	(5,4)	(4,3)	(4,3)	(4,2)	(4,2)	(3,2)	(3,2)
3	(5,4)	(5,5)	(5,3)	(4,4)	(4,2)	(4,3)	(4,2)	(4,3)
4	(5,3)	(4,3)	(4,4)	(4,3)	(4,2)	(3,2)	(3,2)	(3,2)
5	(5,4)	(5,4)	(4,4)	(4,3)	(4,3)	(4,2)	(4,3)	(4,2)
6	(5,3)	(5,4)	(4,2)	(4,3)	(3,3)	(4,2)	(3,2)	(3,2)
C_s	2135	**2125**	1810	**1770**	1590	**1565**	1410	**1405**
A_s	0.9904	0.9900	0.9901	0.9900	0.9910	0.9902	0.9901	0.9901