An adaptive particle swarm optimization method for multi-objective system reliability optimization

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]*Chair System Science and the Energy Challenge, Fondation Electricité de France (EDF), CentraleSupélec,Université Paris-Saclay, Grande Voie des Vignes, 92290 Chatenay-Malabry, France*

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

Abstract

Multi-objective system reliability optimization has attracted the attention of several researchers, due to its importance in industry. In practice, the optimization regards multiple objectives, e.g., maximize the reliability, minimize the cost, weight and volume. In this paper, an adaptive particle swarm optimization (ADAP-PSO) is presented for multi-objective system reliability optimization. The approach uses a Lévy flight for some particles of the swarm, for avoiding local optima and insuring diversity in the exploration of the search space. The multi-objective problem is converted to a single-objective problem by resorting to the weighted-sum method and a penalty function is implemented to handle the constraints. Nine numerical case studies are presented as benchmark problems for comparison; the results show that the proposed approach has superior performance than a standard PSO.

Keywords: Multi-objective optimization; Reliability-redundancy optimization; Adaptive particle swarm optimization (ADAP-PSO).

Notations

R_s	system reliability.
Q_s	$=1-R_s$, system unreliability (failure probability).

m	number of subsystems in the system.
r_i	reliability of each component in subsystem i, $1 \le i \le m$.
r	$=(r_1, r_2, \ldots, r_m)$, vector of component reliabilities for the system.
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, $1 \le i \le m$.
$n_{i,\max}$	maximum number of components in subsystem i, $1 \le i \le m$.
M	number of constraints.
g_j	jth constraint function, $j=1, \ldots, M$.
$C(r_i)$	$=\alpha_i(-T/\ln r_i)^{\beta_i}$, cost of each component with reliability r_i in subsystem i, $1 \le i \le m$.
T	operating time during which the component must not fail (mission time).
w_i	weight of each component in subsystem i, $1 \le i \le m$.
v_i	volume of each component in subsystem i, $1 \le i \le m$.
β_i, α_i	parameters (shaping and scaling factors, respectively) of each component at subsystem i, $1 \le i \le m$.
W, C	upper limits on the weight and cost of the system, respectively.
V	upper limit on the sum of the subsystems' products of volume and weight.
W_s, C_s	weight and cost of the system, respectively.
V_s,	sum of the subsystems' products of volume and weight.
$\exp(n_i/4)$	term accounting the interconnecting hardware.

1. Introduction

System reliability optimization aims at having in each subsystem, an optimal number of allocated redundant components, or allocated reliabilities, or both. The goal is to maximize the overall system reliability, subject to the limits of cost, weight and volume [1]. The problem may be single-objective or multi-objective (i.e., only one objective is considered or multiple objectives, under design constraints), according to the goal of the application.

The redundancy allocation problem involves integer variables and the reliability allocation problem involves real variables; then, the reliability-redundancy allocation problem is mixed. Various methods have been proposed for solving reliability optimization problems (ROPs). These methods may be divided into two main categories [1,2]: mathematical programming (exact, such as Branch & Bound [3], dynamic programming [4], surrogate constraint [5],

branch-and-cut algorithm [6] and approximation, such as linear programming [7], mixed integer linear programming [8]), heuristic [9] and metaheuristic methods, such as genetic algorithms [10–13], artificial bee colony [14], particle swarm optimization [15], cuckoo search [16], and penalty-guided stochastic fractal search [17]. The genetic algorithms have been successfully implemented for multi-objective RAMS (reliability, availability, maintainability and safety) [11] and condition-based maintenance [18] optimization problems. In particular, the metaheuristics, often called artificial intelligence methods or intelligent computation, have proven their effectiveness in solving reliability optimization problems for realistic systems and are currently considered the most promising solution methods. They are able to find optimal solutions in reasonable CPU time [19,20]. Zhang et al. [21] combined the bare-bones particle swarm optimization with the sensitivity-based clustering (BBMOPSO). The system reliability, cost and weighted have been generated in the Pareto front. Fang et al. [22] used the nondominated sorting binary differential evolution (NSBDE) algorithm to optimize the system resilience and cost. Kong et al. [23] used a simplified version of particle swarm optimization (SPSO) to optimize the system reliability with multiple strategy choices. Abouei [24] implemented a modified version of the GA (MVGA) by considering the standby strategy in the optimization of the system reliability. Recently, the system availability of a parallel-series system has been optimizing by resorting to the genetic algorithm (GA), differential evolution (DE), particle swarm optimization (PSO), cuckoo optimization algorithm (COA), and flower pollination algorithm (FPA) [25]. In [26,27], the authors used the nondominated sorting genetic algorithm II (NSGA-II) to optimize the system reliability and cost as both objectives.

Dealing with both objectives of design is a great challenge for the decision maker. On the other hand, most of the previous approaches are limited and need to be more efficient in the search for optimal solutions. The goal of this paper is to propose a novel solution approach

based on an adaptive particle swarm optimization (ADAP-PSO), for solving the multi-objective system reliability optimization problem. The multi-objective problem is converted to a single-objective by using the weighted-sum method [28]. The remainder of the paper is organized as follows: Section 2 presents the multi-objective system reliability optimization problem and mentions various techniques for its solution. Section 3 describes the adaptive particle swarm optimization implemented. Nine numerical case studies of systems with various subsystems' configurations are presented in Section 4. Results and discussion are given in Section 5. Finally, the last Section concludes this paper with some remarks.

2. Multi-objective system reliability optimization

The general mathematical multi-objective system reliability optimization is written as follows:

$$\begin{aligned} &\text{Maximize} \quad R_s(r,n) = R_s(r_1, r_2, ..., r_m;\ n_1, n_2, ..., n_m) \\ &\text{Minimize} \quad V_s(r,n) = V_s(r_1, r_2, ..., r_m;\ n_1, n_2, ..., n_m) \\ &\text{Minimize} \quad C_s(r,n) = C_s(r_1, r_2, ..., r_m;\ n_1, n_2, ..., n_m) \\ &\text{Minimize} \quad W_s(r,n) = W_s(r_1, r_2, ..., r_m;\ n_1, n_2, ..., n_m) \end{aligned} \tag{1}$$

subject to

$$R_s(r,n) \geq R_{\min} \tag{2}$$

$$V_s(r,n) \leq V_{\max} \tag{3}$$

$$C_s(r,n) \leq C_{\max} \tag{4}$$

$$W_s(r,n) \leq W_{\max} \tag{5}$$

$$\begin{gathered} 0.5 \leq r_i \leq 1;\ 1 \leq n_i \leq n_{i,\max},\ \ i = 1, 2, ..., m; \\ r_i \in [0.5, 1] \subset \mathbb{R}^+;\ n_i \in \mathbb{Z}^+ \end{gathered} \tag{6}$$

where $R_s(\bullet)$ is the overall system reliability, $R_{\min}$ is its minimum allowable value, $V_s(\bullet)$ is the sum of the subsystems' products of volume and weight, $V_{\max}$ is its upper limit, $C_s(\bullet)$ is the system cost, $C_{\max}$ is its upper limit, $W_s(\bullet)$ is the system weight and $W_{\max}$ is its upper limit; r_i

and n_i are the reliability and the number of redundant components in the ith subsystem, respectively, and m is the number of subsystems in the system.

The methods for solving the multi-objective optimization problems are diverse and can be classified in three main categories:

- Weighted-sum method by assigning weights to each function in order to convert the multi-objective problem to a single-objective problem [28].
- Optimize one objective and include the other objectives as constraints [29].
- Generate and analyse the set of Pareto optimal solutions [11,30].

In this paper, the weighted-sum method is implemented to solve the multi-objective system reliability optimization problem. The objective functions of Eq.(1) are converted to a single-objective function as follows:

$$\text{Minimize} \quad f(r,n) = \delta_1 Q_s + \delta_2 V_s + \delta_3 C_s + \delta_4 W_s \tag{7}$$

where Q_s is the unreliability (failure probability) of the system (Q_s=1−R_s) and δ=(δ_1, δ_2, δ_3, δ_4) is the weight vector, such that $\sum_{a=1}^{4} \delta_a = 1$. The values of δ_i depend on the targets of the decision maker.

3. Adaptive particle swarm optimization

The particle swarm optimization (PSO) has been proposed by Kennedy and Eberhart [31]. It is inspired by the movement mode of swarms in nature, such as birds and fishes. The comprehensive concepts of PSO can be referred to [31,32,20,33]. In this paper, some particles fly using a Lévy flight for diversification and finding the best solutions by exploring the search space of the multi-objective system reliability optimization problem. The objectives are normalized in a single-objective by resorting to the weighted-sum method described in Eq.(7). Furthermore, a penalty function is used to handle the constraints.

The fitness value of each particle is evaluated after handling the constraints by adding penalty terms to Eq.(7):

$$\text{Fitness value} = f(r,n) + \psi(r,n) \tag{8}$$

where $\psi(r_1, r_2,\ldots, r_m, n_1, n_2, \ldots, n_m)$ is the penalty function.

The penalty function is computed as follows [17,34]:

$$\begin{aligned}\psi(r_1,r_2,...,r_m,n_1,n_2,...,n_m) = \phi_1 \cdot \max(0, R_s(r,n))^2 + \phi_2 \cdot \max(0, V_s(n))^2 \\ + \phi_3 \cdot \max(0, C_s(r,n))^2 + \phi_4 \cdot \max(0, W_s(n))^2\end{aligned} \tag{9}$$

where ϕ_l (l=1,…,4) is the penalty factor. If the operand is negative, then, the absolute value is taken; otherwise, it is set to zero. The vector of the variables for the redundancy numbers is rounded to the nearest positive integer value before evaluating the fitness.

In practice, some particles in a swarm may fly a bit farther than the majority of particles and their position could be more interesting to explore for finding food and habitat. These particles, then, perform a random walk, using the Lévy flight, as done in the cuckoo search (CS) algorithm in the successive search iterations for finding new solutions to the optimization problem [35–37]. Some authors have tried to combine the principles of PSO and CS for solving optimization problems. In [38], each particle has a limit value of the solution and in case that the particle is unable to improve this solution, then, this value is increased; on the contrary, if this limit has been exceeded, then, the particle performs a Lévy flight into the search space. In [39], the velocity of the particles is updated using the Lévy flight. In [40], the Lévy flight has been applied to the inertia coefficient of PSO. In [41], the Lévy flight has been applied to a percentage of particles except the global best particle, which uses a polynomial mutation.

In the current proposed algorithm, at each iteration, L particles move using a Lévy flight:

$$x_z^{t+1} = x_z^t + \gamma \text{Lévy}(\lambda) \tag{10}$$

where x_z^{t+1} is the new solution, z is the index of the particle travelling according to a Lévy flight (z=1,...,L), γ is the step size scaling factor (it is advised to keep γ=1), and Lévy(λ) is the Lévy distribution.

The approach proposed is called an Adaptive Particle Swarm Optimization (ADAP-PSO) and the baseline is:

Step 1: Input the parameters.

Step 2: Initialize particles.

Step 3: Evaluate the fitness of each particle of the swarm and handle the constraints.

Step 4: If the fitness value is better than the best fitness value (*pbest*) thus far, then, set the current value as the new *pbest*.

Step 5: Select the particle with the best fitness value of all the particles as the *gbest*.

Step 6: Move (M–L) particles using the following equation:

$$v_j^{t+1} = \theta v_j^t + C_1\varepsilon_1(pbest_j^t - x_j^t) + C_2\varepsilon_2(gbest^t - x_j^t) \tag{11}$$

where M is the number of particles in the swarm, L is the number of particles flying using a Lévy flight model, j is the index of the particle in the swarm (j=1,…,M–L), v_j^{t+1} is the velocity at the (t+1)th iteration, θ is the inertia weight (we use θ=0.5 as a medium value), C_1 and C_2 are acceleration constants (it is advised to keep C_1=C_2=2), ε_1 and ε_2 are random numbers uniformly distributed in [0,1], and x_j^t is the position of the particle.

Step 7: Move L particles with a Lévy flight.

Step 8: Update the position of the particles.

The position of the (M–L) particles is updated using the following equation:

$$x_j^{t+1} = x_j^t + v_j^{t+1} \tag{12}$$

Step 9: Repeat **Steps 3** to **8** until the number of iterations is reached; then, display the optimal solutions.

The pseudo-code of the ADAP-PSO implemented is described in Algorithm 1 and its flowchart is represented in Figure 1, where M is the number of particles in the swarm, L is the is the number of particles flying using a Lévy flight model, N_{Iter} is the number of iterations. ϕ (1,…,4) is the vector of the penalty factor and δ=(δ_1, δ_2, δ_3, δ_4) is the weight vector.

Algorithm 1 – Pseudo-code of the implemented ADAP-PSO.

1: Input the parameters: M, L, N_{Iter}, ϕ_1, ϕ_2, ϕ_3, ϕ_4, δ_1, δ_2, δ_3, δ_4.
2: Initialize particles.
3: While $K \leq N_{Iter}$
4: Evaluate the fitness according to Eq.(8).
5: Update *pbest*.
6: Update *gbest*.
7: Move (M–L) particles using Eq.(11).
8: Move L particles using Eq.(10).
9: Update position of the (M–L) particles using Eq.(12).
10: End While
11: Display the results.

Insert: Figure 1 – Flowchart of the implemented ADAP-PSO.

The ADAP-PSO starts with random particles and the loop begins until the number of iterations has been reached. The loop evaluates the fitness value using Eq. (8) and both positions are updated (*pbest* and *gbest*). (M–L) and L particles are moved by resorting to Eqs. (10) and (11), respectively. At this stage, the overall solution is improved by updating the position of the (M–L) particles using Eq. (12).

4. Numerical case studies

In this section, we present nine case studies for different system configurations.

4.1. Series system

Insert: Figure 2 – Series system.

Insert: Table 1 – Data used in series system and complex -bridge- system.

The multi-objective optimization problem of the series system represented in Figure 1 [5,17] is written as follows:

$$\begin{aligned}
&\text{Maximize} \quad R_s(r,n) = \prod_{i=1}^{5}\left[1-(1-r_i)^{n_i}\right] \\
&\text{Minimize} \quad V_s(n) = \sum_{i=1}^{5} v_i n_i^2 \\
&\text{Minimize} \quad C_s(r,n) = \sum_{i=1}^{5} \alpha_i (-T/\ln r_i)^{\beta_i}\left[n_i + \exp(n_i/4)\right] \\
&\text{Minimize} \quad W_s(n) = \sum_{i=1}^{5} w_i n_i \exp(n_i/4)
\end{aligned} \tag{13}$$

subject to

$$R_s(r,n) \geq R_{\min} \tag{14}$$

$$V_s(n) \leq V_{\max} \tag{15}$$

$$C_s(r,n) \leq C_{\max} \tag{16}$$

$$W_s(n) \leq W_{\max} \tag{17}$$

$$0.5 \leq r_i \leq 1,\ r_i \in [0.5,1] \subset \mathbb{R}^+;\ \ 1 \leq n_i \leq 5,\ n_i \in \mathbb{Z}^+;\ \ i = 1,2,...,5$$

The minimum allowable system reliability is 0.90 ($R_{\min}$=0.9). The data are reported in Table 1.

4.2. Series-parallel system

Insert: Figure 3 – Series–parallel system.

Insert: Table 2 – Data used in series-parallel system.

The series-parallel here considered (see Table 2 and Figure 3 [14]) is written as follows:

$$
\begin{aligned}
&\text{Maximize } R_s(r,n) = 1-\left(1-R_1R_2\right)\left[1-\left(R_3+R_4-R_3R_4\right)R_5\right] \\
&\text{Minimize } V_s(n) = \sum_{i=1}^{5} v_i n_i^2 \\
&\text{Minimize } C_s(r,n) = \sum_{i=1}^{5} \alpha_i (-T/\ln r_i)^{\beta_i} \left[n_i + \exp(n_i/4)\right] \\
&\text{Minimize } W_s(n) = \sum_{i=1}^{5} w_i n_i \exp(n_i/4)
\end{aligned}
\tag{18}
$$

subject to

$$R_s(r,n) \geq R_{\min} \tag{19}$$

$$V_s(n) \leq V_{\max} \tag{20}$$

$$C_s(r,n) \leq C_{\max} \tag{21}$$

$$W_s(n) \leq W_{\max} \tag{22}$$

$$0.5 \leq r_i \leq 1,\ r_i \in [0.5,1] \subset \mathbb{R}^+;\ \ 1 \leq n_i \leq 5,\ n_i \in \mathbb{Z}^+;\ \ i = 1,2,...,5$$

where $R_{\min}$=0.9.

4.3. Complex (bridge) system

Insert: Figure 4 – Complex (bridge) system.

Figure 4 shows the configuration of the complex (bridge) system [14]; the mathematical definition of the related multi-objective optimization problem is given as follows:

$$
\begin{aligned}
&\text{Maximize } R_s(r,n) = R_5\left(1-Q_1Q_3\right)\left(1-Q_2Q_4\right) + Q_5\left[1-\left(1-R_1R_2\right)\left(1-R_3R_4\right)\right] \\
&\text{Minimize } V_s(n) = \sum_{i=1}^{5} v_i n_i^2 \\
&\text{Minimize } C_s(r,n) = \sum_{i=1}^{5} \alpha_i (-T/\ln r_i)^{\beta_i} \left[n_i + \exp(n_i/4)\right] \\
&\text{Minimize } W_s(n) = \sum_{i=1}^{5} w_i n_i \exp(n_i/4)
\end{aligned}
\tag{23}
$$

subject to

$$R_s(r,n) \geq R_{\min} \tag{24}$$

$$V_s(n) \leq V_{\max} \tag{25}$$

$$C_s(r,n) \leq C_{\max} \tag{26}$$

$$W_s(n) \le W_{\max} \tag{27}$$

$$0.5 \le r_i \le 1,\ r_i \in [0.5,1] \subset \mathbb{R}^+;\ \ 1 \le n_i \le 5,\ n_i \in \mathbb{Z}^+;\ \ i = 1,2,...,5$$

where $R_{\min}$=0.9.

4.4. Overspeed protection system

Insert: Figure 5 – Overspeed protection system.

Insert: Table 3 – Data used in overspeed protection system.

The details of the overspeed protection system (see Figure 4 and Table 3) are given in [17]. The multi-objective optimization problem of this system can be written as follows:

$$\begin{aligned}
&\text{Maximize } \ R_s(r,n) = \prod_{i=1}^{4}\left[1-(1-r_i)^{n_i}\right] \\
&\text{Minimize } \ V_s(n) = \sum_{i=1}^{4} v_i n_i^2 \\
&\text{Minimize } \ C_s(r,n) = \sum_{i=1}^{4} \alpha_i (-T/\ln r_i)^{\beta_i}\left[n_i + \exp(n_i/4)\right] \\
&\text{Minimize } \ W_s(n) = \sum_{i=1}^{4} w_i n_i \exp(n_i/4)
\end{aligned} \tag{28}$$

subject to

$$R_s(r,n) \ge R_{\min} \tag{29}$$

$$V_s(n) \le V_{\max} \tag{30}$$

$$C_s(r,n) \le C_{\max} \tag{31}$$

$$W_s(n) \le W_{\max} \tag{32}$$

$$0.5 \le r_i \le 1,\ r_i \in [0.5,1] \subset \mathbb{R}^+;\ \ 1 \le n_i \le 10,\ n_i \in \mathbb{Z}^+;\ \ i = 1,2,...,4$$

where $R_{\min}$=0.9999.

4.5. Complex bridge network system

Insert: Figure 6 – Complex bridge network system.

Figure 6 shows the complex bridge network system [17]. The multi-objective problem can be written as follows:

$$\begin{aligned}&\text{Maximize}\quad R_s(r_1,r_2,r_3,r_4,r_5)=r_1r_4+r_2r_5+r_2r_3r_4+r_1r_3r_5+2r_1r_2r_3r_4r_5\\&\qquad -r_2r_3r_4r_5-r_1r_3r_4r_5-r_1r_2r_4r_5-r_1r_2r_3r_5-r_1r_2r_3r_4\\&\text{Minimize}\quad C_s(r_1,r_2,r_3,r_4,r_5)=\sum_{i=1}^{5}a_i\exp\left(\frac{b_i}{1-r_i}\right)\end{aligned}\tag{33}$$

subject to

$$0.5\le r_i\le 1,\ \text{for } i=1,2,3,4,5 \tag{34}$$

$$0.99\le R_s\le 1 \tag{35}$$

where $a_i=1,\ b_i=0.0003,\ \forall i$.

4.6. Life-support system in a space capsule

The multi-objective optimization problem of the life-support system in a space capsule considered here gives rise to two cases, derived from the single-objective problem solved in [42] (Case 1) and in [17,42] (Case 2), respectively.

Case 1:

$$\begin{aligned}&\text{Maximize}\quad R_s(r_1,r_2,r_3,r_4)=1-r_3\left[\left(1-r_1\right)\left(1-r_4\right)\right]^2-\left(1-r_3\right)\left[1-r_2\left(1-\left(1-r_1\right)\left(1-r_4\right)\right)\right]^2\\&\text{Minimize}\quad C_s(r_1,r_2,r_3,r_4)=2\sum_{i=1}^{4}K_ir_i^{\alpha_i}\end{aligned}\tag{36}$$

subject to

$$0.5\le r_i\le 1,\ \text{for } i=1,2,3,4 \tag{37}$$

$$0.9\le R_s(r,n)\le 1 \tag{38}$$

where K_1=K_2=100, K_3=200, K_4=150, and α_i=0.6 $\forall i$.

Case 2:

$$\begin{aligned}&\text{Maximize}\quad R_s(r_1,r_2,r_3,r_4)=1-r_3\left[\left(1-r_1\right)\left(1-r_4\right)\right]^2-\left(1-r_3\right)\left[1-r_2\left(1-\left(1-r_1\right)\left(1-r_4\right)\right)\right]^2\\&\text{Minimize}\quad C_s(r_1,r_2,r_3,r_4)=\sum_{i=1}^{4}K_i\left[\tan\left(\frac{\pi}{2}r_i\right)\right]^{\alpha_i}\end{aligned}\tag{39}$$

$$0.5\le r_i\le 1,\ \text{for } i=1,2,3,4 \tag{40}$$

$$0.99\le R_s(r)\le 1 \tag{41}$$

where K_1=K_2=25, K_3=50, K_4=37.5, and , and α_i=1 $\forall i$.

4.7. Large scale reliability-redundancy allocation problem

Insert: Table 4 – Data used in large scale reliability-redundancy allocation problem.

Table 4 reports the data for the large scale reliability-redundancy allocation problem [17,43]; the multi-objective problem can be written as follows:

$$\begin{aligned}
&\text{Maximize } R_s(r,n) = \prod_{i=1}^{20}\left[1-(1-r_i)^{n_i}\right] \\
&\text{Minimize } V_s(n) = \sum_{i=1}^{20} v_i n_i^2 \\
&\text{Minimize } C_s(r,n) = \sum_{i=1}^{20} \alpha_i (-T/\ln r_i)^{\beta_i}\left[n_i + \exp(n_i/4)\right] \\
&\text{Minimize } W_s(n) = \sum_{i=1}^{20} w_i n_i \exp(n_i/4)
\end{aligned} \tag{42}$$

subject to

$$R_s(r,n) \geq R_{\min} \tag{43}$$

$$V_s(n) \leq V_{\max} \tag{44}$$

$$C_s(r,n) \leq C_{\max} \tag{45}$$

$$W_s(n) \leq W_{\max} \tag{46}$$

$$0.5 \leq r_i \leq 1,\ r_i \in [0.5,1] \subset \mathbb{R}^+;\ \ 1 \leq n_i \leq 10,\ n_i \in \mathbb{Z}^+;\ \ i = 1,2,...,20$$

where $R_{\min}$=0.88.

4.8. Pharmaceutical plant

Insert: Figure 7 – Pharmaceutical plant.

Insert: Table 5 – Data used in pharmaceutical plant.

The single-objective optimization problem of the pharmaceutical plant (see Figure 7 and Table 5) has been proposed in [44] and solved in [17]. The multi-objective problem of this system can be written as follows:

$$\begin{aligned}
&\text{Maximize } R_s(r,n) = \prod_{i=1}^{10}\left[1-(1-r_i)^{n_i}\right] \\
&\text{Minimize } V_s(n) = \sum_{i=1}^{5} v_i n_i^2 \\
&\text{Minimize } C_s(r,n) = \sum_{i=1}^{5} \alpha_i (-T/\ln r_i)^{\beta_i}\left[n_i + \exp(n_i/4)\right] \\
&\text{Minimize } W_s(n) = \sum_{i=1}^{5} w_i n_i \exp(n_i/4)
\end{aligned} \tag{47}$$

subject to

$$R_s(r,n) \geq R_{\min} \tag{48}$$

$$V_s(n) \leq V_{\max} \tag{49}$$

$$C_s(r,n) \leq C_{\max} \tag{50}$$

$$W_s(n) \leq W_{\max} \tag{51}$$

$$0.5 \leq r_i \leq 1-10^{-6},\ r_i \in \left[0,1-10^{-6}\right] \subset \mathbb{R}^+;\ \ 1 \leq n_i \leq 5,\ n_i \in \mathbb{Z}^+;\ \ i = 1,2,...,5$$

where $R_{\min}$=0.95.

5. Results and discussion

The developed ADAP-PSO has been implemented using Matlab 2015 and run on a PC with the following specifics: Intel Pentium Processor G620, Sandy Bridge, 2.60GHz, 4GB of RAM, 3Mo Cache, Windows 7 - 64 bits. A PSO has been run for the same problems, on the same computer. The number of particles in the swarm is 20 (M=20) for each algorithm and the number of particles flying according to a Lévy flight in the ADAP-PSO is 5 (L=5). For the series system, series-parallel system, complex bridge system, overspeed protection system, large scale reliability-redundancy allocation problem, and the pharmaceutical plant, the

weights are equally fixed to 0.25 (δ_a=0.25, a=1,…,4) for R_s, V_s, C_s and W_s, respectively. For the complex bridge network and life-support system in a space capsule (cases 1 and 2), the value of δ is 0.5 (δ_1=δ_2=0.5) for both R_s and C_s. We consider that there is no preference between the objectives, i.e., the weights are equal for each objective. However, the decision maker may change these values, according to her/his preferences on the targets. The values of the cost, weight and volume are assumed in arbitrary units.

Tables 6–14 report the results of PSO and ADAP-PSO for the numerical case studies illustrated in Section 4. The optimal solutions, the number of function evaluations (NFE) needed by each algorithm for converging to the optimal solutions, and the CPU time required are compared. The bold represents the best values.

Insert: Table 6 – Results for the series system.

Insert: Table 7 – Results for the series-parallel system.

From Table 6, it can be observed that the ADAP-PSO provides better results (R_s=0.91; V_s=73; W_s=164.99906) than the PSO (R_s=0.90924; V_s=88; W_s=189.42752). Also, the ADAP-PSO uses fewer function evaluations (600) for shorter CPU time (10.2 s) than the PSO (1000 and 25.8 s, respectively). Thus, the ADAP-PSO has outperformed the PSO in solving the multi-objective series system, despite that the PSO achieves a smaller system cost C_s. The results reported in Table 7 for the series-parallel system also reveal that the computations of the ADAP-PSO are better than those of the PSO, as discussed for the series system.

Insert: Table 8 – Results for the complex (bridge) system.

In Table 8, the optimal results of the ADAP-PSO for the complex (bridge) system are: R_s=0.90011, V_s=15, C_s=49.14271 and W_s=62.88688, whereas those of the PSO are: R_s=0.90011, V_s=21, C_s=32.83563 and W_s=78.99422. Therefore, it can be observed that the

values of V_s and W_s provided by the ADAP-PSO are better. Furthermore, the NFE and CPU time of the ADAP-PSO are smaller, i.e., (700, 13.1 s) vs. (1240, 32 s).

Insert: Table 9 – Results for the overspeed protection system.

Insert: Table 10 – Results for the complex bridge network system.

Insert: Table 11 – Results for the life-support system in a space capsule (Case 1).

Insert: Table 12 – Results for the life-support system in a space capsule (Case 2).

Tables 9 and 10 show that the values of C_s, NFE, and CPU time for the ADAP-PSO are smaller than those of the PSO for the overspeed protection system and the complex bridge network system, respectively. In Table 11, the results of the ADAP-PSO for the life-support system in a space capsule (Case 1) are: R_s=0.90111, C_s=642.34313, NFE=480 and CPU time=8.4 s, whereas those of the PSO are: R_s=0.9, C_s=647.78277, NFE=1080 and CPU time=24.6 s. It can be observed that the ADAP-PSO has again outperformed the PSO. For Case 2 (see Table 12), the ADAP-PSO has also outperformed the PSO, despite that the value R_s=0.99 is the same.

Insert: Table 13 – Results for the large scale reliability-redundancy allocation problem.

Insert: Table 14 – Results for the pharmaceutical plant.

In Table 13, the results for the large scale reliability-redundancy allocation problem, consisting of 40 mixed real-integer decision variables, show again that the ADAP-PSO is better than the PSO. The same is seen in Table 14 for the pharmaceutical plant. On the other hand, it can be observed that the standard deviations (SD) of the ADAP-PSO are smaller than those of the PSO. It reveals that the ADAP-PSO is more stable when solving the current problem. Finally, Figure 8 clearly shows that the PDP-PSO required the fewer number of function evaluations.

6. Conclusions

In this paper, we have proposed a novel solution method for the multi-objective system reliability optimization problem. An adaptive particle swarm optimization (ADAP-PSO) has been developed, based on the Lévy flight of some particles in the swarm at each search iteration for efficiently exploring the search space, ensuring the diversity, and avoiding local optima. The multi-objective problem has been converted to a single-objective problem by the weighted sum method and the constraints have been handled by a penalty function.

Nine single-objective numerical case studies from the literature have been considered and solved with the proposed ADAP-PSO and the standard PSO. The results have been compared and the ADAP-PSO has outperformed the simple PSO in terms of fitness values, number of function evaluations and CPU time. In our future works we intend to investigate complex industrial problems to show the practical value of the approach.

References

[1] Kuo W, Prasad V, Tillman F, Hwang C. Optimal reliability design: fundamentals and applications. New York: University Press; n.d.

[2] Soltani R. Reliability optimization of binary state non-repairable systems: A state of the art survey. Int J Ind Eng Comput 2014;5:339–364.

[3] Djerdjour M, Rekab K. A branch and bound algorithm for designing reliable systems at a minimum cost. Appl Math Comput 2001;118:247–59. doi:10.1016/S0096-3003(99)00217-9.

[4] Kulshrestha DK, Gupta MC. Use of dynamic programming fo reliability engineers. IEEE Trans Reliab 1973;R-22:240–241.

[5] Hikita M, Nakagawa Y, Nakashima K, Narihisa H. Reliability optimization of systems by a surrogate-constraints algorithm. IEEE Trans Reliab 1992;41:413–480.

[6] Caserta M, Voß S. An exact algorithm for the reliability redundancy allocation problem. Eur. J. Oper. Res., vol. 244, 2015, p. 110–6. doi:10.1016/j.ejor.2015.01.008.

[7] Kolesar PJ. Linear programming and the reliability of multi-component systems. Nav Res Logist Q 1967;15:317–327.

[8] Ramirez-Marquez JE, Coit DW, Konak A. Redundancy allocation for series-parallel systems using a max-min approach. IIE Trans 2004;36:891–898.

[9] Aggarwal KK. Redundancy Optimization in General Systems. Reliab IEEE Trans 1976;R-25:330–2. doi:10.1109/TR.1976.5220030.

[10] Hsieh Y-C, Chen T-C, Bricker DL. Genetic algorithms for reliability design problems. Microelectron Reliab 1998;38:1599–605. doi:10.1016/S0026-2714(98)00028-6.

[11] Marseguerra M, Zio E, Martorell S. Basics of genetic algorithms optimization for RAMS applications. Reliab Eng Syst Saf 2006;91:977–991.

[12] Mellal MA, Zio E. System reliability-redundancy allocation by evolutionary computation. 2nd Int. Conf. Syst. Reliab. Saf., Milan, Italy: IEEE; 2017. doi:10.1109/ICSRS.2017.8272790.

[13] 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.

[14] Yeh W-C, Hsieh T-J. Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Comput Oper Res 2011;38:1465–73. doi:10.1016/j.cor.2010.10.028.

[15] 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.

[16] Garg H. An approach for solving constrained reliability-redundancy allocation problems using cuckoo search algorithm. Beni-Suef Univ J Basic Appl Sci 2015;4:14–25. doi:10.1016/j.bjbas.2015.02.003.

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

[18] Marseguerra M, Zio E, Podofillini L. Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation. Reliab Eng Syst Saf 2002;77:151–65. doi:10.1016/S0951-8320(02)00043-1.

[19] Mellal MA, Williams EJ. A survey on ant colony optimization, particle swarm optimization, and cuckoo algorithms. Handb. Res. Emergent Appl. Optim. Algorithms. IGI Global, USA: 2018.

[20] Talbi EG. Population-based metaheuristics. Metaheuristics From Des. to Implement. John Wiley, New Jersey: 2009, p. 190–307.

[21] Zhang E, Wu Y, Chen Q. A practical approach for solving multi-objective reliability

redundancy allocation problems using extended bare-bones particle swarm optimization. Reliab Eng Syst Saf 2014;127:65–76. doi:10.1016/j.ress.2014.03.006.

[22] Fang Y, Pedroni N, Zio E. Optimization of Cascade-Resilient Electrical Infrastructures and its Validation by Power Flow Modeling. Risk Anal 2015;35:594–607. doi:10.1111/risa.12396.

[23] Kong X, Gao L, Ouyang H, Li S. Solving the redundancy allocation problem with multiple strategy choices using a new simplified particle swarm optimization. Reliab Eng Syst Saf 2015;144:147–58. doi:10.1016/j.ress.2015.07.019.

[24] Abouei Ardakan M, Sima M, Zeinal Hamadani A, Coit DW. A novel strategy for redundant components in reliability--redundancy allocation problems. IIE Trans (Institute Ind Eng 2016;48:1043–57. doi:10.1080/0740817X.2016.1189631.

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

[26] Muhuri PK, Ashraf Z, Lohani QMD. Multi-objective Reliability-Redundancy Allocation Problem with Interval Type-2 Fuzzy Uncertainty. IEEE Trans Fuzzy Syst 2017. doi:10.1109/TFUZZ.2017.2722422.

[27] 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.

[28] 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.

[29] Martorell S, Carlos S, Sánchez A, Serradell V. Constrained optimization of test intervals using a steady-state genetic algorithm. Reliab Eng Syst Saf 2000;67:215–32. doi:10.1016/S0951-8320(99)00074-5.

[30] Giuggioli Busacca P, Marseguerra M, Zio E. Multiobjective optimization by genetic algorithms: application to safety systems. Reliab Eng Syst Saf 2001;72:59–74.

[31] Kennedy J, Eberhart R. Particle swarm optimization. Neural Networks, 1995 Proceedings, IEEE Int Conf 1995;4:1942–8 vol.4. doi:10.1109/ICNN.1995.488968.

[32] Shi Y, Eberhart R. Parameter selection in particle swarm optimization. Evol. Program. VII, 1998, p. 591–600. doi:10.1007/BFb0040810.

[33] Garg H, Sharma SP. Multi-objective reliability-redundancy allocation problem using particle swarm optimization. Comput Ind Eng 2013;64:247–55. doi:10.1016/j.cie.2012.09.015.

[34] Mezura-Montes E, Coello Coello CA. Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm Evol Comput 2011;1:173–94.

doi:10.1016/j.swevo.2011.10.001.

[35] Yang XS, Deb S. Cuckoo search: Recent advances and applications. Neural Comput Appl 2014;24:169–74. doi:10.1007/s00521-013-1367-1.

[36] Yang X-S, Deb S. Engineering Optimisation by Cuckoo Search. Int J Math Model Numer Optim 2010;1:330–43. doi:10.1504/IJMMNO.2010.035430.

[37] Yang X-S, Deb S. Cuckoo Search via Levy Flights. World Congr Nat Biol Inspired Comput 2009:210–4. doi:10.1109/NABIC.2009.5393690.

[38] Hakli H, Uğuz H. A novel particle swarm optimization algorithm with Levy flight. Appl Soft Comput J 2014;23:333–45. doi:10.1016/j.asoc.2014.06.034.

[39] Jensi R, Jiji GW. An enhanced particle swarm optimization with levy flight for global optimization. Appl Soft Comput J 2016;43:248–61. doi:10.1016/j.asoc.2016.02.018.

[40] Hariya Y, Kurihara T, Shindo T. Lévy flight PSO. 2015 IEEE Congr. Evol. Comput., Japan: 2015.

[41] Jana ND, Sil J. Particle swarm optimization with Lévy flight and adaptive polynomial mutation in gbest particle. Recent Adv. Intell. Informatics. Springer, 2014, p. 275–82.

[42] Ravi V, Murty BSN, Reddy J. Nonequilibrium simulated-annealing algorithm applied to reliability\noptimization of complex systems. IEEE Trans Reliab 1997;46:2119–25. doi:10.1109/24.589951.

[43] Zhang H, Hu X, Shao X, Li Z, Wang Y. IPSO-based hybrid approaches for reliability–redundancy allocation problems. Sci China Technol Sci 2013;56:2854–2864.

[44] Garg H, Sharma SP. Reliability-redundancy allocation problem of pharmaceutical plant. J Eng Sci Technol 2013;8:190–8.

Table captions

Figure captions

Table 1 – Data used in series system and complex (bridge) system.

Subsystem i	$10^5\alpha_i$	β_i	v_i	w_i	V	C	W	T (h)
1	2.330	1.5	1	7	110	175	200	1000
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 2 – Data used in series-parallel system.

Subsystem i	$10^5\alpha_i$	β_i	v_i	w_i	V	C	W	T (h)
1	2.500	1.5	2	3.5	180	175	100	1000
2	1.450	1.5	4	4				
3	0.541	1.5	5	4				
4	0.541	1.5	8	3.5				
5	2.100	1.5	4	4.5				

Table 3 – Data used in overspeed protection system.

Subsystem i	$10^5\alpha_i$	β_i	v_i	w_i	V	C	W	T (h)
1	1.0	1.5	1	6	250	400	500	1000
2	2.3	1.5	2	6				
3	0.3	1.5	3	8				
4	2.3	1.5	2	7				

Table 4 – Data used in large scale reliability-redundancy allocation problem.

Subsystem i	$10^5\alpha_i$	β_i	v_i	w_i	V	C	W	T (h)
1	0.6	1.5	2	8	600	700	900	1000
2	0.1	1.5	5	9				
3	1.2	1.5	5	6				
4	0.3	1.5	4	10				
5	2.9	1.5	4	8				
6	1.7	1.5	1	9				
7	2.6	1.5	1	9				
8	2.5	1.5	4	7				
9	1.3	1.5	4	9				
10	1.8	1.5	3	8				
11	2.4	1.5	3	9				
12	1.3	1.5	1	8				
13	1.2	1.5	1	7				
14	2.1	1.5	3	10				
15	0.9	1.5	4	6				
16	1.3	1.5	5	7				
17	1.9	1.5	1	7				
18	2.7	1.5	4	8				
19	2.8	1.5	2	9				
20	1.5	1.5	1	9				

Table 5 – Data used in pharmaceutical plant.

Subsystem i	$10^5\alpha_i$	β_i	v_i	w_i	V	C	W	T (h)
1	0.611360	1.5	4	9	289	553	483	1000
2	4.032464	1.5	5	7				
3	3.578225	1.5	3	5				
4	3.654303	1.5	2	9				
5	1.163718	1.5	3	9				
6	2.966955	1.5	4	10				
7	2.045865	1.5	1	6				
8	2.649522	1.5	1	5				
9	1.982908	1.5	4	8				
10	3.516724	1.5	4	6				

Table 6 – Results for the series system.

Method	(n_1, n_2, n_3, n_4, n_5)	(r_1, r_2, r_3, r_4, r_5)	R_s	V_s	C_s	W_s	NFE	CPU (s)	SD
ADAP-PSO	(3, 2, 2, 3, 2)	(0.76606, 0.86232, 0.89586, 0.69454, 0.85095)	**0.91000**	**73**	167.75780	**164.99906**	**600**	**10.2**	**3.7E–08**
PSO	(3, 2, 3, 3, 2)	(0.76226, 0.85684, 0.82116, 0.68724, 0.84569)	0.90924	88	**153.71193**	189.42752	1000	25.8	5.1E–03

Table 7 – Results for the series-parallel system.

Method	(n_1, n_2, n_3, n_4, n_5)	(r_1, r_2, r_3, r_4, r_5)	R_s	V_s	C_s	W_s	NFE	CPU (s)	SD
ADAP-PSO	(1, 1, 1, 1, 1)	(0.73520, 0.77132, 0.79152, 0.79278, 0.82612)	**0.90927**	**23**	43.61613	**25.03849**	**640**	**11.6**	**4.8E–08**
PSO	(2, 1, 1, 1, 1)	(0.63311, 0.78775, 0.75261, 0.75100, 0.76100)	0.90900	29	**34.07636**	32.08545	1100	29.3	2.9E–02

Table 8 – Results for the complex (bridge) system.

Method	(n_1, n_2, n_3, n_4, n_5)	(r_1, r_2, r_3, r_4, r_5)	R_s	V_s	C_s	W_s	NFE	CPU (s)	SD
ADAP-PSO	(2, 1, 1, 1, 1)	(0.68488, 0.84998, 0.81378, 0.57235, 0.57923)	0.90011	**15**	49.14271	**62.88688**	**700**	**13.1**	**1.6E−11**
PSO	(2, 2, 1, 1, 1)	(0.63609, 0.71268, 0.73690, 0.51368, 0.50591)	0.90011	21	**32.83563**	78.99422	1240	32	2.7E−05

Table 9 – Results for the overspeed protection system.

Method	(n_1, n_2, n_3, n_4)	(r_1, r_2, r_3, r_4)	R_s	V_s	C_s	W_s	NFE	CPU (s)	SD
ADAP-PSO	(5, 5, 4, 5)	(0.89134, 0.87504, 0.93966, 0.87255)	0.99990	173	**359.15906**	418.56759	**660**	**12.6**	**3.9E−12**
PSO	(5, 5, 4, 5)	(0.89040, 0.87481, 0.94066, 0.87457)	0.99990	173	362.11345	418.56759	1460	35.8	4.1E−04

Table 10 – Results for the complex bridge network system.

Method	(r_1, r_2, r_3, r_4, r_5)	R_s	C_s	NFE	CPU (s)	SD
ADAP-PSO	(0.93625, 0.93625, 0.79463, 0.93625, 0.93625)	0.99042	**5.02032**	**800**	**14.7**	**1.3E–10**
PSO	(0.93625, 0.93626, 0.79461, 0.93625, 0.93625)	0.99042	5.02033	1680	40.2	2.7E–06

Table 11 – Results for the life-support system in a space capsule (Case 1).

Method	(r_1, r_2, r_3, r_4, r_5)	R_s	C_s	NFE	CPU (s)	SD
ADAP-PSO	(0.5, 0.84296, 0.5, 0.5)	**0.90111**	**642.34313**	**480**	**8.4**	**6.4E–08**
PSO	(0.5, 0.5, 0.88572, 0.5)	0.90000	647.78277	1080	24.6	5.1E–05

Table 12 – Results for the life-support system in a space capsule (Case 2).

Method	(r_1, r_2, r_3, r_4, r_5)	R_s	C_s	NFE	CPU (s)	SD
ADAP-PSO	(0.82543, 0.89022, 0.62715, 0.72922)	0.99000	**390.57229**	**540**	**9.6**	**2.7E–08**
PSO	(0.82515, 0.89022, 0.62832, 0.72885)	0.99000	390.59632	1380	30.1	8.3E–05

Table 13 – Results for the large scale reliability-redundancy allocation system.

Method	(n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8,n_9,n_{10},n_{11},n_{12},n_{13}, n_{14},n_{15},n_{16},n_{17},n_{18},n_{19},n_{20})	(r_1,r_2,r_3,r_4,r_5,r_6,r_7,r_8,r_9,r_{10},r_{11},r_{12},r_{13},r_{14},r_{15},r_{16},r_{17},r_{18},r_{19},r_{20})	R_s	V_s	C_s	W_s	NFE	CPU (s)	SD
ADAP-PSO	(2, 2, 3, 2, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3)	(0.91432, 0.95113, 0.82342, 0.92682, 0.79160, 0.87902, 0.82813, 0.80264, 0.84649, 0.82548, 0.86274, 0.86767, 0.82950, 0.81622, 0.86856, 0.85531, 0.83680, 0.82385, 0.80989, 0.83882)	**0.88248**	**237**	315.41744	**446.88191**	**2040**	**32.7**	**4.9E–06**
PSO	(3, 2, 3, 2, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3)	(0.84786, 0.94708, 0.82790, 0.92802, 0.80147, 0.88172, 0.80901, 0.80075, 0.83150, 0.81812, 0.88752, 0.84490, 0.84807, 0.81841, 0.86157, 0.83602, 0.83042, 0.81191, 0.88257, 0.82947)	0.88000	247	**293.84520**	471.31037	4120	71.4	1.8E–03

Table 14 – Results for the pharmaceutical plant.

Method	(n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8,n_9,n_{10})	(r_1,r_2,r_3,r_4,r_5,r_6,r_7,r_8,r_9,r_{10})	R_s	V_s	C_s	W_s	NFE	CPU (s)	SD
ADAP-PSO	(2, 3, 3, 3, 3, 3, 3, 4, 3, 3)	(0.92469, 0.80899, 0.80789, 0.82220, 0.85845, 0.81766, 0.84563, 0.76321, 0.84496, 0.81642)	0.95015	**266**	**507.81054**	**439.69862**	**1540**	**29.3**	**4.9E−10**
PSO	(3, 3, 3, 3, 3, 3, 3, 3, 3, 3)	(0.89017, 0.81901, 0.83056, 0.82097, 0.83260, 0.81388, 0.80819, 0.83819, 0.83303, 0.81810)	**0.95018**	279	509. 17010	444.57000	3440	65.8	7.2E−06

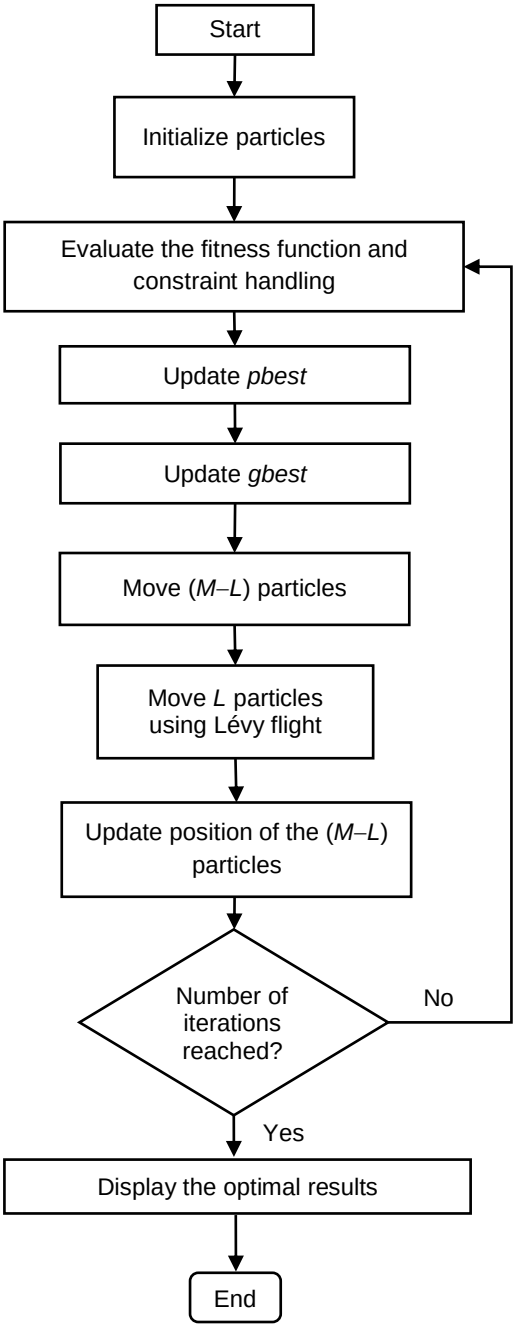

Figure 1 – Flowchart of the implemented ADAP-PSO.

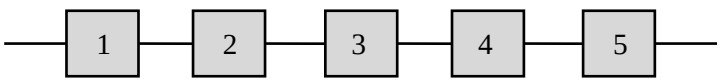

Figure 2 – Series system.

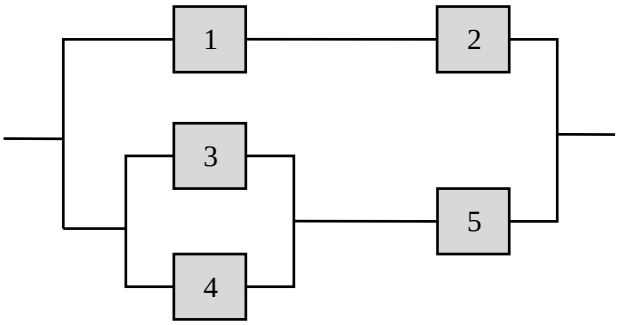

Figure 3 – Series-parallel system.

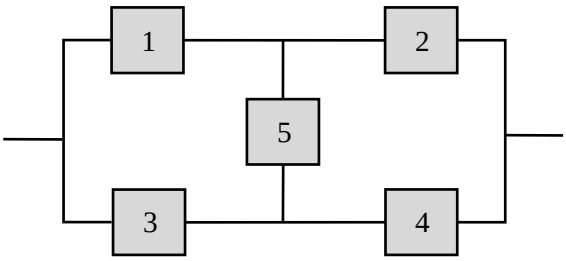

Figure 4 – Complex (bridge) system.

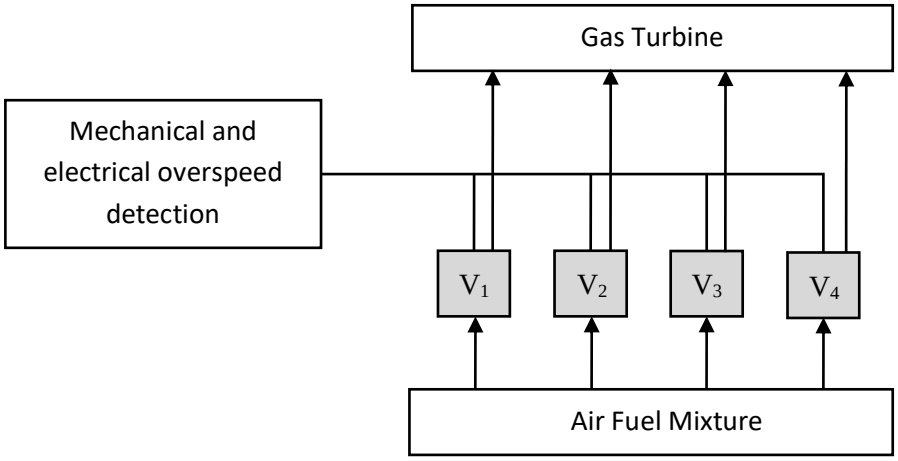

Figure 5 – Overspeed protection system.

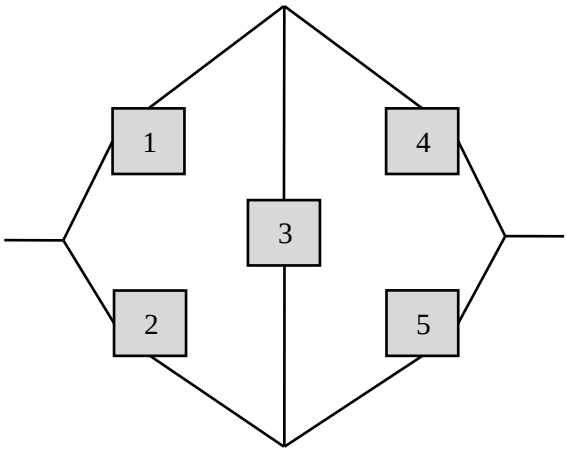

Figure 6 – Complex bridge network system.

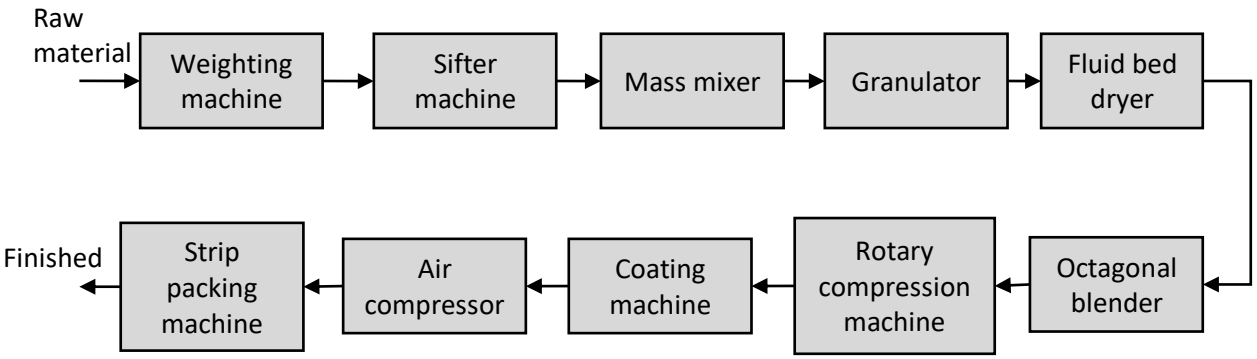

Figure 7 – Pharmaceutical plant.

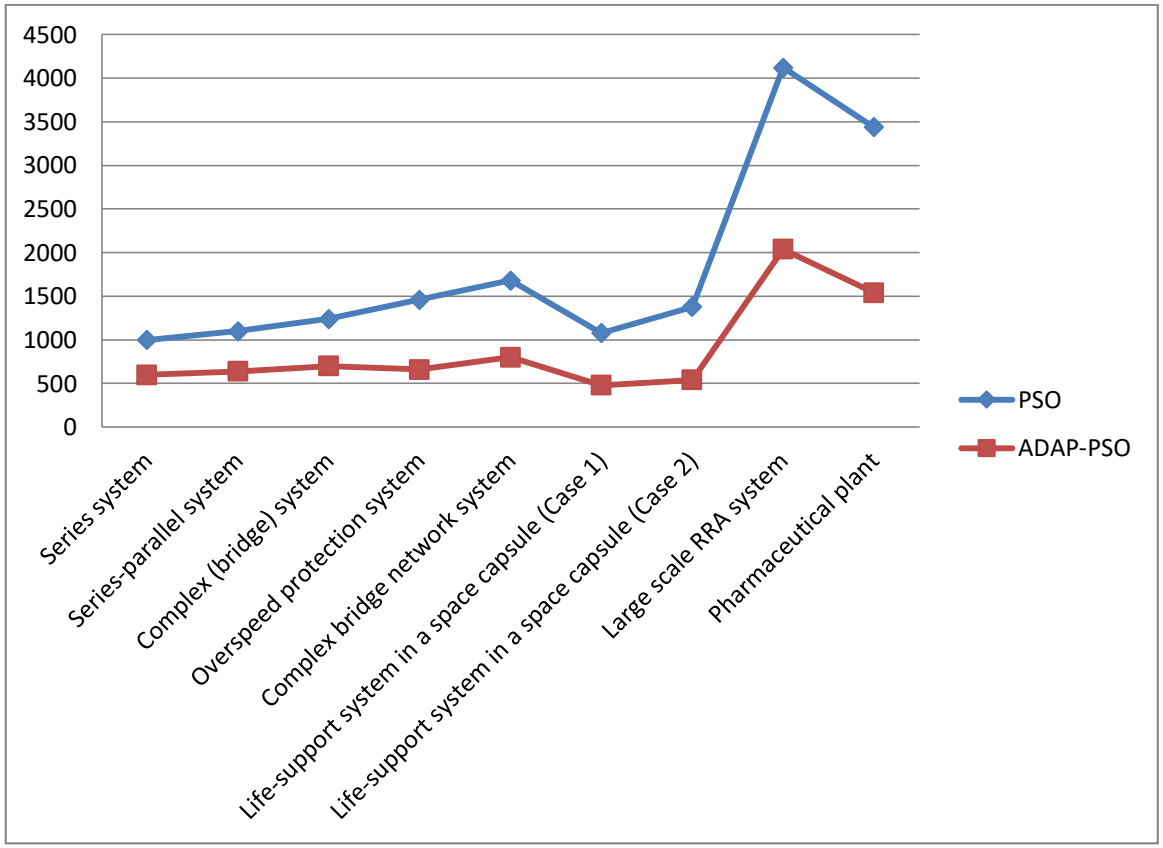

Figure 8 – NFE required by each algorithm.