Mellal Mohamed Arezki (Orcid ID: 0000-0003-0667-8851)

Cost minimization of repairable systems subject to availability constraints using efficient cuckoo optimization algorithm

Mohamed Arezki Mellal[a,b] *, Enrico Zio[c, d], and Edward J. Williams[e, f]

[a]*LMSS, Faculty of Technology, M'Hamed Bougara University, Boumerdes, Algeria*

[b]*Center for Advanced Life Cycle Engineering (CALCE), University of Maryland, College Park, MD 20742, USA*

(*) Corresponding author: mellal.mohamed@gmail.com, mamellal@umd.edu, mellal.mohamed@univ-boumerdes.dz

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

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

[e]*Industrial and Manufacturing Systems Engineering Department, College of Engineering and Computer Science, University of Michigan, Dearborn 48126, USA*

[f]*Decision Sciences, College of Business, University of Michigan, Dearborn 48126, USA*

Abstract

System availability is a key element for any industry. System designers and operators try to do their best to maintain the required availability of the systems, to avoid production stoppages. They set up and undertake different maintenances and these interventions imply cost. Therefore, the goal is to minimize the cost, but considering the constraint of the availability requirement. The problem involves three main aspects: redundancy allocation, component failure rates and repair rates. In this paper, a novel solution approach is proposed, based on an efficient cuckoo optimization algorithm (EF-COA). Two numerical case studies are solved and the results confirm the effectiveness of the approach proposed.

Keywords: System cost; availability requirement; repairable systems; efficient cuckoo optimization algorithm (EF-COA).

Notations

A_S	system availability.
m	number of subsystems in the system.

A_i	availability of each component in subsystem i, $1 \le i \le m$.
A	$=(A_1, A_2,\ldots, A_m)$, vector of component availabilities for the system.
A_S^*	system availability requirement.
n_i	number of components in subsystem i, $1 \le i \le m$.
n	$=(n_1, n_2,\ldots, n_m)$, vector of redundancy allocation for the system.
R_i	$=1-(1-r_i)^{n_i}$, reliability of the ith subsystem, $1 \le i \le m$.
λ_i	failure rate of each component in subsystem i, $1 \le i \le m$.
λ	$=(\lambda_1, \lambda_2,\ldots, \lambda_m)$, vector of component failure rates for the system.
μ_i	repair rate of each component in subsystem i, $1 \le i \le m$.
μ	$=(\mu_1, \mu_2,\ldots, \mu_m)$, vector of component repair rates for the system.
M	number of constraints.
g_j	*jth* constraint function, $j=1,\ldots, M$.
b	vector of resource limitation.
T	operating time during which the component must not fail (mission time, T=1000).
w_i	weight of each component in subsystem i, $1 \le i \le m$.
β_i, α_i	parameters representing physical features (shaping and scaling factors, respectively) of each component at subsystem i, $1 \le i \le m$.
P	limitation on product of weight and volume.
λ_{iL}, μ_{iL}	lower limits on the failure rate and repair rate of each component in subsystem i, $1 \le i \le m$.
λ_{iU}, μ_{iU}	upper limits on the failure rate and repair rate of each component in subsystem i, $1 \le i \le m$.

1. Introduction

A competitive industrial plant or infrastructure requires a highly dependable system with minimum functioning cost. The system dependability is a challenge that simultaneously incorporates reliability, availability, maintainability and safety (RAMS).[1] The focus of the designer depends on the target, criteria and system nature, such as nuclear power plants[2,3] and network systems (e.g. electric power transmission/distribution systems, water/oil/gas distribution systems, computer/communication systems, rail/road transportation systems).[4]

Higher RAMS allocation improves system dependability, but also increases system cost.[5] Most of RAMS problems are described as optimization problems with single or multi objective functions subject to the constraints fixed by the specifications (e.g. weight and volume). Evolutionary computation methods, also referred to artificial intelligence methods (AI), have successfully dealt with RAMS problems. In[6-13], the authors used the artificial bee colony (ABC)[6], immune based algorithm (IA)[7], differential evolution with Lévy flight (DE)[8], the biogeography based optimization algorithm (BBO)[9], particle swarm optimization (PSO)[10,11], penalty guided stochastic fractal search[12], and the gray wolf optimizer algorithm[13] for system reliability models. In[1], a multi-objective approach based on genetic algorithm has been presented for simultaneously dealing with the following objectives: system reliability, system maintainability, system safety and cost (RAMS&C). In[14,15], a method was proposed combining Tabu search and genetic algorithm (TA–GA) for minimizing the system cost under availability constraint. An ant algorithm for single and multi-objective system reliability problem has been developed in[16]. Recently, three evolutionary computation methods have been applied to a pharmaceutical plant in order to increase the overall system reliability[17]. The maintainability of a system by considering the failure and repair processes has been investigated in[18], whereas a new mathematical model of reliability for multi-state degraded repairable system has been proposed in[19].

The great challenge is to effectively deal with the dependability of the system and improve its elements. In this paper, we propose a novel solution approach for minimizing the system cost under system availability constraints, by resorting to a modification of the basic cuckoo optimization algorithm[20], in the present work called efficient cuckoo optimization algorithm (EF-COA). The remainder of the paper is organized as follows. Section 2 describes the system cost minimization problem subject to availability constraint. Section 3 presents the schemes of the EF-COA. In Section 4, two numerical case studies are presented. Finally, conclusions are drawn at closure.

2. Problem description

The general mathematical formulation of the considered cost minimization problem of repairable systems is given as follows[14,15]:

$$\text{Minimize} \quad C_S(n,\lambda,\mu) = C_S(n_1,n_2,...,n_m;\ \lambda_1,\lambda_2,...,\lambda_m;\mu_1,\mu_2,...,\mu_m) \tag{1}$$

where $C_S(\bullet)$ is the total system cost, n_i is the number of redundant components in the ith subsystem, λ_i is the failure rate of the components in the ith subsystem, and μ_i is the repair rate of the components in the ith subsystem,

subject to

$$g_j(n_1,n_2,...,n_m) \le b \tag{2}$$

$$A_S(n,\lambda,\mu) \ge A_S^* \tag{3}$$

$$\begin{aligned} & n_i \ge 1; n_i \in \mathbf{Z}^+ \\ & \lambda_i \in [\lambda_{iL}, \lambda_{iU}] \subset \Re^+ \\ & \mu_i \in [\mu_{iL}, \mu_{iU}] \subset \Re^+ \\ & i = 1,2,...,m \end{aligned}$$

where $g(\bullet)$ is the set of constraints, b is the vector of resource limitation, $A_S(\bullet)$ is the system availability, A_S^* is the system availability requirement, and m is the number of subsystems in the system.

3. Efficient cuckoo optimization algorithm

The cuckoo optimization algorithm (COA) is a bio-inspired evolutionary optimization method developed by Rajabioun.[20] The basic principles are based on the lifestyle and behavior of the birds cuckoos for their reproduction. Several works available in the literature used the main concepts of this algorithm for solving various engineering problems, such as multivariable controller design[20], replacement of obsolete components in industrial plants[21,22], data clustering[23], machining parameters[24-26], job scheduling[27], warranty period definition[28], nonconvex combined heat and power economic dispatch[29], and recognition of control chart patterns[30]. The standard cuckoo optimization algorithm (COA) implies the major steps reported in the Appendix.[20]

To effectively solve the system cost minimization subject to the availability constraint described in Section 2, the basic cuckoo optimization algorithm is improved for better performance and the new approach is called EF-COA. The new steps of the algorithm are described as follows:

Step 1: Random initialization of nests.

A fixed number of habitats and one cuckoo for each habitat only are considered. A random number of nests is generated for each habitat separately. Each nest represents a potential solution as follows:

$$\begin{cases} Habitat_1 = \begin{cases} Nest_{1,1} = C_{S_{1,1}}(n,\lambda,\mu) = C_{S_{1,1}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ Nest_{2,1} = C_{S_{2,1}}(n,\lambda,\mu) = C_{S_{2,1}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ \vdots \qquad \vdots \qquad \vdots \\ Nest_{k_1,1} = C_{S_{k_1,1}}(n,\lambda,\mu) = C_{S_{k_1,1}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \end{cases} \\ Habitat_2 = \begin{cases} Nest_{1,2} = C_{S_{1,2}}(n,\lambda,\mu) = C_{S_{1,2}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ Nest_{2,2} = C_{S_{2,2}}(n,\lambda,\mu) = C_{S_{2,2}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ \vdots \qquad \vdots \qquad \vdots \\ Nest_{k_2,2} = C_{S_{k_2,2}}(n,\lambda,\mu) = C_{S_{k_2,2}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \end{cases} \\ \vdots \qquad \qquad \vdots \\ Habitat_H = \begin{cases} Nest_{1,H} = C_{S_{1,H}}(n,\lambda,\mu) = C_{S_{1,H}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ Nest_{2,H} = C_{S_{2,H}}(n,\lambda,\mu) = C_{S_{2,H}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ \vdots \qquad \vdots \qquad \vdots \\ Nest_{k_H,H} = C_{S_{k_2,H}}(n,\lambda,\mu) = C_{S_{k_2,H}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \end{cases} \end{cases} \tag{4}$$

$$k_1, k_2, ..., k_H \in \{2,3,4,...,K\}$$

where H is the number of habitats, K is the maximum number of nests which can be generated in the habitats.

Step 2: Evaluate the potential solutions.

The objective function value of each nest in each habitat is evaluated as follows:

$$Nest(n,\lambda,\mu) = C_S(n,\lambda,\mu) = C_S(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \tag{5}$$

Step 3: Constraint handling.

To deal with the inequality constraint described in Eqs. (2)–(3), the penalty method is used and the constrained problem is converted to an unconstrained one, by adding penalty terms as follows[29,31,32]:

$$NEST(n,\lambda,\mu) = Nest(n,\lambda,\mu) + \Omega\left\langle A_S(n,\lambda,\mu)\right\rangle + \Phi_j \sum_{j=1}^{M} \left\langle g_j(n)\right\rangle \tag{6}$$

where $NEST(n, \lambda, \mu)$ is the fitness value, $Nest(n, \lambda, \mu)$ is the objective function value, $A_S(n, \lambda, \mu)$ is the system availability constraint, $g_j(n)$ are the other inequality constraints, M is the number of constraints, Ω and Φ are penalty parameters. The values of these parameters are set by trial-and-error and based on experience. The operator $< \cdot >$ denotes the absolute value of the operand, if it is negative; otherwise it is zero. The real numbers of the vector of redundancy allocation n are rounded to the nearest integer value.

Step 4: Identification of the best solution (minimum cost) and migration.

All the nests of each habitat are classified and the best one is identified. The worst nests in each habitat mean that the eggs were recognized by the host birds and have been destroyed. Therefore, the best habitat includes the identified best nest (minimum system cost) and implies that this habitat represents the migration target for the cuckoo:

$$Habitat_{Best} = \left\{ Nest_{Best} = C_{S_{Best}}(n,\lambda,\mu) = C_{S_{Best}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \right. \tag{7}$$

where $Habitat_{Best}$ is the best habitat with the best $Nest_{Best}$.

Step 5: Use the best solution of the last previous cuckoo's generation (iteration) in the next one.

The best nest (best solution) of the last previous cuckoo's generation (i.e. iteration) is considered a fixed nest for each habitat in the current iteration and the remaining nests are randomly generated. This step improves the solution's quality from one iteration to the next:

$$\begin{cases} Habitat_1 = \begin{cases} Nest_{Best} = C_{S_{Best}}(n,\lambda,\mu) = C_{S_{Best}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ Nest_{2,1} = C_{S_{2,1}}(n,\lambda,\mu) = C_{S_{2,1}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ Nest_{k_1,1} = C_{S_{k_1,1}}(n,\lambda,\mu) = C_{S_{k_1,1}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \end{cases} \\ Habitat_2 = \begin{cases} Nest_{Best} = C_{S_{Best}}(n,\lambda,\mu) = C_{S_{Best}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ Nest_{2,2} = C_{S_{2,2}}(n,\lambda,\mu) = C_{S_{2,2}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ Nest_{k_2,2} = C_{S_{k_2,2}}(n,\lambda,\mu) = C_{S_{k_2,2}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \end{cases} \\ \vdots \qquad\qquad\qquad\qquad \vdots \\ Habitat_H = \begin{cases} Nest_{Best} = C_{S_{Best}}(n,\lambda,\mu) = C_{S_{Best}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ Nest_{2,H} = C_{S_{2,H}}(n,\lambda,\mu) = C_{S_{2,H}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \\ \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ Nest_{k_H,H} = C_{S_{k_2,H}}(n,\lambda,\mu) = C_{S_{k_2,H}}(n_1,n_2,...,n_m,\lambda_1,\lambda_2,...,\lambda_m,\mu_1,\mu_2,...,\mu_m) \end{cases} \end{cases} \tag{8}$$

$$k_1, k_2, ..., k_H \in \{2,3,4,...,K\}$$

Step 6: Steps 2 to 5 are repeated for a fixed number of iterations; then, the minimum system cost with the optimal values are displayed.

The pseudo-code of the developed EF-COA for the cost minimization of repairable systems subject to availability constraint is presented in Algorithm 1, and Figure 1 shows its flowchart.

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

Input the parameters: H, K, Ω, Φ_j, N_{Iter}.

Generate random number of nests for each habitat according to Eq. (4).

While $G \leq N_{Iter}$

    Evaluate the system cost (each nest) according to Eq. (5).

    Constraint handling using Eq. (6).

    Identify the best solution (minimum system cost) and migration (save this best solution) according to Eq. (7).

7: Use the saved solution to create new habitats and nests according to Eq. (8).

9: End while

10: Display the minimum system cost and the optimal values.

Insert: Figure 1 – Flowchart of the implemented EF-COA.

4. Case studies

4.1. Parallel-series system

Insert: Figure 2 – Parallel-series system.

The overall system cost of five subsystems connected in parallel-series configuration (see Figure 2) is given by the mathematical model[14]:

$$\text{Minimize} \quad C_s(n,\lambda,\mu) = \sum_{i=1}^{5}\left[\left(\alpha_i\left(\lambda_i\right)^{-\beta_i} + \mu_i mc_i\right)\left(n_i + \exp(n_i/4)\right)\right] \tag{9}$$

Subject to

$$\sum_{i=1}^{5} p_i(n_i)^2 \le 150 \tag{10}$$

$$\sum_{i=1}^{5} w_i n_i \exp(n_i/4) \le 200 \tag{11}$$

$$\prod_{i=1}^{5}\left[1-\left(1-\frac{\mu_i}{\lambda_i+\mu_i}\right)^{n_i}\right] \ge 0.9 \tag{12}$$

$$n_i \ge 1; n_i \in \mathbf{Z}^+$$
$$\lambda_i \in \left[10^{-7}, 10^{-3}\right] \subset \Re^+$$
$$\mu_i \in \left[32\times10^{-7}, 32\times10^{-3}\right] \subset \Re^+$$
$$i = 1, 2, \ldots, 5$$

where Eq. (10) is the system design configuration constraint of weight, Eq. (11) is the system design configuration constraint of the product of weight and volume, and Eq. (12) is the system availability requirement constraint. The above problem involves five integer variables and ten real variables: Table 1 reports the relevant data.

Insert: Table 1 – Data used in parallel-series and n-stage standby systems.

4.2. *n*-stage standby system

Insert: Figure 3 – n-stage standby system.

The *n*-stage standby system considered includes five subsystems[15] (see Figure 3), and the corresponding optimization reads:

$$\text{Minimize} \quad C_S(n,\lambda,\mu)=\sum_{i=1}^{5}\left[\left(\alpha_i\left(\lambda_i\right)^{-\beta_i}+\mu_i mc_i\right)\left(n_i+\exp(n_i/4)\right)\right] \tag{13}$$

Subject to

$$\sum_{i=1}^{5} p_i(n_i)^2 \le 150 \tag{14}$$

$$\sum_{i=1}^{5} w_i n_i \exp(n_i/4) \le 200 \tag{15}$$

$$\prod_{i=1}^{5}\left[1-\left(\sum_{k=0}^{n_i}\left(\frac{\lambda_i}{\mu_i}\right)^{k}\right)^{-1}\left(\frac{\lambda_i}{\mu_i}\right)^{n_i}\right] \ge 0.9 \tag{16}$$

$$n_i \ge 1; n_i \in \mathbf{Z}^+$$
$$\lambda_i \in \left[10^{-7},10^{-3}\right] \subset \Re^+$$
$$\mu_i \in \left[32\times10^{-7},32\times10^{-3}\right] \subset \Re^+$$

$$i=1,2,\ldots,5$$

The data and the constraints (14) and (15) are the same as for the parallel-series system. However, the system availability requirement constraint formulated in Eq. (16) is more complex than that in Eq. (12).

5. Results and discussion

The developed EF-COA has been coded using the MATLAB programming language and run on a personal computer with a Processor G620 (2.60 GHz Sandy Bridge, 4 GB of RAM and 3 Mo of cache memory) under the Windows 7 - 64bits operating system. The number of habitats and the maximum number of nests per habitat is 10. The number of iterations is fixed at 50, i.e. the maximum number of function evaluations that the algorithm may use is 5000. The base COA has been also applied to compare the results.

Tables 2 and 3 report the results for the two case studies. The best values of the system cost and number of function evaluations (NFE) are highlighted in bold. In Table 2, the cost obtained by the EF-COA for the parallel-series system is 214.1934 (in arbitrary cost units), which is smaller than that of COA (214.2662) and TA–GA (214.7794).[14] The EF-COA also used the lowest NFE (5,000) compared to the other methods, 30,000 and 40,000, respectively. From Table 3, it can be observed that the cost provided by the EF-COA (234.9172) for the *n*-stage standby system is less than the result of the TA–GA (236.8314)[15] and the COA (236.2035). The NFE performance is also better, as for the parallel-series system. Furthermore, the standard deviations (SD) of twenty independent runs reveal that the EF-COA is more stable than the COA, i.e. smaller SD.

Figures 4–6 highlight the performances of the proposed EF-COA for the parallel-series system and the *n*-stage standby system, respectively.

Insert: Table 2 – Results for the parallel-series system.

Insert: Table 3 – Results for the n-stage standby system.

Insert: Figure 4 – System cost for the parallel-series system.

Insert: Figure 5 – System cost for the n-stage standby system.

Insert: Figure 6 – NFE for the parallel-series system and the n-stage standby system.

6. Conclusions

In this paper, a new solution approach for minimizing the system cost of repairable systems subject to availability constraints has been proposed. A novel method based on the habitats and floating nests of the cuckoo, called the efficient cuckoo optimization algorithm (EF-COA), has been developed. The standard COA uses the Egg Laying Radius (ELR), which may slow down the algorithm when solving a complex problem. In the EF-COA, various habitats consisting of different nests are implemented in order to improve the quality on the solution and the performances. Therefore, the ELR has been avoided, while the system cost has been modeled as a nest. Application to two numerical case studies, i.e. parallel-series system and *n*-stage standby system has demonstrated the effectiveness of the proposed method in terms of better solutions and fewer function evaluations. Future research efforts will be devoted to extending the method for treating and addressing multi-objective optimization problems and a comprehensive industrial case study.

Appendix

Step 1: Generate initial cuckoo habitat.

The initial set of solutions represents the cuckoo habitat:

$$\text{Habitat} = \left[x_1, x_2, ..., x_{N_{var}} \right] \tag{17}$$

where X is the vector of the solutions and N_{var} is the number of variables in the problem. A matrix of size $N_{pop} \times N_{var}$ is generated.

Step 2: Evaluation.

The fitness of each line is evaluated, where N_{pop} is the number of lines. The habitat is evaluated as a fitness function.

Step 3: Egg allocation.

Dedicate some eggs to each cuckoo.

Step 4: Egg laying radius.

The cuckoos start to lay eggs in the area according to a distance called egg laying radius (ELR):

$$ELR = \alpha \times \frac{Number\ of\ current\ cuckoo's\ eggs}{Total\ number\ of\ eggs} \times (var_{hi} - var_{low}) \quad (18)$$

where α is an integer, var_{hi} and var_{low} are the upper and lower bounds of the variables. Some eggs will be destroyed by the host birds and 10% of the survival cuckoos will starve.

Step 5: Migration.

When the cuckoos become mature, the cuckoos' swarm will migrate to achieve the best goal. The different groups are classified using the *K*–means clustering. Each cuckoo fly *U%* of all the way toward destination (where *U* is a random number uniformly distributed between 0 and 1), with a deviation φ ($\Pi/6$ rad).

Step 6: Population limit.

A maximum number of cuckoos is considered to limit the population.

Step 7: Repeat steps 2–6 until the stopping condition is satisfied.

References

1. Marseguerra M, Zio E, Martorell S (2006) Basics of genetic algorithms optimization for RAMS applications. Reliab Eng Syst Saf 91:977–991
2. Marseguerra M, Zio E, Podofillini L (2004) Optimal reliability/availability of uncertain systems via multi-objective genetic algorithms. IEEE Trans Reliab 53:424–434
3. Lin YH, Li YF, Zio E (2015) Fuzzy Reliability Assessment of Systems with Multiple-Dependent Competing Degradation Processes. IEEE Trans Fuzzy Syst 23:1428–1438. https://doi.org/10.1109/TFUZZ.2014.2362145
4. Zio E (2009) Reliability engineering: Old problems and new challenges. Reliab Eng Syst Saf 94:125–141
5. Giuggioli Busacca P, Marseguerra M, Zio E (2001) Multiobjective optimization by genetic algorithms: application to safety systems. Reliab Eng Syst Saf 72:59–74

6. Garg H, Rani M, Sharma SP (2013) An efficient two phase approach for solving reliability-redundancy allocation problem using artificial bee colony technique. Comput Oper Res 40:2961–2969. https://doi.org/10.1016/j.cor.2013.07.014

7. Hsieh YC, You PS (2011) An effective immune based two-phase approach for the optimal reliability-redundancy allocation problem. Appl Math Comput 218:1297–1307. https://doi.org/10.1016/j.amc.2011.06.012

8. Liu Y, Qin G (2015) A DE algorithm combined with Lévy flight for reliability redundancy allocation problems. Int J Hybrid Inf Technol 8:113–118

9. Garg H (2015) An efficient biogeography based optimization algorithm for solving reliability optimization problems. Swarm Evol Comput 24:1–10. https://doi.org/10.1016/j.swevo.2015.05.001

10. Zhang E, Chen Q (2016) Multi-objective reliability redundancy allocation in an interval environment using particle swarm optimization. Reliab Eng Syst Saf 145:83–92. https://doi.org/10.1016/j.ress.2015.09.008

11. Lins ID, Moura MC, Zio E, Droguett EL (2012) A particle swarm-optimized support vector machine for reliability prediction. Qual Reliab Eng Int 28:141–158

12. Mellal MA, Zio E (2016) A penalty guided stochastic fractal search approach for system reliability optimization. Reliab Eng Syst Saf 152:213–227

13. Kumar A, Pant S, Ram M (2016) System Reliability Optimization Using Gray Wolf Optimizer Algorithm. Qual. Reliab. Eng. Int.

14. Liu GS (2012) Availability optimization for repairable parallel-series system by applying Tabu-GA combination method. In: 10th IEEE International Conference on Industrial Informatics. Beijin, China, pp 803–808

15. Liu GS (2013) Availability optimization for repairable n-stage standby system by applying Tabu-GA combination method. Int J Model Optim 3:245–250

16. Shelokar PS, Jayaraman VK, Kulkarni BD (2002) Ant algorithm for single and multiobjective reliability optimization problems. Qual Reliab Eng Int 18:497–514. https://doi.org/10.1002/qre.499

17. Mellal MA, Zio E (2017) System reliability-redundancy allocation by evolutionary computation. In: 2nd International Conference on System Reliability and Safety. IEEE, Milan, Italy

18. Cha JW, Finkelstein M (2018) On preventive maintenance under different assumptions on the failure/repair processes. Qual Reliab Eng Int 34:66–77

19. Yu J, Zheng S, Pham H, Chen T (2018) Reliability modeling of multi-state degraded repairable systems and its applications to automotive systems. Qual Reliab Eng Int. https://doi.org/10.1002/qre.2265

20. Rajabioun R (2011) Cuckoo optimization algorithm. Appl Soft Comput 11:5508–5518

21. Mellal MA, Adjerid S, Williams EJ, Benazzouz D (2012) Optimal replacement policy

for obsolete components using cuckoo optimization algorithm based-approach: Dependability context. J Sci Ind Res (India) 71:715–721

22. Mellal MA, Adjerid S, Williams EJ (2013) Optimal selection of obsolete tools in manufacturing systems using cuckoo optimization algorithm. Chem Eng Trans 33:355–360. https://doi.org/10.3303/CET1333060
23. Amiri E, Mahmoudi S (2016) Efficient protocol for data clustering by fuzzy cuckoo optimization algorithm. Appl Soft Comput 41:15–21
24. Mellal MA, Williams EJ (2016) Parameter optimization of advanced machining processes using cuckoo optimization algorithm and hoopoe heuristic. J Intell Manuf 27:927–942
25. Mellal MA, Williams EJ (2015) Cuckoo optimization algorithm for unit production cost in multi-pass turning operations. Int J Adv Manuf Technol 76:647–656. https://doi.org/10.1007/s00170-014-6309-2
26. Mellal MA, Williams EJ (2016) Total production time minimization of a multi-pass milling process via cuckoo optimization algorithm. Int J Adv Manuf Technol 87:747–754. https://doi.org/10.1007/s00170-016-8498-3
27. Rabiee M, Sajedi H (2013) Job scheduling in grid computing with cuckoo optimization algorithm. Int J Comput Appl 62:38–44
28. Roozitalab A, Asgharizadeh E (2013) Optimizing the warranty period by cuckoo meta-heuristic algorithm in heterogeneous customers' population. J Ind Eng Int 9:1–6
29. Mellal MA, Williams EJ (2015) Cuckoo optimization algorithm with penalty function for combined heat and power economic dispatch problem. Energy 93:1711–1718. https://doi.org/10.1016/j.energy.2015.10.006
30. Khormali A, Addeh J (2016) A novel approach for recognition of control chart patterns: Type-2 fuzzy clustering optimized support vector machine. ISA Trans 63:256–264. https://doi.org/10.1016/j.isatra.2016.03.004
31. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186:311–338. https://doi.org/10.1016/S0045-7825(99)00389-8
32. Subbaraj P, Rengaraj R, Salivahanan S (2009) Enhancement of combined heat and power economic dispatch using self adaptive real-coded genetic algorithm. Appl Energy 86:915–921. https://doi.org/10.1016/j.apenergy.2008.10.002

AUTHOR BIOGRAPHIES

Mohamed Arezki Mellal has a Ph.D. in mechatronics. He is an Associate Professor (with accreditation to supervise research) at the Department of Mechanical Engineering, Faculty of

Technology, M'Hamed Bougara University, Algeria and a Visiting Scholar at the Center for Advanced Life Cycle Engineering, Department of Mechanical Engineering, University of Maryland, College Park, MD, USA. Likewise, he was a Visiting Scholar at Osaka Electro-Communication University – Japan, Korea Aerospace University – South Korea, and University of Essex – UK. He is a member of the Algerian National Laboratory for Maintenance Education in conjunction with the European Union (Erasmus+). He has published in several journal and conference proceedings. He has edited three books and authored four book chapters. Dr. Mellal was an Invited Keynote Lecturer at the plenary sessions of the Day of Science 2015, Boumerdes, Algeria and of the International Symposium on Technology & Sustainable Industry Development, ISTSID 2019, El-Oued, Algeria. He was a Session Chair for the 2nd and 3rd International Conference on System Reliability and Safety, ICSRS 2017 & 2018, Milan (Italy) and Barcelona (Spain), respectively. He has been selected as a Best Reviewer for the International Conference on Vision, Image and Signal Processing, ICVISP 2017 & 2018, Osaka (Japan) and Las Vegas (USA), respectively. He has also been a committee member for over seventy international conferences. He serves as a regular reviewer for eighteen SCI-indexed journals and an editorial board member in seven peer-reviewed international journal.

Enrico Zio received the M.Sc. degree in nuclear engineering from the Politecnico di Milano, Milan, Italy, in 1991, the M.Sc. degree in mechanical engineering from the University of California at Los Angeles, Los Angeles, CA, USA, in 1995, the Ph.D. degree in nuclear engineering from the Politecnico di Milano, in 1996, and the Ph.D. degree in probabilistic risk assessment from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1998. He is currently a Full Professor with the Centre for Research on Risk and Crises, Ecole de Mines, ParisTech, PSL University, Sophia Antipolis, France, a Full Professor and the President of the Alumni Association at Politecnico di Milano, an Eminent Scholar with Kyung Hee University, Seoul, South Korea, a Distinguished Guest Professor with Tsinghua University, Beijing, China, an Adjunct Professor with the City University of Hong Kong, Hong Kong, Beihang University, Beijing, and Wuhan University, Wuhan, China, and the Co-Director of the Center for Reliability and Safety of Critical Infrastructures and the SinoFrench

Laboratory of Risk Science and Engineering, Beihang University. He has authored or co-authored over 7 books and 300 papers on international journals. His current research interests include modeling of the failure– repair–maintenance behavior of components and complex systems, analysis of their reliability, maintainability, prognostics, safety, vulnerability, resilience, and security characteristics, and development and use of the Monte Carlo simulation methods, artificial techniques, and optimization heuristics. Dr. Zio is the Chairman and Co-Chairman of several international conferences, an Associate Editor of several international journals, and a referee of more than 20.

Edward J. Williams holds bachelor's and master's degrees in Mathematics (Michigan State University, 1967; University of Wisconsin, 1968). From 1969 to 1971, he did statistical programming and analysis of biomedical data at Walter Reed Army Hospital, Washington, D.C. He joined Ford Motor Company in 1972, where he worked until retirement in December 2001 as a computer software analyst supporting statistical and simulation software. After retirement from Ford, he joined PMC, Dearborn, Michigan, as a senior simulation analyst. Also, since 1980, he has taught classes at the University of Michigan, including both undergraduate and graduate simulation classes using GPSS/H™, SLAM II™, SIMAN™, ProModel®, SIMUL8®, or Arena®. He is a member of the Institute of Industrial Engineers [IIE], the Society for Computer Simulation International [SCS], and the Michigan Simulation Users Group [MSUG]. He serves on the editorial board of the International Journal of Industrial Engineering – Applications and Practice. During the last several years, he has given invited plenary addresses on simulation and statistics at conferences in Monterrey, México; İstanbul, Turkey; Genova, Italy; Rīga, Latvia; and Jyväskylä, Finland. He served as a co-editor of Proceedings of the International Workshop on Harbour, Maritime and Multimodal Logistics Modelling & Simulation 2003, a conference held in Rīga, Latvia. Likewise, he served the Summer Computer Simulation Conferences of 2004, 2005, and 2006 as Proceedings co-editor. He was the Simulation Applications track coordinator for the 2011 Winter Simulation Conference. A paper he co-authored with three of his simulation students won "best paper in track" award at the Fifth International Conference on Industrial Engineering and Operations Management, held in Dubai, United Arab Emirates, in March

2015.

List of figures

List of tables

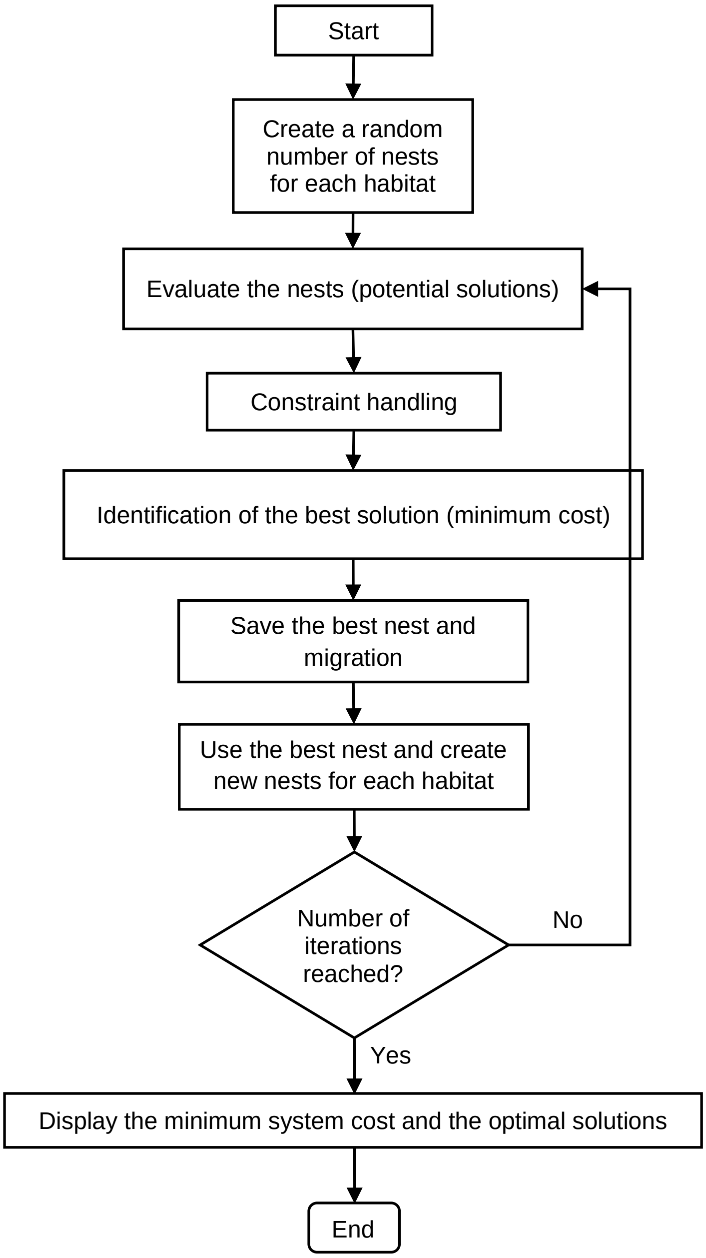


Figure 1 – Flowchart of the implemented EF-COA.

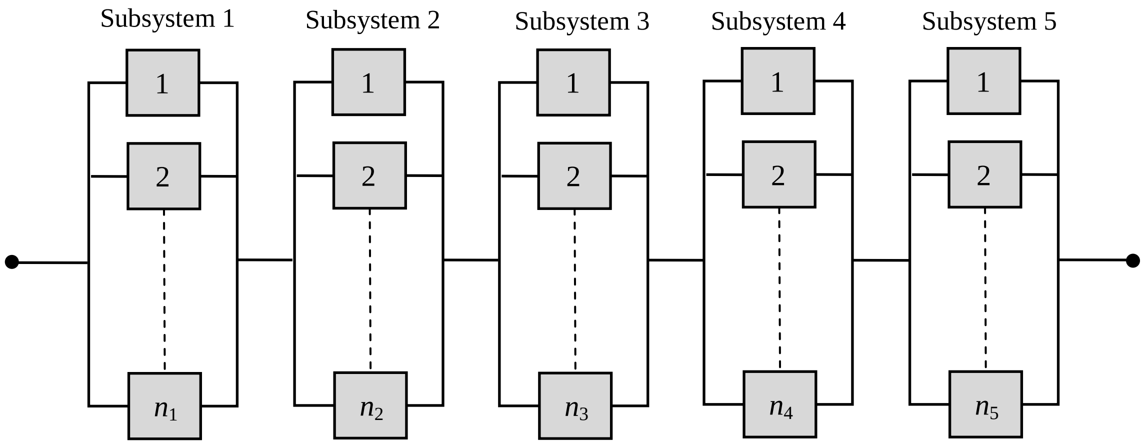


Figure 2 – Parallel-series system.

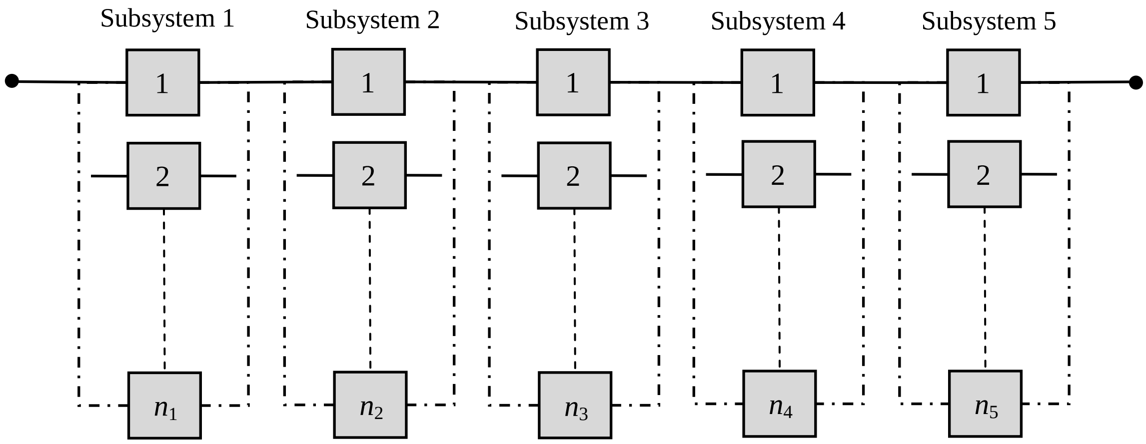
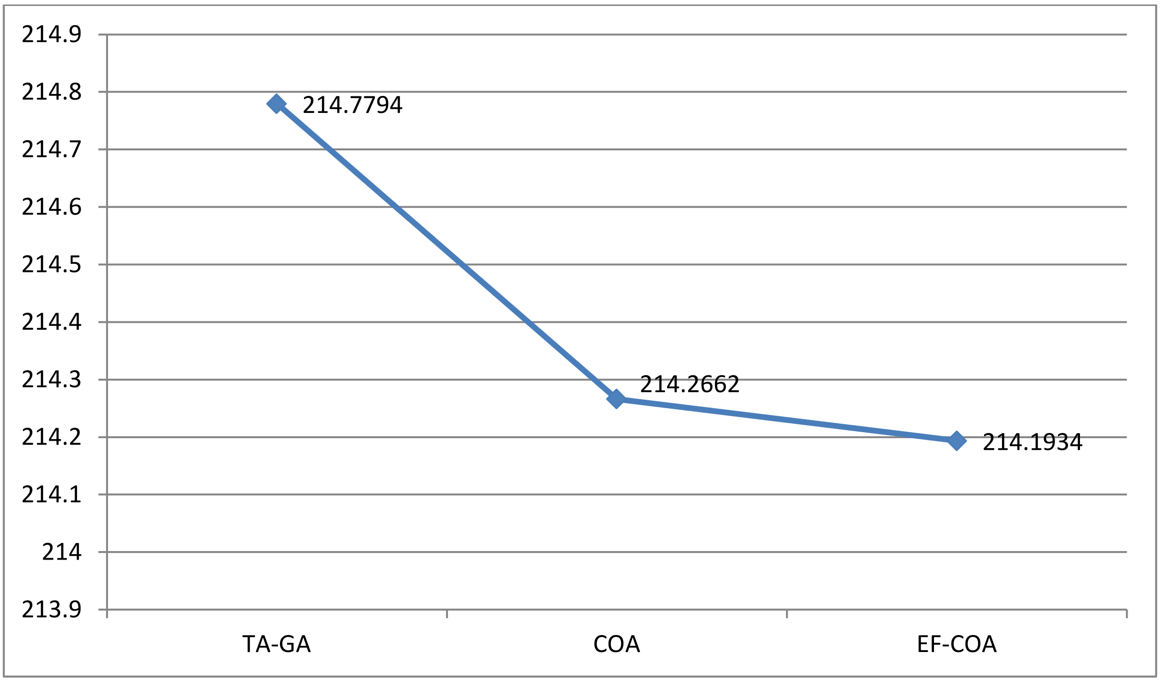
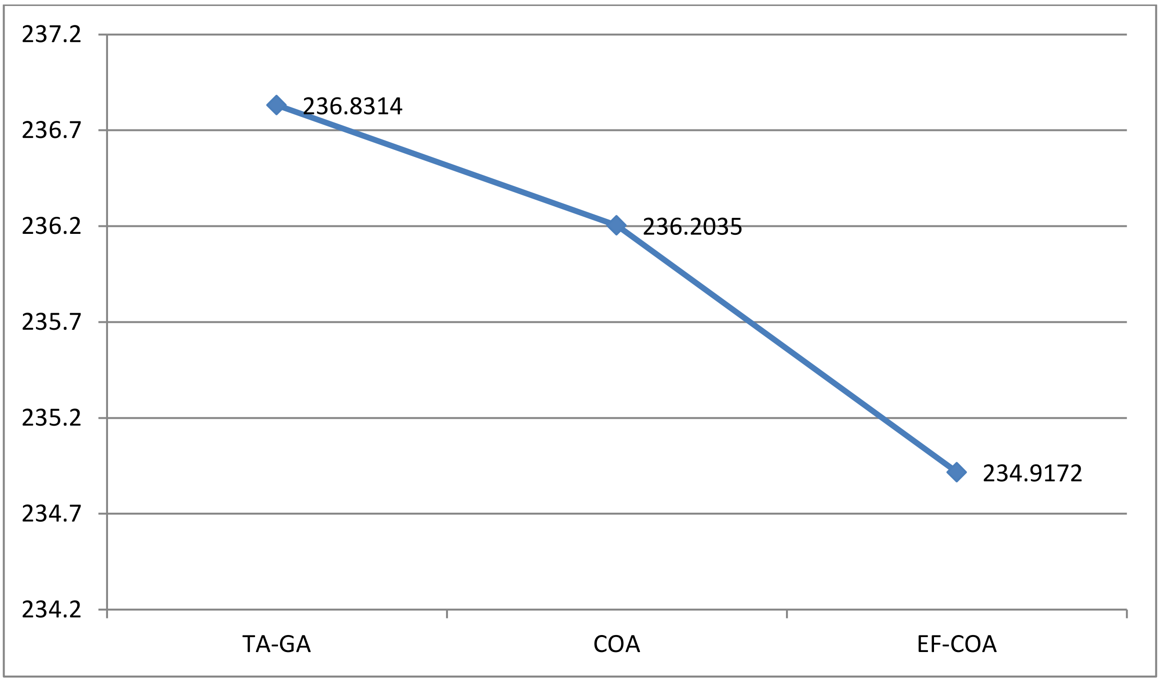


Figure 3 – *n*-stage standby system.



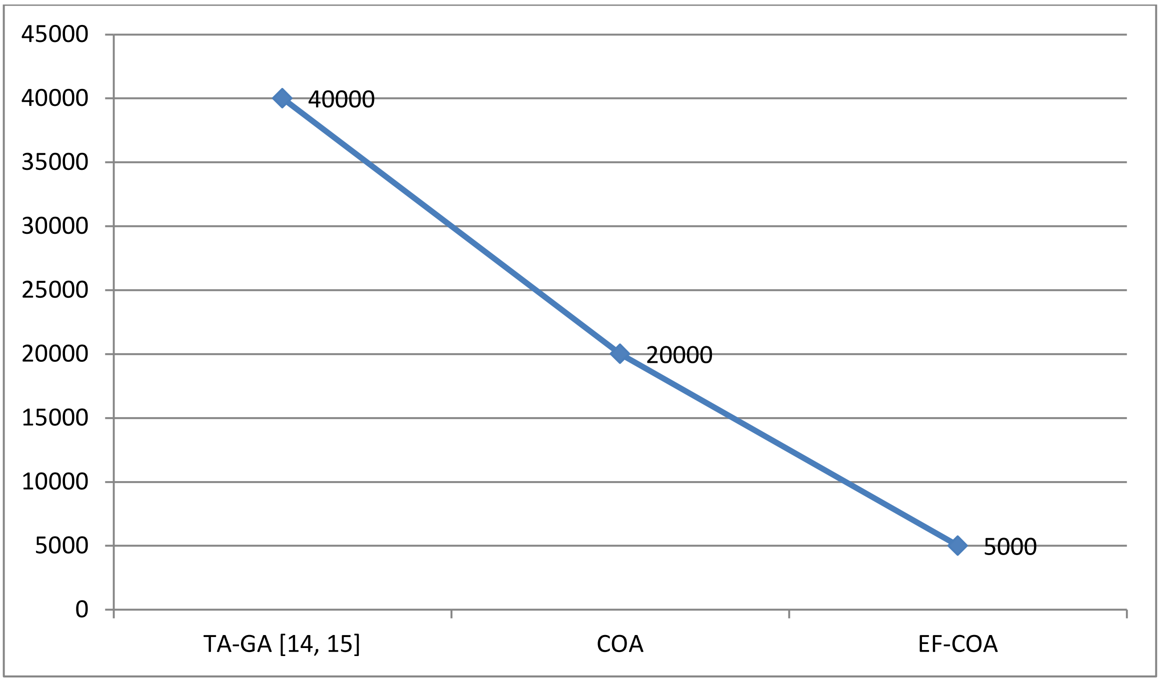
TA-GA[14]

Figure 4 – System cost for the parallel-series system.



TA-GA[15]

Figure 5 – System cost for the *n*-stage standby system.



TA-GA[14, 15]

Figure 6 – NFE for the parallel-series system and the n-stage standby system.

Table 1 – Data used in parallel-series and n-stage standby systems.

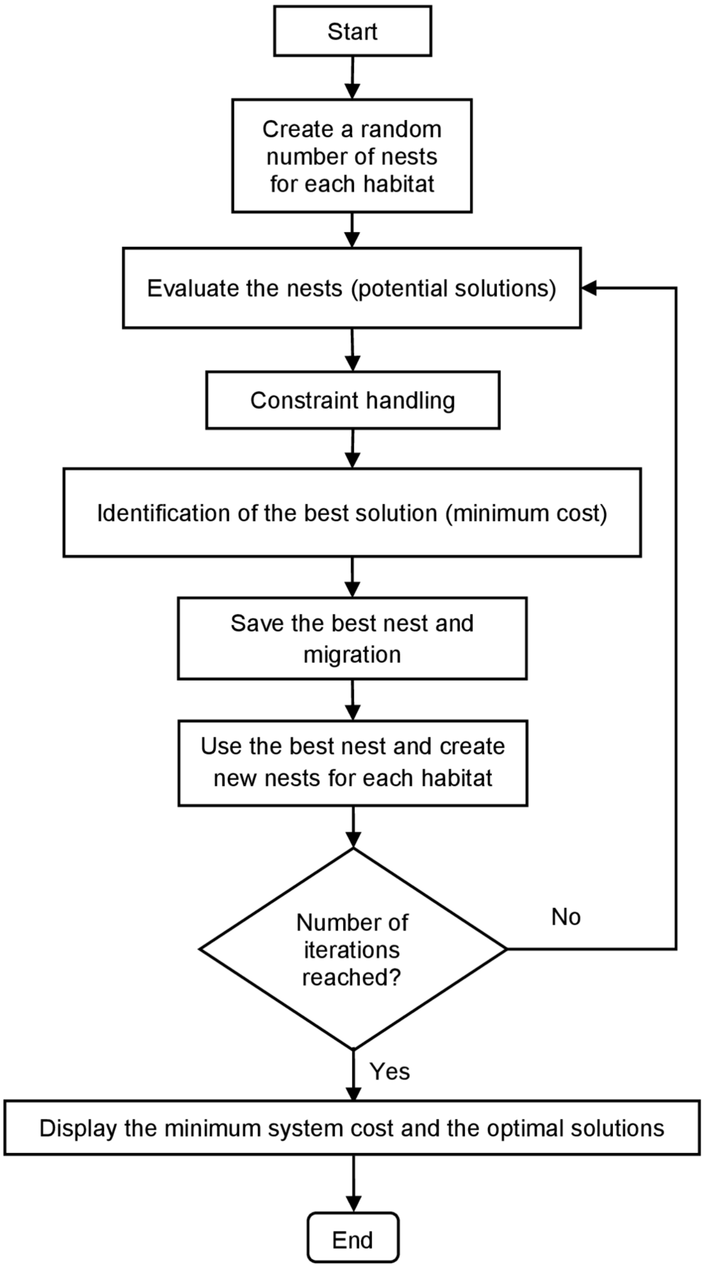
Subsystem i	α_i (10^{-5})	β_i	mc_i	p_i	w_i
1	2.33	1.5	5000	1	7
2	1.45	1.5	5000	2	8
3	0.541	1.5	5000	3	8
4	8.05	1.5	5000	4	6
5	1.95	1.5	5000	2	9

Table 2 – Results for the parallel-series system.

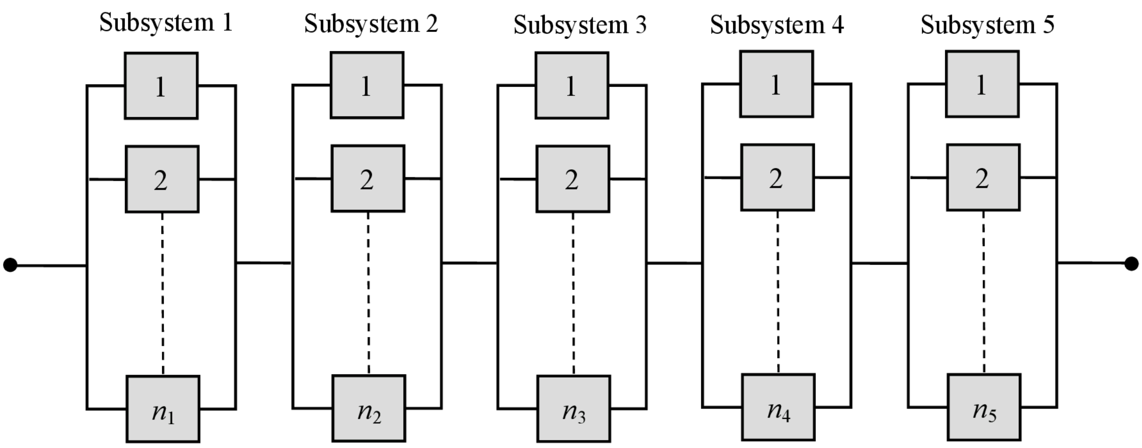
Method	n	λ (10^{-3})	μ (10^{-2})	A	A_s	C_s	NFE	SD
TA–GA[14]	(3, 2, 2, 3, 3)	(0.3584, 0.2236, 0.1447, 0.6568, 0.3401)	(0.10, 0.14, 0.09, 0.15, 0.09)	(0.9830, 0.9802, 0.9812, 0.9726, 0.9789)	0.9000	214.7794	40,000	–
COA	(3, 2, 2, 3, 3)	(0.3640, 0.2307, 0.1466, 0.6469, 0.3419)	(0.1018, 0.1317, 0.0999, 0.1449, 0.0965)	(0.9817, 0.9777, 0.9836, 0.9705, 0.9820)	0.9000	214.2662	30,000	6.32E–03
EF-COA (Proposed approach)	(3, 2, 2, 3, 3)	(0.3636, 0.2249, 0.1425, 0.6494, 0.3360)	(0.1004, 0.1283, 0.0972, 0.1463, 0.0952)	(0.9812, 0.9777, 0.9836, 0.9709, 0.9822)	0.9000	**214.1934**	**5,000**	**4.29E–05**

Table 3 – Results for the *n*-stage standby system.

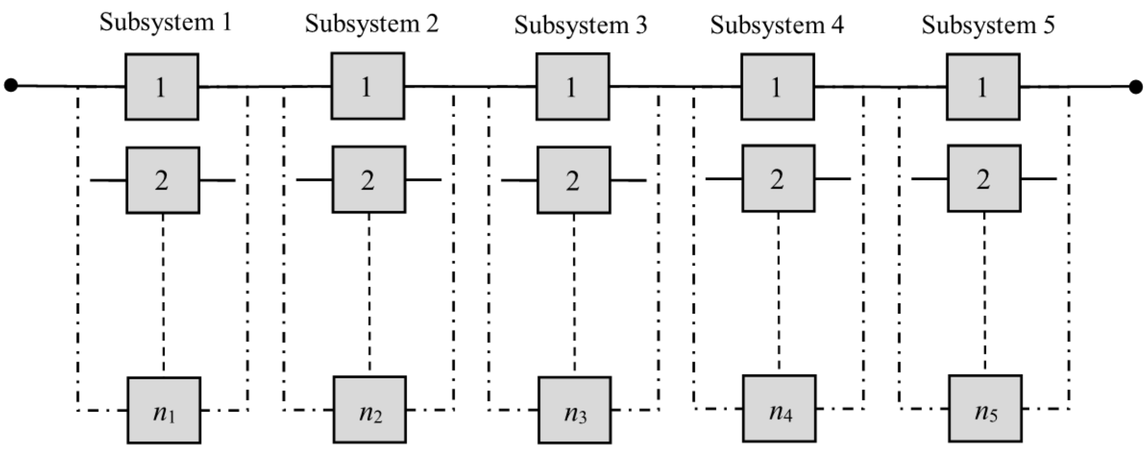
Method	n	λ (10^{-3})	μ (10^{-2})	A	A_s	C_s	NFE	SD
TA–GA[15]	(3, 3, 2, 3, 2)	(0.3261, 0.2749, 0.1507, 0.5963, 0.2555)	(0.12, 0.10, 0.10, 0.17, 0.15)	(0.9852, 0.9838, 0.9808, 0.9710, 0.9750)	0.9000	236.8314	40,000	–
COA	(2, 3, 2, 3, 3)	(0.2737, 0.2712, 0.1382, 0.5901, 0.3106)	(0.1548, 0.0970, 0.0998, 0.1685, 0.1068)	(0.9741, 0.9841, 0.9834, 0.9716, 0.9824)	0.9000	236.2035	30,000	5.17E–02
EF-COA (Proposed approach)	(3, 3, 3, 2, 2)	(0.3329, 0.2694, 0.1653, 0.4893, 0.2486)	(0.1133, 0.0972, 0.0672, 0.2288, 0.1502)	(0.9819, 0.9845, 0.9887, 0.9636, 0.9770)	0.9000	**234.9172**	**5,000**	**7.04E–04**



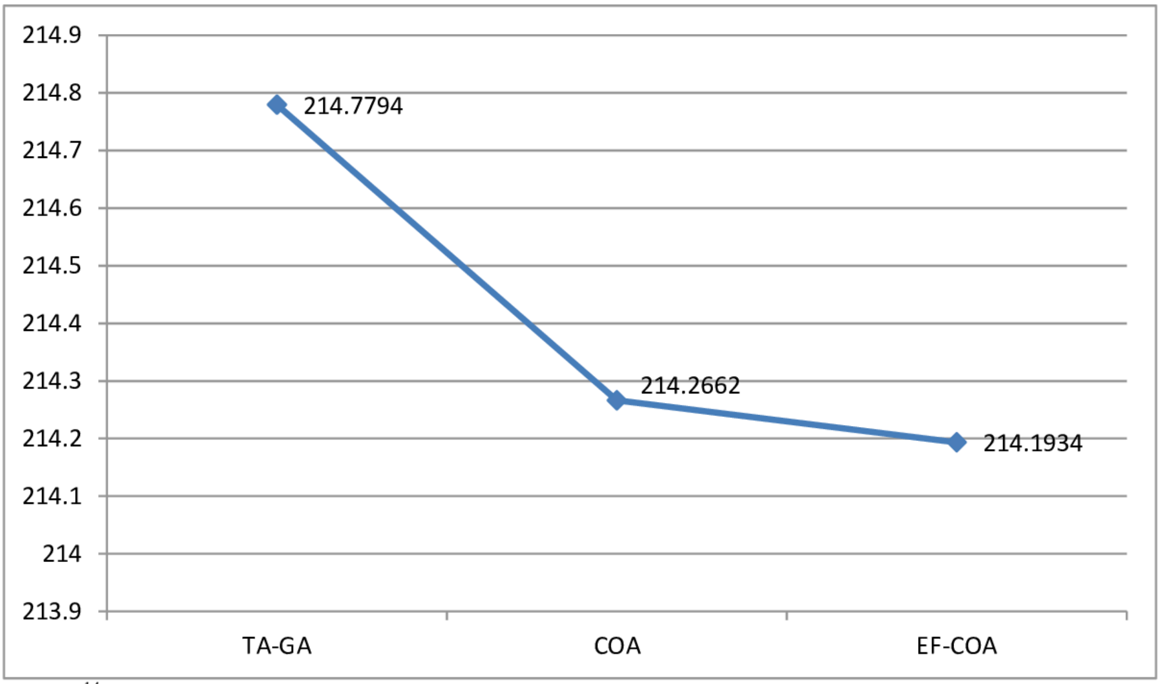
QRE_2617_F1.tiff



QRE_2617_F2.tiff

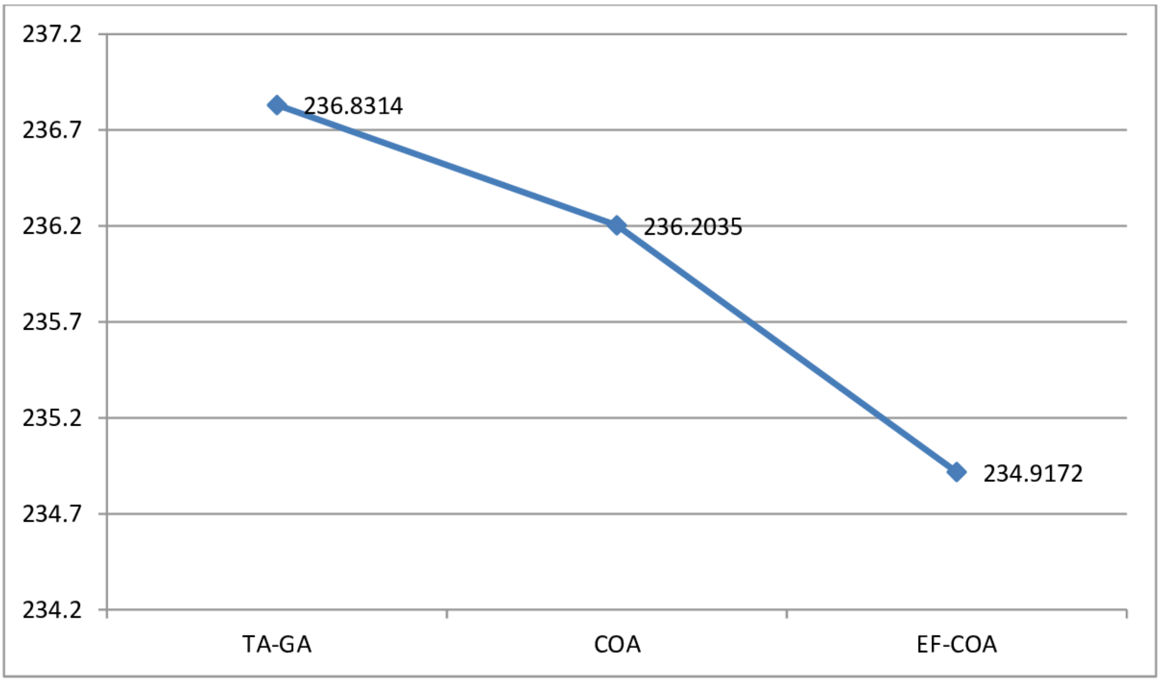


QRE_2617_F3.tiff



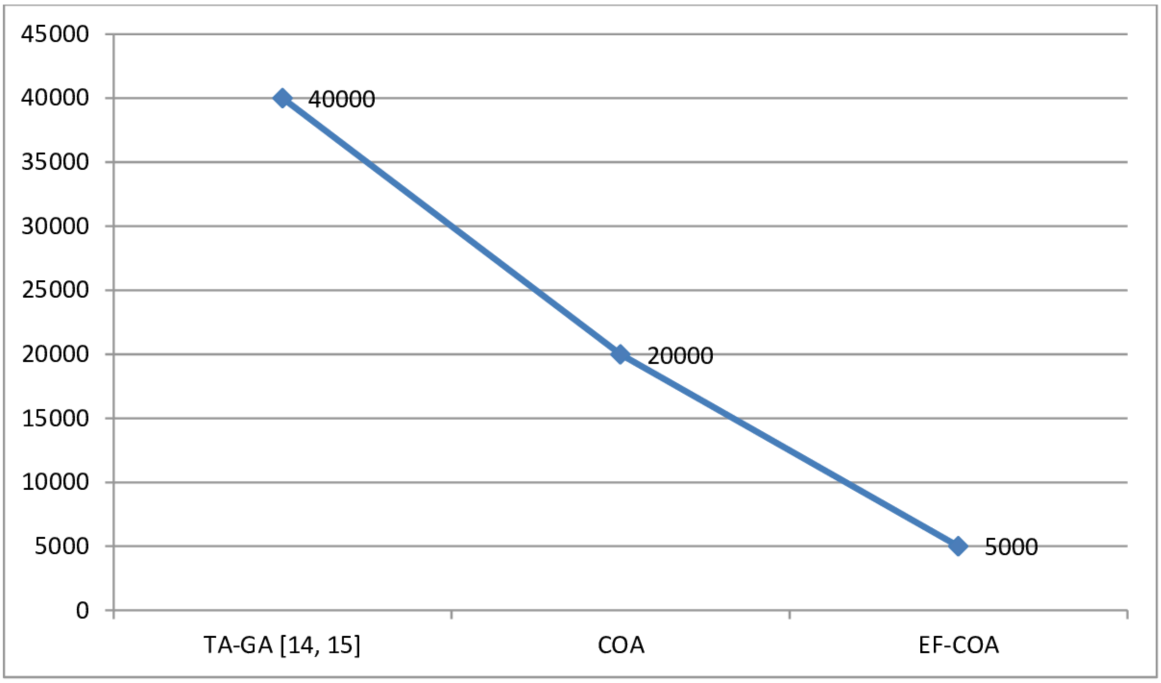
TA-GA[14]

QRE_2617_F4.tiff



TA-GA[15]

QRE_2617_F5.tiff



TA-GA[14, 15]

QRE_2617_F6.tiff