A hybrid Jaya Algorithm for Reliability–Redundancy Allocation Problems

Sahand Ghavidel[a*], Ali Azizivahed[b], Li Li[a]

[a]Faculty of Engineering and Information Technology, University of Technology, Sydney, PO Box 123, Broadway, NSW 2007, Australia

[b]Department of Electrical and Electronics Engineering, Shiraz University of Technology, Shiraz, Iran

[*]Corresponding Author: S. Ghavidel, Faculty of Engineering and Information Technology, University of Technology, Sydney, PO Box 123, Broadway, NSW 2007, Australia (e-mail: sahand.ghavideljirsaraie@student.uts.edu.au).

Abstract:

This paper proposes an efficient improved hybrid Jaya algorithm based on Time-Varying Acceleration Coefficients (TVAC) and learning phase introduced in Teaching-Learning-Based Optimization (TLBO), named LJaya-TVAC algorithm, for solving various types of nonlinear mixed-integer Reliability–Redundancy Allocation Problems (RRAPs) and standard real-parameter test functions. RRAPs include series, series–parallel, complex (bridge) and overspeed protection systems. The search power of proposed LJaya-TVAC algorithm for finding the optimal solutions is firstly tested on the standard real-parameter uni-modal and multi-model functions with dimension of 30 to 100, and then tested on various types of nonlinear mixed-integer RRAPs. The results are compared with the original Jaya algorithm and best results reported in the recent literature. The obtained optimal results of proposed LJaya-TVAC algorithm provide evidence for the better and acceptable optimization performance compared to the original Jaya algorithm and other reported optimal results.

Keywords: Jaya algorithm, Reliability–Redundancy Allocation Problems (RRAPs), Teaching-Learning-Based Optimization (TLBO) algorithm, Time-Varying Acceleration Coefficients.

Nomenclature:

C	upper limit on the cost of the system	R_s	reliability of the system
$f(.,.)$	objective function of reliability of the system	r_d (d=1:m)	reliability of every component available for the dth subsystem
$g(.,.)$	set of constraint functions	T	operating time of problem 4
m	number of subsystems in the system	V	upper limit on the sum of the subsystems' products of volume
n_d (d=1:m)	number of components in the dth subsystem	V_1 to V_4	control valves of problem 4
$P1$	problem 1 or series test system	v_d	volume of each component in the dth subsystem
$P2$	problem 2 or series–parallel test system	W	upper limit on the weight of the system
$P3$	problem 3 or complex test system	w_d	weight of each component in the dth subsystem
$P4$	problem 4 or overspeed protection test system	Z^+	set of positive integers in the discrete space
$P5$	problem 5 or large-scale test system	α_d and β_d	physical characteristics of the system components
q_d (d=1:m)	failure probability of each component in the dth subsystem		
R_d (d=1:m)	reliability of the dth subsystem		

1. Introduction

The main goal of maximizing the Reliability–Redundancy Allocation Problems (RRAPs) includes a selection of the levels and redundancy of the components to maximize and improve the system reliability and performance. The RRAPs are beneficial for the design of the systems that are brought together on a large scale and produced in a large-scale industrial operation using off-the-shelf components. These days, as a consequence of increasing system complications and unpredictable behaviors, evaluating the reliability of the systems and the requirement for improving the reliability of the systems have become very interesting and significant (Zhang et al. 2013).

Various optimization programming and evolutionary techniques have been employed to optimize various types of RRAPs, such as: Genetic Algorithm (GA) (Hsieh, Chen, and Bricker 1998; He et al. 2013), Particle Swarm Optimization (PSO) (Wu et al. 2011; Zhang et al. 2013; Tan, Tan, and Deng 2013), Simulated Annealing (SA) (Dohi et al. 2006; Suman 2003), Harmony Search (HS) (Zou et al. 2010; Zou et al. 2011; dos Santos Coelho, Diego, and Mariani 2011), Tabu Search (TS) (Jang and Kim 2011; Liu and Qin 2014), a cold-standby redundancy strategy (Ardakan and Hamadani 2014a), a combination search algorithm based on Hooke–Jeeves pattern search and dynamic programming (Liu 2006), the RRAP of parking facilities in the real system using a hybrid GA (Hamadani et al. 2013), Ant Colony Optimizer (ACO) (Liang and Smith 2004), Cuckoo Search (CS) (Valian and Valian 2013; Valian et al. 2013), Memetic Algorithm (MA) (Pourdarvish and Ramezani 2013), Imperialist Competitive Algorithm (ICA) (Afonso, Mariani, and dos Santos Coelho 2013), Artificial Bee Colony (ABC) (Yeh and Hsieh 2011), a hybrid algorithm of space partitioning and tabu-genetic (SP/TG) (Ouzineb, Nourelfath, and Gendreau 2011) for non-homogeneous RRAP, Honey Bee Mating Optimization (HBMO) (Sadjadi and Soltani 2012), a new mixed strategy which uses cold-standby and active strategies with a proposed GA for reliability optimization of series–parallel systems (Ardakan and Hamadani 2014b), compromise programming (Soltani, Sadjadi, and Tavakkoli-Moghaddam 2015), binary equivalent models and Mixed Integer Nonlinear Programming (MINLP) for the cold standby RRAP (Feizollahi, Soltani, and Feyzollahi 2015), Immune Algorithm (IA) (Chen and You 2005; Chen 2006), a multi-objective multi-stage reliability growth planning strategy (Li, Mobin, and Keyser 2016) using a modified non-dominated sorting GA (NSGA-II) in the early product-development stage and also multi-objective reliability optimization using GA proposed by (Ardakan, Hamadani, and Alinaghian 2015), Improved Bat Algorithm (IBA) (Liu 2016), neighbourhood search heuristic method with nonlinear programming (Chatwattanasiri, Coit, and Wattanapongsakorn 2016), and a Penalty Guided Stochastic Fractal Search (PSFS) (Mellal and Zio 2016), a new interpretation and formulation of the RRAP (Ardakan Abouei et al. 2016) using the mixed new strategy and a modified version of the GA (MVGA), showing distinct advantages compared to traditional methods, and etc. A state of the art survey of optimization techniques for various types of RRAP to 2014 is presented in (Soltani 2014).

Jaya algorithm (Rao 2016) is a new simple and efficient algorithm. Similar to the other algorithms, it only has the common parameters that will be determined by the user like population number and iterations of algorithm without need of any specific control parameters that would be determined by the user. This algorithm is based on the best and the worst candidate solutions in the iterations (Rao 2016). It has good feasibility and performance in solving different engineering optimization problems such as complex constrained design optimization (Venkata Rao and Waghmare 2016), dimensional optimization of a micro-channel heat sink (Rao et al. 2016), and surface grinding process optimization (Rao, Rai, and Balic 2016). To the authors' best knowledge, this is the first time that the Jaya algorithm is used for

RRAPs in this study.It can be shown that the results obtained by Jaya algorithm for RRAPs are suitable and good.

RRAPs are an important requirement of various systems. In many systems, balance in weights, number of components in subsystems and/or low cost are desired. In various cases, the methods and optimization algorithms described provide solutions very close to optimality for RRAPs of various systems. In this study, an improved new optimization algorithm has been presented to meet these requirements to solve various RRAPs. This paper proposes a hybrid enhanced Jaya algorithm based on the learning phase of Teaching-Learning-Based Optimization (TLBO) algorithm introduced in (Rao, Savsani, and Vakharia 2011; Rao 2015; Rao and Patel 2012) with its applications (Rao 2015; Ghasemi 2014, 2015), and a new Time-Varying Acceleration Coefficients (TVAC) proposed by (Ratnaweera, Halgamuge, and Watson 2004) for solving various types of nonlinear mixed-integer RRAPs. In the first phase of proposed Jaya-TVAC algorithm, a TVAC is added to the Jaya algorithm, and then in the second phase, a learning phase of the TLBO algorithm (Rao, Savsani, and Vakharia 2011; Rao 2015; Rao and Patel 2012) is added to the Jaya-TVAC algorithm (LJaya-TVAC algorithm) for finding the better final solutions with higher convergence rate compared with the original algorithm. The two new time-varying acceleration coefficients added to the Jaya algorithm increase the search power around the global optimal solution in the primary iterations for faster convergence. The added learning phase also increases the search power in the final iterations for finding the better final solutions with higher convergence rate through the increased local search of Jaya.

This study is arranged as follows: Section 2 provides a formulation and description of the RRAPs for test systems such as series, series–parallel, complex and overspeed protection system. In Sections 3 and 4, the Jaya and hybrid enhanced Jaya algorithms using TVAC and learning phase are presented. Section 5 shows performance of the proposed optimization algorithms in solving RRAPs for the various systems and also standard real-parameter test functions. We end this study with some conclusions for the hybrid enhanced Jaya algorithm in Section 6.

2. Reliability-Redundancy Allocation Problems (RRAPs)

The main purpose of optimization of the RRAPs is to enhance the reliability of these systems (maximization of the overall system reliability) by means of using component reliabilities allocation ($r = (r_1, r_2, \ldots, r_m)$) and redundancy allocation number ($n= (n_1, n_2, \ldots, n_m)$). The nonlinear mixed-integer programming model of these problems can be formulated by maximizing the reliability of the system as the objective function subject to multiple nonlinear constraints as the following equations:

$$\text{Maximize } R_s = f\left(r,n\right), \tag{1}$$

$$\begin{aligned} &\text{subject to } g\left(r,n\right) \le l \\ &\qquad 0 \le r_d \le 1,\ n_d \in Z^+,\ 0 \le d \le m. \end{aligned} \tag{2}$$

where R_s is the reliability of the system, f(.,.) and g(.,.) are the objective function and constraints of the RRAPs, respectively; g(.) is usually associated with the system cost, volume and weight limitations. $r = (r_1, r_2, \ldots, r_m)$ and $n= (n_1, n_2, \ldots, n_m)$ are the component reliabilities and redundancy allocation number vectors for m subsystems, and also, l is the system resource limitation.

Four RRAPs including the series system (problem 1 ($P1$)), series–parallel system (problem 2 ($P2$)), complex (bridge) system (problem 3 ($P3$)) and overspeed protection system of a gas turbine (problem 4 ($P4$)) are evaluated in this paper as follows.

2.1. Series system (*P*1)

The series system with *m*=5 subsystems (for *d*=1:*m*) for nonlinear mixed-integer RRAP was presented in (Chen 2006; Hsieh, Chen, and Bricker 1998). The block diagram of the series system with five subsystems is shown in Fig. 1 (Afonso, Mariani, and dos Santos Coelho 2013). The RRAP of the series system can be formulated as follows (Chen 2006):

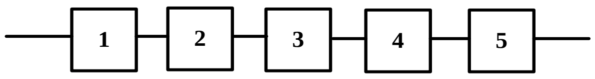


Fig. 1. Block diagram of the series system

$$\text{Maximize } f_1(r,n)=\prod_{d=1}^{m} R_d(n_d) \qquad (3)$$
$$0 \le r_d \le 1,\ n_d \in Z^+ \text{(positive integer in the discrete space).}$$

Here, $R_d(n_d)=1-q_d^{n_d}$ is the reliability of *d*th subsystem, q_d = 1- r_d is the failure probability of each component in *d*th subsystem, and n_d is the number of components in the *d*th subsystem. The objection function is subject to the following constraints:

1- The combination of weight, volume, and redundancy allocation number constraint $g_1(r,n)$:

$$g_1(r,n)=\sum_{d=1}^{m} w_d v_d^2 n_d^2 \le V \qquad (4)$$

where, w_d is the weight of each component in *d*th subsystem; v_d is the volume of each component in *d*th subsystem, *V* is the upper limit on the sum of the subsystems' products of volume.

2- The system cost constraint $g_2(r,n)$:

$$g_2(r,n)=\sum_{d=1}^{m} \alpha_d \left(-\frac{1000}{\ln r_d}\right)^{\beta_d} \left[n_d + e^{0.25 n_d}\right] \le C \qquad (5)$$

where, α_d and β_d are physical characteristics of the system components. Also, *C* is the upper limit on the cost of the system.

3- The system weight constraint $g_3(r,n)$:

$$g_3(r,n)=\sum_{d=1}^{m} w_d n_d e^{0.25 n_d} \le W \tag{6}$$

where, W is the upper limit on the weight of the system. The parameters of the series system (Chen 2006; Hsieh, Chen, and Bricker 1998) are given in Table A.1 in Appendix.

2.2. Series–parallel system (*P*2)

The series–parallel system (*P*2), with the same $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$ constraints as those of *P*1, is shown in Fig. 2 (Chen 2006; Hsieh, Chen, and Bricker 1998). The input parameters of this test system (*P*2) (Chen 2006; Hsieh, Chen, and Bricker 1998) are given in Table A.2 in Appendix.

The nonlinear mixed-integer RRAP of the series–parallel system can be formulated as follows (Dohi et al. 2006):

$$\begin{gathered}\text{Maximize } f_2(r,n)=1-(1-R_1R_2)\big(1-\big(1-(1-R_3)(1-R_4)\big)R_5\big)\\ 0\le r_d\le 1,\ n_d\in Z^{+}\text{(positive integer in the discrete space)},\ 1\le d\le 5.\end{gathered} \tag{7}$$

subject to $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$.

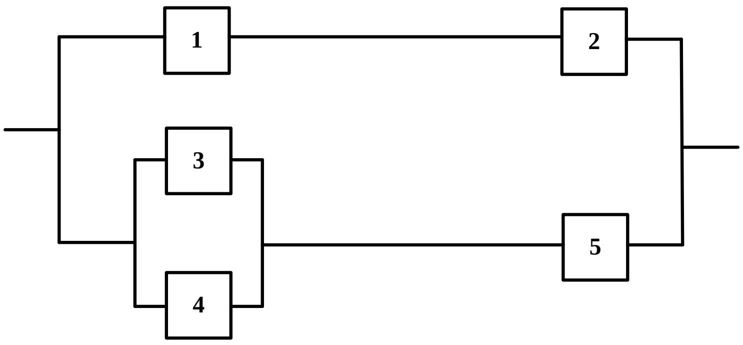


Fig. 2. Block diagram of the series–parallel system.

2.3. Complex (bridge) system (*P*3)

The complex (bridge) system (*P*3) is shown in Fig. 3 which has the same non-linear constraints $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$ as those of the series and series–parallel systems optimization problems (Chen 2006). The input parameters of this test system (*P*3) (Chen 2006; Hsieh, Chen, and Bricker 1998) are given in Table A.3 in Appendix.

The nonlinear mixed-integer RRAP of the complex (bridge) system (*P*3) can be formulated as follows:

$$\begin{aligned}&\text{Maximize } f_3(r,n)=R_1R_2+R_3R_4+R_1R_4R_5+R_2R_3R_5-R_1R_2R_3R_4-R_1R_2R_3R_5\\&-R_1R_2R_4R_5-R_1R_3R_4R_5-R_2R_3R_4R_5+2R_1R_2R_3R_4R_5\\&\quad 0\le r_d\le 1,\ n_d\in Z^+,\ 1\le d\le 5.\end{aligned}\tag{8}$$

subject to $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$.

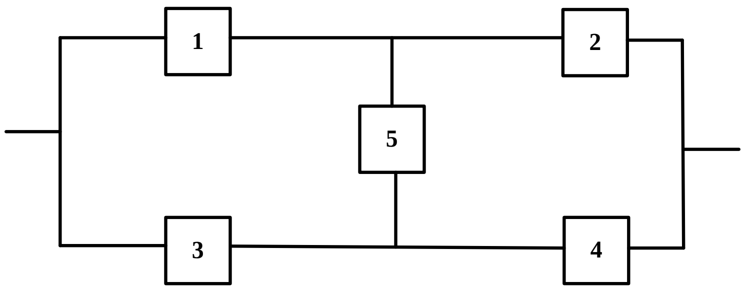


Fig. 3. Block diagram of the complex (bridge) system.

2.4. Overspeed protection system of a gas turbine (*P*4)

The overspeed detection is constantly supplied by the mechanical and electrical systems. When an overspeed happens, it is essential to stop the fuel source by means of using control valves (V_1 to V_4). The overspeed protection system of a gas turbine for the fourth nonlinear mixed-integer RRAP is shown in Fig. 4. The input parameters of the overspeed protection system (Chen 2006) are given in Table A.4 in Appendix.

This RRAP of the overspeed protection system a gas turbine can be formulated as follows:

$$\text{Maximize } f_4(r,n) = \prod_{d=1}^{m}\left[1-(1-r_d)^{n_d}\right].$$
$$0.5 \le r_d \le \left(1-10^{-6}\right), \tag{9}$$
$$1 \le n_d \le 10,\ n_d \in Z^+.$$

This objective function is subject to the following constraints:

1- The combination of weight, volume, and redundancy allocation number constraint $g_1(r,n)$:

$$g_1(r,n) = \sum_{d=1}^{m} v_d^2 n_d^2 \le V \tag{10}$$

2- The system cost constraint $g_2(r,n)$:

$$g_2(r,n) = \sum_{d=1}^{m} C(r_d)\left[n_d + e^{0.25 n_d}\right] \le C,$$
$$C(r_d) = \alpha_d \left(-\frac{T}{\ln r_d}\right)^{\beta_d}. \tag{11}$$

where, $C(r_d)$ is the cost of each component with reliability r_d at the dth subsystem, and T is the operating time in which the component must not fail.

3- The system weight constraint $g_3(r,n)$:

$$g_3(r,n) = \sum_{d=1}^{m} w_d n_d e^{0.25 n_d} \le W \tag{12}$$

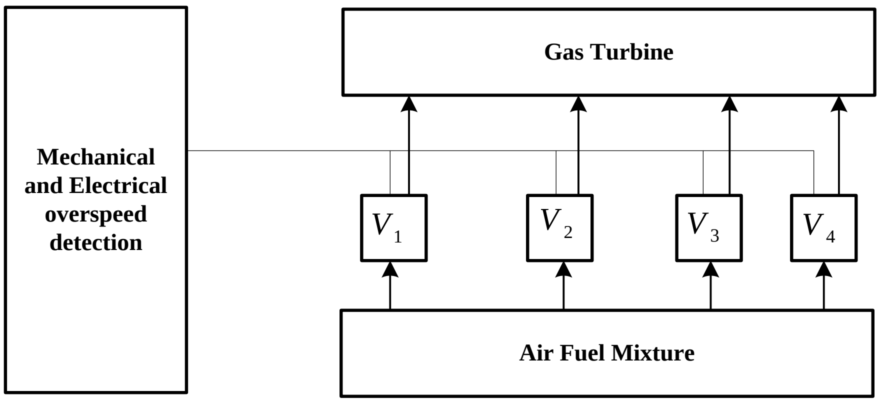


Fig. 4. The diagram block for the overspeed protection system of a gas turbine.

2.5. Large scale RRAP (*P*5):

To clearly show the effectiveness of the proposed LJaya-TVAC algorithm for RRAP, a large scale system (Zhang et al. 2013) is used with the same non-linear constraints $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$ as those of the overspeed protection test system (Mellal and Zio 2016). The formulation of this problem can be written as follows (Zhang et al. 2013):

$$\begin{aligned}&\text{Maximize } f_5(r,n)=\prod_{d=1}^{m}\left[1-(1-r_d)^{n_d}\right].\\&\qquad 0.5\le r_d\le\left(1-10^{-6}\right),\ m=20,\\&1\le n_d\le 10,\ n_d\in Z^{+}.\end{aligned} \tag{13}$$

The large-scale test system includes forty decisions variables (=m*2=40). The data and input parameters for the large-scale test system are given in (Zhang et al. 2013).

3. Jaya algorithm

Jaya algorithm (Rao 2016) is a recently proposed algorithm which is a powerful and simple optimizer for real-world optimization problems. The original flowchart of the optimization process for Jaya algorithm is shown in Fig. 5 (Rao 2016). In the Jaya algorithm, each member of all population (N), has its own location (solution) in the ith iteration (i = 1:$i_{\max}$) of the algorithm. X_k^i (k =1: N) is defined by the optimization problem parameters in the d-dimensional solution search space: $X_k^i = \left[x_{1,k}^i, x_{2,k}^i, ..., x_{d,k}^i\right]$. The new location value $X_k^{i+1} = \left[x_{1,k}^{i+1}, x_{2,k}^{i+1}, ..., x_{d,k}^{i+1}\right]$ for the kth member X_k^i is achieved by updating the locations iteratively. If $f\left(X_k^{i+1}\right) \le f\left(X_k^i\right)$, the new location value (X_k^{i+1}) replaces the old location value (X_k^i) using the following equation (Rao 2016):

$$X_k^{i+1} = X_k^i + \text{rand}_1^i\left(X_{\text{best}}^i - \left|X_k^i\right|\right) - \text{rand}_2^i\left(X_{\text{worst}}^i - \left|X_k^i\right|\right) \tag{14}$$

where, $X_{\text{best}}^i = \left[x_{1,\text{best}}^i, x_{2,\text{best}}^i, ..., x_{d,\text{best}}^i\right]$ and $X_{\text{worst}}^i = \left[x_{1,\text{worst}}^i, x_{2,\text{worst}}^i, ..., x_{d,\text{worst}}^i\right]$are the best and worst solutions obtained until the ith iteration of the algorithm, respectively. $\text{rand}_1^i = \left[\text{rand}_{1,1}^i, \text{rand}_{1,2}^i, ..., \text{rand}_{1,d}^i\right]$ andare two set of $\text{rand}_2^i = \left[\text{rand}_{2,1}^i, \text{rand}_{2,2}^i, ..., \text{rand}_{2,d}^i\right]$ random numbers in the range [0, 1] in the ith iteration of the algorithm. Also, $\left|X_k^i\right|$is the absolute value of X_k^i.

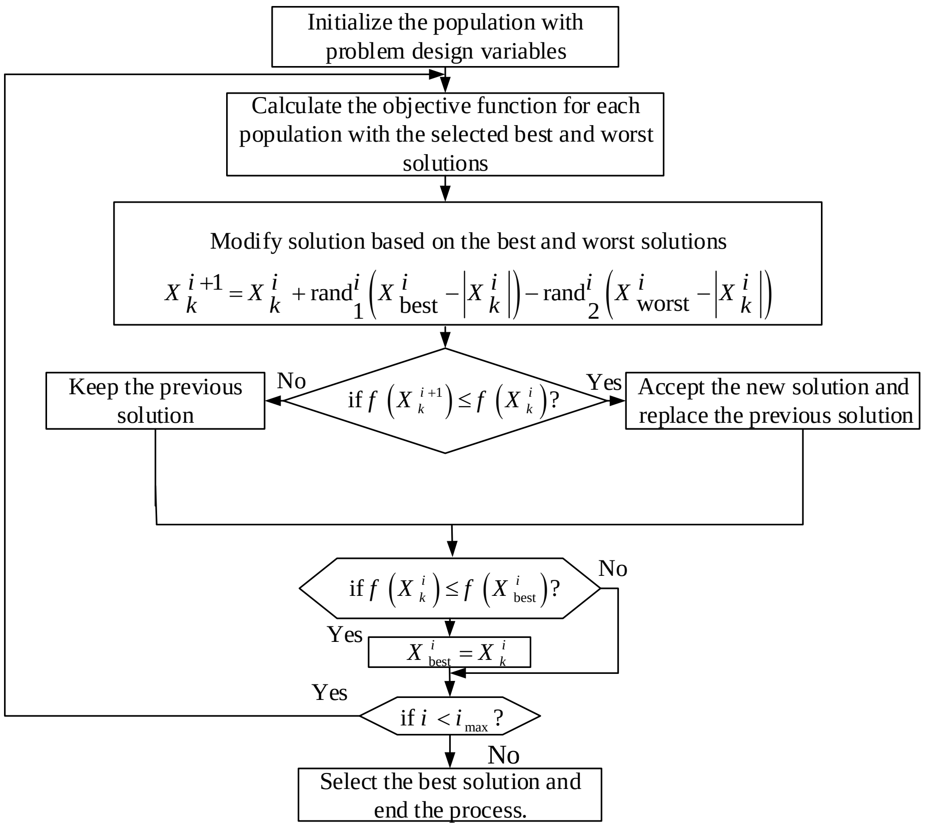

Initialize the population with problem design variables
Calculate the objective function for each population with the selected best and worst solutions
Modify solution based on the best and worst solutions
$X_k^{i+1} = X_k^i + \text{rand}_1^i\left(X_{\text{best}}^i - \left|X_k^i\right|\right) - \text{rand}_2^i\left(X_{\text{worst}}^i - \left|X_k^i\right|\right)$
Keep the previous solution
No
if $f\left(X_k^{i+1}\right) \le f\left(X_k^i\right)$?
Yes
Accept the new solution and replace the previous solution
if $f\left(X_k^i\right) \le f\left(X_{\text{best}}^i\right)$?
No
Yes
$X_{\text{best}}^i = X_k^i$
Yes
if $i < i_{\max}$?
No
Select the best solution and end the process.

Fig. 5. The optimization process of the original Jaya algorithm.

4. The hybrid enhanced Jaya algorithm

In this section, the hybrid enhanced Jaya algorithms using TVAC and learning phase is presented. These algorithms increase the search power around the global optimal solution (X_{best}) in the primary iterations for faster convergence, and also increase the search power in the latest iterations.

4.1. Jaya algorithm with time-varying acceleration coefficients (Jaya-TVAC)

In the first phase, two new time-varying acceleration coefficients c_1^i and c_2^i are proposed based on the method by (Ratnaweera, Halgamuge, and Watson 2004) to improve the Jaya algorithm, which is called Jaya-TVAC algorithm. The new location value for X_k^i is then modified as follows:

$$X_k^{i+1} = X_k^i + c_1^i \times \text{rand}_1^i \left(X_{\text{best}}^i - \left| X_k^i \right| \right) - c_2^i \times \text{rand}_2^i \left(X_{\text{worst}}^i - \left| X_k^i \right| \right) \tag{15}$$

$$c_1^i = c_1^{i=1} - \left(c_1^{i=1} - c_1^{i=i_{\max}} \right) \left(\frac{i}{i_{\max}} \right) \tag{16}$$

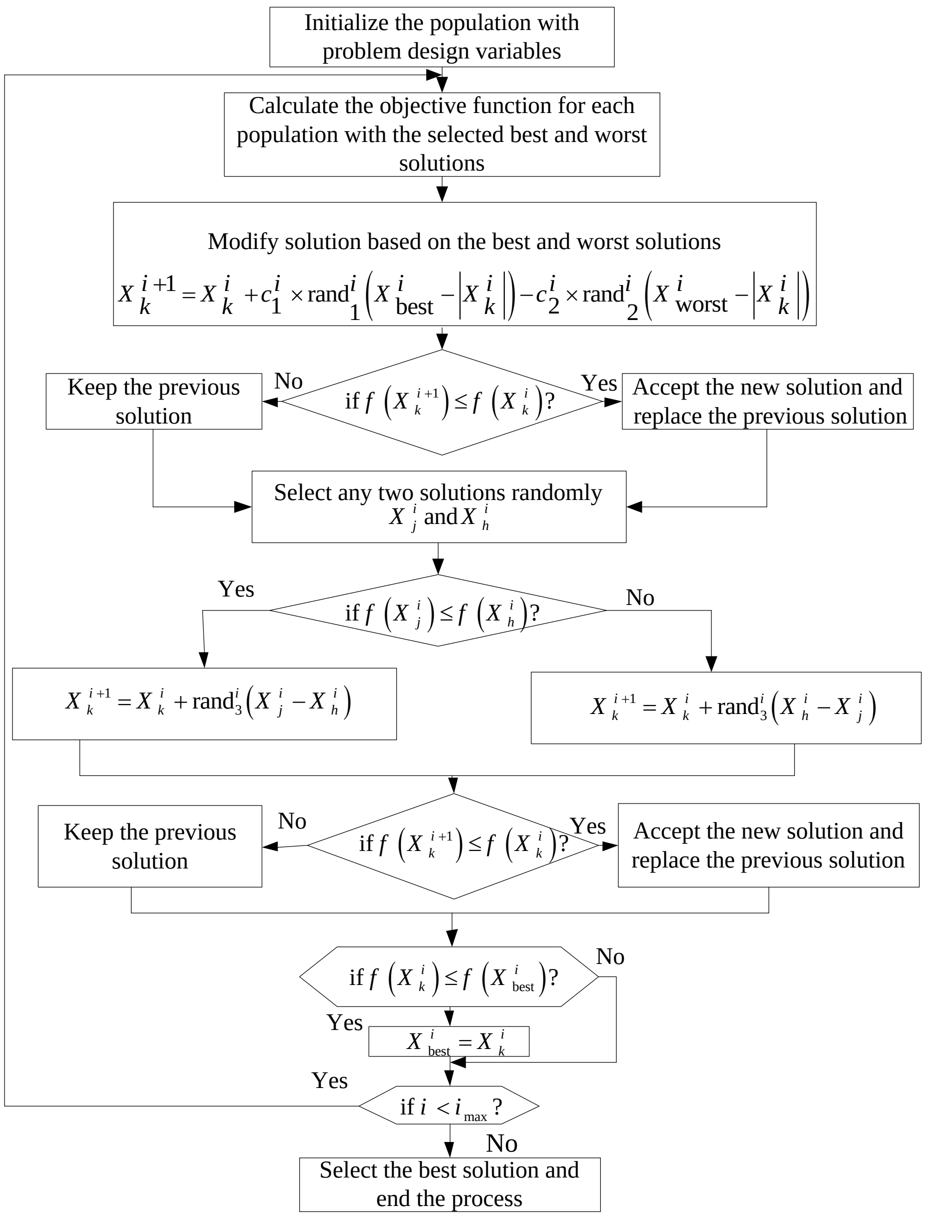
$$c_2^i = c_2^{i=i_{\max}} - \left(c_2^{i=i_{\max}} - c_2^{i=1} \right) \left(\frac{i_{\max} - i}{i_{\max}} \right) \tag{17}$$

where, $c_1^{i=1}=c_2^{i=1}=1$ and $c_1^{i=i_{\max}}=0.5$ and $c_2^{i=i_{\max}}=0$ are obtained for the best values.

4.2. Hybrid Jaya-TVAC algorithm with learning phase (LJaya-TVAC) based on TLBO algorithm

In the second phase, a learning phase introduced in (Rao, Savsani, and Vakharia 2011; Rao 2015; Rao and Patel 2012) is added to the proposed algorithm for finding the better final solutions with higher convergence rate through increased local search of Jaya. The flowchart of the optimization process for **LJaya-TVAC** algorithm is shown in Fig. 6. The new location value X_k^{i+1} can be achieved using (18). Here two solution variables X_j^i (jth member of the population) and X_h^i (hth member of the population) are randomly selected as shown in (18). If the value of the objective function for the new location value X_k^{i+1} is better than the old location value X_k^i ($f\left(X_k^{i+1}\right) \le f\left(X_k^i\right)$), the new location value X_k^{i+1} will replace the old location value X_k^i.

$$X_k^{i+1} = X_k^i + \text{rand}_3^i \times \begin{cases} X_j^i - X_h^i & if\ f\left(X_j^i\right) \le f\left(X_h^i\right) \\ X_h^i - X_j^i & \text{else.} \end{cases} \tag{18}$$



Initialize the population with problem design variables
Calculate the objective function for each population with the selected best and worst solutions
Modify solution based on the best and worst solutions
$X_k^{i+1} = X_k^i + c_1^i \times \text{rand}_1^i \left(X_{\text{best}}^i - \left|X_k^i\right|\right) - c_2^i \times \text{rand}_2^i \left(X_{\text{worst}}^i - \left|X_k^i\right|\right)$
Keep the previous solution
No
if $f\left(X_k^{i+1}\right) \le f\left(X_k^i\right)$?
Yes
Accept the new solution and replace the previous solution
Select any two solutions randomly X_j^i and X_h^i
Yes
if $f\left(X_j^i\right) \le f\left(X_h^i\right)$?
No
$X_k^{i+1} = X_k^i + \text{rand}_3^i\left(X_j^i - X_h^i\right)$
$X_k^{i+1} = X_k^i + \text{rand}_3^i\left(X_h^i - X_j^i\right)$
Keep the previous solution
No
if $f\left(X_k^{i+1}\right) \le f\left(X_k^i\right)$?
Yes
Accept the new solution and replace the previous solution
if $f\left(X_k^i\right) \le f\left(X_{\text{best}}^i\right)$?
No
Yes
$X_{\text{best}}^i = X_k^i$
Yes
if $i < i_{\max}$?
No
Select the best solution and end the process

Fig. 6. The optimization process of LJaya-TVAC algorithm.

5. Results and discussion

5.1. LJaya-TVAC algorithm for real-parameter problems

In the first phase of the study, in order to validate the performance of the LJaya-TVAC algorithm for the real-parameter test functions, various types of real-parameter test functions are chosen (Suganthan et al. 2005). The details of sixth typical unimodal and multi-modal real-parameter test functions (F) that are selected to evaluate the effectiveness of the proposed algorithms are summarized as follows:

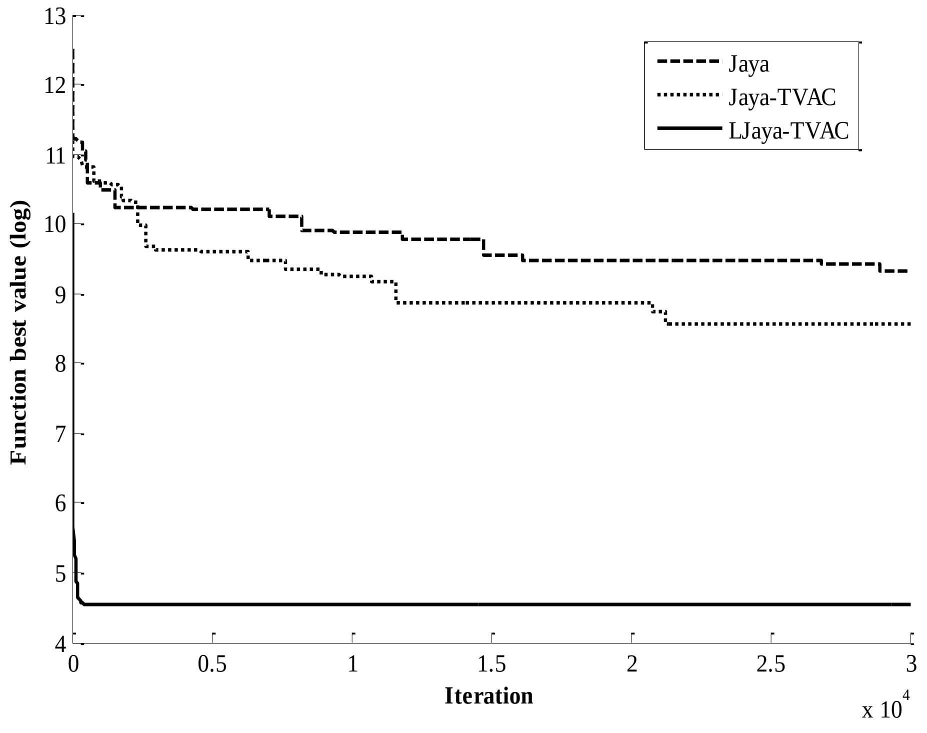
1) F_1 : Shifted Rotated High Conditioned Elliptic (uni-modal, non-separable and scalable test function), $F_1(x)=\sum_{j=1}^{d}\left(10^6\right)^{\frac{j-1}{d-1}} z_j^2, z=(x-o)*M, x=[x_1,x_2,...,x_d],$ with $o=[o_1,o_2,...,o_d]$: the shifted global optimum and, M : orthogonal matrix $x_j \epsilon$ [-100, 100] and $F(x)$ =0.
2) F_2 : Shifted Schwefel's Problem 1.2 with Noise in Fitness (uni-modal, non-separable and scalable test function),
$$F_2(x)=\left(\sum_{j=1}^{d}\left(\sum_{t=1}^{j} z_t\right)^2\right)*\left(1+0.4\left|N(0,1)\right|\right), z=x-o, o=[o_1,o_2,...,o_d]: \text{the shifted global optimum}$$
with $x_j \epsilon$ [-100, 100] and $F(x)$ =0.
3) F_3 : Schwefel's Problem 2.6 with Global Optimum on Bounds (uni-modal, non-separable and scalable test function), $F_3(x)=\max\left\{\left|A_j x-B_j\right|\right\},$ A is a $d*d$ matrix, A_j is the jth row of A, $B_j=A_j*o$.
with $x_j \epsilon$ [-100, 100] and $F(x)$ =0.
4) F_4 : Shifted Rosenbrock's (multi-modal, non-separable and scalable test function),
$$F_4(x)=\sum_{i=1}^{d-1}\left(100\left(z_j^2-z_{j+1}\right)^2+\left(z_j-1\right)^2\right), z=x-o+1.$$ with $x_j \epsilon$ [-100, 100] and $F(x)$ =0.
5) F_5 : Shifted Rotated Ackley's with Global Optimum on Bounds (multi-modal, non-separable and scalable test function),
$$F_5(x)=-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} z_j^2}\right)-\exp\left(\frac{1}{d}\sum_{j=1}^{d}\cos(2\pi z_j)\right)+20+e, z=x-o.$$ with $x_j \epsilon$ [-32.0, 32.0] and $F(x)$ =0.

6) F_6 : Shifted Rastrigin's (multi-modal, separable and scalable test function),
) =0.x(F5.0, 5.0] and -[$\epsilon_j x$with $F_6(x)=\sum_{j=1}^{d}\left(z_j^2-10\cos\left(2\pi z_j\right)+10\right), z=x-o.$

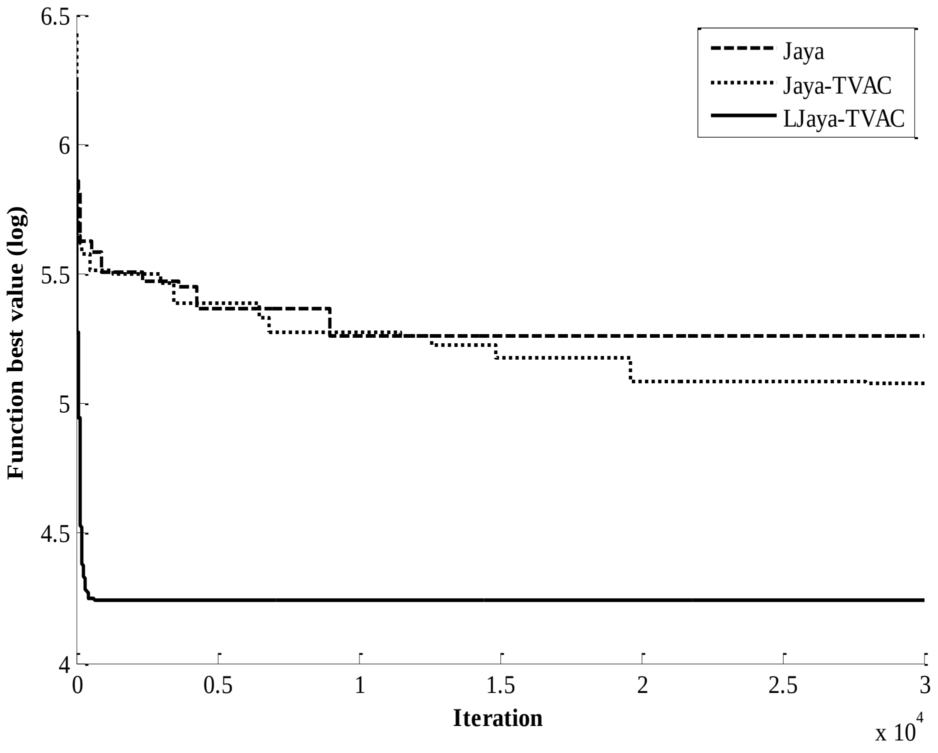
The Mean (mean value of the best results) and Std (standard deviation of the best results) indexes for the proposed Jaya algorithms of each real-parameter problems over 30 runs for d=30 and d=100 with i_{max} =d*1000, and the population sizes of N=50 are given in Table 1. Also Fig. 7 shows the convergence plots of the proposed Jaya algorithms for the real-parameter functions. The proposed LJaya-TVAC algorithm obtains better optimal results with faster convergence characteristics compared to the original Jaya and Jaya-TVAC algorithms. The results show that the proposed LJaya-TVAC method has been successfully implemented to the real-parameter optimization problems with different dimensions.

Table 1. The best results (Mean±Std) obtained from the Jaya algorithms for real-parameter problems.

F	*d*	Jaya	Rank	Jaya-TVAC	Rank	LJaya-TVAC	Rank
F_1	30	5.70e+07± 8.64e+06	3	2.91e+07±2.16e+06	2	**1.93e+04±1.42e+04**	**1**
	100	1.29e+09±1.57e+08	3	8.07e+08±1.13e+08	2	**1.76e+06±6.78e+05**	**1**
F_2	30	1.26e+04±1.27e+03	3	6.07e+03±5.32e+03	2	**9.98e+01± 7.56e+01**	**1**
	100	3.68e+05± 2.31e+04	3	4.42e+04±2.87e+04	2	**2.43e+04± 1.85e+04**	**1**
F_3	30	3.90e+03±2.45e+03	3	4.77e+02± 5.83e+02	2	**1.82e+02± 1.90e+02**	**1**
	100	3.86e+04±3.09e+03	3	1.07e+03±4.65e+03	2	**9.30e+02± 2.24e+03**	**1**
F_4	30	8.19e+07±3.82e+07	3	2.52e+07± 8.14e+06	2	**3.29e+00± 3.21e+00**	**1**
	100	5.36e+09±1.03e+09	3	1.89e+09±7.36e+08	2	**2.98e+00±1.25e+00**	**1**
F_5	30	20.871±0.083	3	20.818±0.044	2	**20.72±0.061**	**1**
	100	21.271±0.027	3	21.065±0.017	2	**20.85±0.012**	**1**
F_6	30	200.50±7.166	3	180.59±12.42	2	**72.41±5.45**	**1**
	100	896.75±69.301	3	854.71±38.59	2	**464.64±26.36**	**1**



(a):



(b):

Fig. 7. Convergence plots of the proposed Jaya algorithms for the real-parameter function with d=30: (a) F_2 and (b): F_6.

5.2. The optimization of RRAPs using LJaya-TVAC algorithm

In the second study, the proposed LJaya-TVAC method is implemented for solving different RRAPs in various test systems. The optimization process of the proposed LJaya-TVAC algorithm can be summarized as follows:

Step 1: Set the initial parameters $X_d^{\min}$ and $X_d^{\max}$ (the minimum and maximum limits of variables for d=1:2*m), i_{max}, N, $c_1^{i=1}=c_2^{i=1}=1$, $c_1^{i=i_{\max}}=0.5$ and $c_2^{i=i_{\max}}=0$, and call out the needed information for intended test system, such as m, V, C, W, w_d, v_d, α_d and β_d ,for all subsystems (for d=1:m).

Step 2: Produce the initial random population matrix (N×2*m) using the minimum and maximum limits of the variables.

Step 3: Calculate the objective function $f(r,n)$ of RRAP by imposing the non-linear constraints $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$ for every available solution in the initial population of the LJaya-TVAC algorithm.

Step 4: Produce the new population of LJaya-TVAC algorithm using (15) (the first phase).

Step 5: Calculate the objective function $f(r,n)$ of RRAP by imposing the non-linear constraints $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$ for the generated population in Step 4.

Step 6: Produce the new population of the LJaya-TVAC algorithm using (18) (the second phase).

Step 7: Calculate the objective function $f(r,n)$ of RRAP by imposing the non-linear constraints $g_1(r,n)$, $g_2(r,n)$ and $g_3(r,n)$ for the generated population in Step 6.

Step 8: Repeat Steps 4-7 till (for i=1: i_{max}) reaching the maximum number of iterations.

The best results obtained from the proposed LJaya-TVAC algorithm compared with previously reported results for RRAPs for five test systems over 30 runs with i_{max} =d*1000, and the population size of N=4*d, (d=2*m), are summarized in Tables 2-6. The best solution values are in bold including $f\left(r,n\right)$ (the obtained best value of the objective function), Mean (the mean value of the best results), Worst (the obtained worst value of the objective function), Std (the standard deviation of the best results) indexes and the maximum possible improvement (MPI) index (dos Santos Coelho 2009). The LJaya-TVAC algorithm is compared with previously reported best results (He et al. 2015). The MPI index is defined as follows (Valian et al. 2013):

$$\begin{aligned}&\mathrm{MPI}\left(\%\right)=\frac{f_{\text{LJaya-TVAC}}\left(r,n\right)-f_{\text{other}}\left(r,n\right)}{1-f_{\text{other}}\left(r,n\right)},\\&f_{\text{LJaya-TVAC}}\left(r,n\right):\text{the best results obtain by LJaya-TVAC algorithm,}\\&f_{\text{other}}\left(r,n\right):\text{the best results reported by previous studies.}\end{aligned} \qquad (19)$$

Also, slack (g) = l-g, for example slack (g_1) = V- $g_1\left(r,n\right)$.

The best result of the series test system by LJaya-TVAC algorithm shown in Table 2 is 0.931682388 which is compared with previously reported results including PSSO (Huang 2015), IA (Chen 2006), IPSO (Wu et al. 2011), a new IA (NIA) (Hsieh and You 2011), AR-ICA (Afonso, Mariani, and dos Santos Coelho 2013), and Improved Cuckoo Search ICS (Valian et al. 2013). The proposed LJaya-TVAC algorithm obtains the best solutions among all the solutions including $f_1\left(r,n\right)$, Mean, Worst and Std indexes as shown in Table 2, with the obtained results of 0.931682386, 0.9316823797 and 8.15e-22 respectively. It can be seen from Table 2 that, for PSSO (Huang 2015), IA (Chen 2006), IPSO (Wu et al. 2011), NIA (Hsieh and You 2011), AR-ICA (Afonso, Mariani, and dos Santos Coelho 2013), ICS (Valian et al. 2013), the corresponding improvements (MPI) made by the LJaya-TVAC algorithm are 1.288e-4, 6.423e-3, 3.554e-3, 7.026e-5, 4.388e-2, and 1.464e-6, respectively.

Table 2. Comparison of the best results obtained by LJaya-TVAC with some of the previously reported results for the series test system.

Parameter	PSSO	IA	IPSO	NIA	AR-ICA	ICS	**LJaya-TVAC**
r_1	0.77946645	0.779266	0.78037307	0.779462304	0.779874	0.779416938	0.779402388
r_2	0.87173278	0.872513	0.87178343	0.871883456	0.872057	0.871833278	0.871835465
r_3	0.90284951	0.902634	0.90240890	0.902800879	0.903426	0.902885082	0.902882077
r_4	0.71148780	0.710648	0.71147356	0.711350168	0.710960	0.711393868	0.711408035
r_5	0.78781644	0.788406	0.78738760	0.787861587	0.786902	0.787803712	0.787793007
n_1	3	3	3	3	3	3	3
n_2	2	2	2	2	2	2	2
n_3	2	2	2	2	2	2	2
n_4	3	3	3	3	3	3	3
n_5	3	3	3	3	3	3	3
$f_1(r,n)$	0.93168230	0.931678	0.93167996	0.93168234	0.93167939	0.931682387	**0.931682388**
Slack (g_1)	-	27	27	27	27	27	27
Slack (g_2)	-	0.001559	0.000101	0.0000005284	0.000099	0.000000265	2.19e-08
Slack (g_3)	-	7.518918	7.518918	7.518918	7.518918	7.518918241	7.51891824
MPI (%)	**1.288e-4**	**6.423e-3**	**3.554e-3**	**7.026e-5**	**4.388e-2**	**1.464e-6**	**-**
Mean	0.871793835	-	0.92847132	0.93168222	0.92182324	0.92987132	**0.931682386**
Worst	0.64815102	-	0.91011333	-	0.82989353	0.92066034	**0.9316823797**
Std	0.055331848	-	5.2382e−03	1.3e-14	0.01863188	1.99046e-03	**8.15e-22**

The best results for the series–parallel test system by LJaya-TVAC algorithm compared with ICS (Valian et al. 2013), MPSO (Liu and Qin 2014), IPSO (Wu et al. 2011), NIA (Hsieh and You 2011), AR-ICA (Afonso, Mariani, and dos Santos Coelho 2013), NAFSA (He et al. 2015) are shown in Table 3. The MPSO, NAFSA and proposed LJaya-TVAC obtain the better results than the other algorithms. The best values obtained by ICS, MPSO, IPSO, NIA, AR-ICA, NAFSA are 0.999976649, 0.9999766491, 0.99997664, 0.999976649, 0.99997661 and0.9999766491 respectively. Also, LJaya-TVAC obtains the better indexes in terms of Mean, Worst and Std than all other algorithms, with the obtained results of 0.9999766491, 0.99997664904 and 8.15e-25 respectively.

Table 3. Comparison of the best results obtained by LJaya-TVAC with some of the previously reported results for the series–parallel test system.

Paramete r	ICS	MPSO	IPSO	NIA	AR-ICA	NAFSA	LJaya-TVAC
r_1	0.819927087	0.8196547522	0.81918526	0.819591561	0.82201264	0.819737753	0.819659132
r_2	0.845267657	0.8449752789	0.84366421	0.844951068	0.84365640	0.844991099	0.844980808
r_3	0.895491554	0.8955087772	0.89472992	0.895428548	0.89129092	0.895529543	0.895506189
r_4	0.895440692	0.8955091117	0.89537628	0.895522339	0.89869886	0.895433687	0.895506537
r_5	0.868318775	0.8684491638	0.86912724	0.868490229	0.86824939	0.868434824	0.868447819
n_1	2	2	2	2	2	2	2
n_2	2	2	2	2	2	2	2
n_3	2	2	2	2	2	2	2
n_4	2	2	2	2	2	2	2
n_5	4	4	4	4	4	4	4
$f_2(r,n)$	0.999976649	**0.9999766491**	0.99997664	0.999976649	0.99997661	**0.9999766491**	**0.9999766491**
Slack (g_1)	40	40	40	40	40	40	40
Slack (g_2)	0.0000161	8.4e-9	0.000561	0.0	0.000396	1.39152e-10	7.959e-010
Slack (g_3)	1.6092890	1.6092889667	1.609289	1.609289	1.609289	1.609288966	1.6092889667
MPI (%)	**4.282e-04**	**0**	**3.896e-02**	**4.282e-04**	**1.672e-01**	0	-
Mean	0.99997090	0.9999766174	0.99996974	0.999976649	0.99994991	0.9999766490	**0.9999766491**
Worst	0.99994886	0.9999765280	0.99994106	-	0.99984762	-	**0.99997664904**
Std	4.45e-06	3.87e-08	1.336e−05	3.0e-21	2.58e-06	3.18206e-10	**8.15e-25**

Also, the best results for the complex (bridge) test system and overspeed protection system by different algorithms are summarized in Tables 4-5. The comparison of the best results shows that the proposed LJaya-TVAC obtains the better results than all the other algorithms. It can be seen from Table 4 for complex (bridge) test system that, for IA (Chen 2006), EGHS (Zou et al. 2011), IPSO (Wu et al. 2011), AR-ICA (Afonso, Mariani, and dos Santos Coelho 2013), PSFS (Mellal and Zio 2016), NAFSA (He et al. 2015), the corresponding improvements (MPI) made by the LJaya-TVAC algorithm are 3.858e-01, 3.398e-02, 6.815e-03, 6.815e-03, 1.087e-05, and 1.370e-03, respectively.

Also, it can be seen from Table 5 for the overspeed protection system test system that, for IA (Chen 2006), EGHS (Zou et al. 2011), PSSO (Huang 2015), AR-ICA (Afonso, Mariani, and dos Santos Coelho 2013), NAFSA (He et al. 2015), GA-PSO (Sheikhalishahi et al. 2013), the corresponding improvements (MPI) made by the LJaya-TVAC algorithm are 21.853, 9.847e-02, 1.03e-02, 3.699e-03, 1.496e-05, and 1.03e-02, respectively.

Table 4. Comparison of the best results obtained by LJaya-TVAC with some of the previously reported results for the complex (bridge) test system.

Parameter	IA	EGHS	IPSO	AR-ICA	PSFS	NAFSA	LJaya-TVAC
r_1	0.812485	0.82983999	0.82868361	0.82764257	0.82812141729	0.82832179189	0.828081997
r_2	0.867661	0.85798911	0.85802567	0.85747845	0.85781341076	0.85797450730	0.857823532
r_3	0.861221	0.91333926	0.91364616	0.91419677	0.91423927822	0.91422098825	0.914227868
r_4	0.713852	0.64674479	0.64803407	0.64927379	0.64807680660	0.64775717018	0.648117404
r_5	0.756699	0.70310972	0.70227595	0.70409200	0.70424641245	0.70300666185	0.70436276
n_1	3	3	3	3	3	3	3
n_2	3	3	3	3	3	3	3
n_3	3	2	2	2	2	2	2
n_4	3	4	4	4	4	4	4
n_5	1	1	1	1	1	1	1
$f_3(r,n)$	0.99988921	0.99988960	0.99988963	0.99988963	0.99988963751	0.99988963601	**0.999889637522**
Slack (g_1)	19	5	5	5	5	5	5
Slack (g_2)	0.001494	0.00000594	0.00000359	0.00004428	2.85464e-6	1.5485e-5	2.960e-06
Slack (g_3)	4.264770	1.56046629	1.56046629	1.56046629	1.560466288	1.56046629	1.560466288
MPI (%)	**3.858e-01**	**3.398e-02**	**6.815e-03**	**6.815e-03**	**1.087e-05**	**1.370e-03**	-
Mean	-	0.99988263	0.99988799	0.99979532	-	0.99987756441	**0.99988963752**
Worst	-	0.99982887	0.99970178	0.99939296	-	-	**0.999889637513**

Std	-	1.6e-05	4.0163e-05	1.037e-04	**3.7e-023**	2.1017e-05	8.16e-020

Table 5. Comparison of the best results obtained by LJaya-TVAC with some of the previously reported results for the overspeed protection system.

Parameter	IA	EGHS	PSSO	AR-ICA	NAFSA	GA-PSO	LJaya-TVAC
r_1	0.903800	0.900925066	0.90166461	0.90148988	0.90160779120	0.901628	0.901614807
r_2	0.874992	0.851636929	0.88817296	0.85003526	0.84993077684	0.888230	0.849921181
r_3	0.919898	0.948079849	0.94821033	0.94812952	0.94814603278	0.948121	0.948141393
r_4	0.890609	0.887654500	0.84987084	0.88823833	0.88821809379	0.849921	0.888222817
n_1	5	5	5	5	5	5	5
n_2	5	6	5	5	6	5	6
n_3	5	4	4	4	4	4	4
n_4	5	5	6	5	6	6	5
$f_4(r,n)$	0.999942	0.99995463	0.99995467	0.999954673	0.99995467467	0.99995467	**0.999954674676782**
Slack (g_1)	50	55	-	55	55	55	55
Slack (g_2)	0.002152	0.00000105	-	0.00213782	4.5195e-07	0.000006	1.614e-10
Slack (g_3)	28.803701	24.80188272	-	24.8018827	24.802	15.363463	24.8018827
MPI (%)	**21.853**	**9.847e-02**	**1.03e-02**	**3.699e-03**	**1.496e-05**	**1.03e-02**	-
Mean	-	0.99993588	0.9999416669	0.99993804	0.99995075542	0.99995467	**0.99995467467678**
Worst	-	0.99985315	0.99986938	0.99982276	-	0.99995467	**0.999954674676778**
Std	-	2.2e-05	1.61e-5	0.00002204	4.43e-06	1.0e-16	**4.86e-32**

Table 6 shows that the best solution for a large-scale test system is related to the proposed LJaya-TVAC which is compared with the solution reported by PSFS (Mellal and Zio 2016). The best results are 0.89051730902 and 0.891136424677689 by PSFS (Mellal and Zio 2016) and the proposed algorithm, respectively, and the result provided by the LJaya-TVAC is better than PSFS (Mellal and Zio 2016). The results show that the LJaya-TVAC algorithm is very reliable for the real large-scale optimization problems. In Table 6, for the best result obtain by the PSFS (Mellal and Zio 2016), the corresponding improvement (MPI) made by the LJaya-TVAC algorithm is 5.655e-01.

Table 6. Comparison of the best results obtained by LJaya-TVAC with some of the previously reported results for the large-scale test system.

Parameter	PSFS	LJaya-TVAC	Parameter	PSFS	LJaya-TVAC
r_1	0.920682125899	0.921084976	n_1	2	2
r_2	0.952579760087	0.952404213	n_2	2	2
r_3	0.840370879766	0.841155866	n_3	3	3
r_4	0.934499487329	0.935072836	n_4	2	2
r_5	0.806884188682	0.807515426	n_5	3	3
r_6	0.895206390582	0.894297746	n_6	2	2
r_7	0.811801524606	0.81181724	n_7	3	3
r_8	0.814181963158	0.813453019	n_8	3	3
r_9	0.836220223575	0.901645065	n_9	**3**	**2**
r_{10}	0.827983638973	0.825833294	n_{10}	3	3
r_{11}	0.814585208335	0.815100918	n_{11}	3	3
r_{12}	0.837346324449	0.838160321	n_{12}	3	3
r_{13}	0.841065088128	0.841043767	n_{13}	3	3
r_{14}	0.821075460589	0.820672214	n_{14}	3	3
r_{15}	0.850117124276	0.85096942	n_{15}	3	3
r_{16}	0.838491056144	0.838425251	n_{16}	3	3
r_{17}	0.823073269011	0.824306984	n_{17}	3	3
r_{18}	0.809956458845	0.810315221	n_{18}	3	3
r_{19}	0.807719250830	0.808690588	n_{19}	3	3
R_{20}	0.897948276899	0.833344431	n_{20}	**2**	**3**
Slack (g_1)	143	158	$f_5(r,n)$	0.89051730902	**0.891136424677689**
Slack (g_2)	0.00275018	1.453186e-04	Mean	-	**0.888020914198434**
Slack (g_3)	2.1970797	2.197079756	Std	3.5e-08	**7.15e-12**
Worst	-	**0.886359901275289**	MPI (%)	**5.655e-01**	-

It is obvious from Tables 1-6 that the proposed LJaya-TVAC algorithm can be useful and effective for optimization problems of engineering systems in comparison with the original Jaya algorithm and the best results reported in the recent literature by other algorithms.

6. Conclusion

In this paper, a hybrid enhanced Jaya algorithm based on TVAC and learning phase, called LJaya-TVAC has been proposed to efficiently solve various types of nonlinear mixed-integer RRAPs. The series, series–parallel, complex (bridge) and overspeed protection systems have been considered as RRAPs. The effectiveness of proposed LJaya-TVAC algorithm to achieve the optimal solutions of the standard real-parameter uni-modal and multi-model benchmark, as well as various RRAPs, was tested and compared with original Jaya algorithm and other optimal solutions reported in the recent literature.

In RRAPs, in addition to the reliability objective function, some other objective functions can be considered such as the overall cost (Ardakan, Hamadani, and Alinaghian 2015). The obtained optimal results of the paper have provided evidence for better and effective optimization performance of the proposed LJaya-TVAC algorithm in comparison with optimal solutions reported in the recent literature. Note that although the proposed algorithm is very reliable for the large-scale optimization problems in practice which is successfully tested for different systems in the paper, it adds some complexity to the simple Java algorithm. Addressing this problem is beyond the scope of this paper and would be the focus of our future work. Also in the future work, the proposed LJaya-TVAC algorithm can be considered for multi-objective optimization problems and compared with other algorithms such as NSGA-II and so on in the literature.

References

Afonso, Leonardo Dallegrave, Viviana Cocco Mariani, and Leandro dos Santos Coelho. 2013. "Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability-redundancy optimization." *Expert Systems with Applications* 40 (9):3794-802.

Ardakan Abouei, Mostafa, Mohammad Sima, Ali Zeinal Hamadani, and David W Coit. 2016. "A novel strategy for redundant components in reliability-redundancy allocation problems." *IIE Transactions* 48(11): 1043-1057.

Ardakan, Mostafa Abouei, and Ali Zeinal Hamadani. 2014a. "Reliability–redundancy allocation problem with cold-standby redundancy strategy." *Simulation Modelling Practice and Theory* 42:107-18.

Ardakan, Mostafa Abouei, and Ali Zeinal Hamadani. 2014b. "Reliability optimization of series–parallel systems with mixed redundancy strategy in subsystems." *Reliability Engineering & System Safety* 130:132-9.

Ardakan, Mostafa Abouei, Ali Zeinal Hamadani, and Mehdi Alinaghian. 2015. "Optimizing bi-objective redundancy allocation problem with a mixed redundancy strategy." *ISA transactions* 55:116-28.

Chatwattanasiri, Nida, David W Coit, and Naruemon Wattanapongsakorn. 2016. "System redundancy optimization with uncertain stress-based component reliability: Minimization of regret." *Reliability Engineering & System Safety* 154:73-83.

Chen, Ta-Cheng. 2006. "IAs based approach for reliability redundancy allocation problems." *Applied Mathematics and Computation* 182 (2):1556-67.

Chen, Ta-Cheng, and Peng-Sheng You. 2005. "Immune algorithms-based approach for redundant reliability problems with multiple component choices." *Computers in Industry* 56 (2):195-205.

Dohi, Tadashi, Naoto Kaio, Won Young, Ho-Gyun Kim, Chang-Ok Bae, and Dong-Jun Park. 2006. "Reliability-redundancy optimization using simulated annealing algorithms." *Journal of Quality in Maintenance Engineering* 12 (4):354-63.

dos Santos Coelho, Leandro. 2009. "An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications." *Reliability Engineering & System Safety* 94 (4):830-7.

dos Santos Coelho, Leandro, L de A Diego, and Viviana Cocco Mariani. 2011. "Reliability-Redundancy Optimization Using a Chaotic Differential Harmony Search Algorithm." In *Handbook of Swarm Intelligence*, 503-16. Springer.

Feizollahi, Mohammad Javad, Roya Soltani, and Hadi Feyzollahi. 2015. "The robust cold standby redundancy allocation in series-parallel systems with budgeted uncertainty." *IEEE Transactions on Reliability* 64 (2):799-806.

Hamadani, Ali Zeinal, Mostafa Abouei Ardakan, Taghi Rezvan, and Mohammad Mehran Honarmandian. 2013. "Location-allocation problem for intra-transportation system in a big company by using meta-heuristic algorithm." *Socio-Economic Planning Sciences* 47 (4):309-17.

He, Pan, Kaigui Wu, Jie Xu, Junhao Wen, and Zhuo Jiang. 2013. "Multilevel redundancy allocation using two dimensional arrays encoding and hybrid genetic algorithm." *Computers & Industrial Engineering* 64 (1):69-83.

He, Qiang, Xiangtao Hu, Hong Ren, and Hongqi Zhang. 2015. "A novel artificial fish swarm algorithm for solving large-scale reliability–redundancy application problem." *ISA transactions* 59:105-13.

Hsieh, Y-C, and P-S You. 2011. "An effective immune based two-phase approach for the optimal reliability–redundancy allocation problem." *Applied Mathematics and Computation* 218 (4):1297-307.

Hsieh, Yi-Chih, Ta-Cheng Chen, and Dennis L Bricker. 1998. "Genetic algorithms for reliability design problems." *Microelectronics Reliability* 38 (10):1599-605.

Huang, Chia-Ling. 2015. "A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems." *Reliability Engineering & System Safety* 142:221-30.

Jang, Kil-Woong, and Jae-Hwan Kim. 2011. "A tabu search for multiple multi-level redundancy allocation problem in series-parallel systems." *International Journal of Industrial Engineering: Theory, Applications and Practice* 18 (3).

Li, Zhaojun, Mohammadsadegh Mobin, and Thomas Keyser. 2016. "Multi-Objective and Multi-Stage Reliability Growth Planning in Early Product-Development Stage." *IEEE Transactions on Reliability* 65 (2):769-81.

Liang, Yun-Chia, and Alice E Smith. 2004. "An ant colony optimization algorithm for the redundancy allocation problem (RAP)." *IEEE Transactions on Reliability* 53 (3):417-23.

Liu, Gia-Shie. 2006. "A combination method for reliability-redundancy optimization." *Engineering Optimization* 38 (04):485-99.

Liu, Yubao. 2016. "Improved Bat Algorithm for Reliability-Redundancy Allocation Problems." *International Journal of Security and Its Applications* 10 (2):1-12.

Liu, Yubao, and Guihe Qin. 2014. "A hybrid TS-DE algorithm for reliability redundancy optimization problem." *Journal of Computers* 9 (9):2050-7.

Mellal, Mohamed Arezki, and Enrico Zio. 2016. "A penalty guided stochastic fractal search approach for system reliability optimization." *Reliability Engineering & System Safety* 152:213-27.

Ouzineb, Mohamed, Mustapha Nourelfath, and Michel Gendreau. 2011. "A heuristic method for non-homogeneous redundancy optimization of series-parallel multi-state systems." *Journal of Heuristics* 17 (1):1-22.

Pourdarvish, Ahmad, and Zahra Ramezani. 2013. "Cold standby redundancy allocation in a multi-level series system by memetic algorithm." *International Journal of Reliability, Quality and Safety Engineering* 20 (03):1340007.

Rao, R. 2016. "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems." *International Journal of Industrial Engineering Computations* 7 (1):19-34.

Rao, R, and Vivek Patel. 2012. "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems." *International Journal of Industrial Engineering Computations* 3 (4):535-60.

Rao, R Venkata. 2015. *Teaching Learning Based Optimization Algorithm: And Its Engineering Applications*: Springer.

Rao, R Venkata, Dhiraj P Rai, and Joze Balic. 2016. "Surface Grinding Process Optimization Using Jaya Algorithm." In *Computational Intelligence in Data Mining*, *Springer* 487-495

Rao, Ravipudi V, Vimal J Savsani, and DP Vakharia. 2011. "Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems." *Computer-Aided Design* 43 (3):303-15.

Rao, RV, KC More, J Taler, and P Ocłoń. 2016. "Dimensional optimization of a micro-channel heat sink using Jaya algorithm." *Applied Thermal Engineering* 103:572-82.

Ratnaweera, Asanga, Saman K Halgamuge, and Harry C Watson. 2004. "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients." *IEEE Transactions on evolutionary computation* 8 (3):240-55.

Sadjadi, Seyed Jafar, and Roya Soltani. 2012. "Alternative design redundancy allocation using an efficient heuristic and a honey bee mating algorithm." *Expert Systems with Applications* 39 (1):990-9.

Sheikhalishahi, M, V Ebrahimipour, H Shiri, H Zaman, and M Jeihoonian. 2013. "A hybrid GA–PSO approach for reliability optimization in redundancy allocation problem." *The International Journal of Advanced Manufacturing Technology* 68 (1-4):317-38.

Soltani, Roya. 2014. "Reliability optimization of binary state non-repairable systems: A state of the art survey." *International Journal of Industrial Engineering Computations* 5 (3):339.

Soltani, Roya, Seyed Jafar Sadjadi, and Reza Tavakkoli-Moghaddam. 2015. "Entropy based redundancy allocation in series-parallel systems with choices of a redundancy strategy and component type: A multi-objective model." *Appl. Math* 9 (2):1049-58.

Suganthan, Ponnuthurai N, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. 2005. "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization." *KanGAL report*.

Suman, Balram. 2003. "Simulated annealing-based multiobjective algorithms and their application for system reliability." *Engineering Optimization* 35 (4):391-416.

Tan, Yue, Guan-zheng Tan, and Shu-guang Deng. 2013. "Hybrid particle swarm optimization with differential evolution and chaotic local search to solve reliability-redundancy allocation problems." *Journal of Central South University* 20:1572-81.

Valian, Ehsan, Saeed Tavakoli, Shahram Mohanna, and Atiyeh Haghi. 2013. "Improved cuckoo search for reliability optimization problems." *Computers & Industrial Engineering* 64 (1):459-68.

Valian, Ehsan, and Elham Valian. 2013. "A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems." *Engineering Optimization* 45 (11):1273-86.

Venkata Rao, R, and GG Waghmare. 2017. "A new optimization algorithm for solving complex constrained design optimization problems." *Engineering Optimization* 49 (1): 60-83.

Wu, Peifeng, Liqun Gao, Dexuan Zou, and Steven Li. 2011. "An improved particle swarm optimization algorithm for reliability problems." *ISA transactions* 50 (1):71-81.
Yeh, Wei-Chang, and Tsung-Jung Hsieh. 2011. "Solving reliability redundancy allocation problems using an artificial bee colony algorithm." *Computers & Operations Research* 38 (11):1465-73.
Zhang, HongQi, XiangTao Hu, XiaoDong Shao, ZiCheng Li, and YuHui Wang. 2013. "IPSO-based hybrid approaches for reliability-redundancy allocation problems." *Science China Technological Sciences* 56 (11):2854-64.
Zou, Dexuan, Liqun Gao, Steven Li, and Jianhua Wu. 2011. "An effective global harmony search algorithm for reliability problems." *Expert Systems with Applications* 38 (4):4642-8.
Zou, Dexuan, Liqun Gao, Jianhua Wu, Steven Li, and Yang Li. 2010. "A novel global harmony search algorithm for reliability problems." *Computers & Industrial Engineering* 58 (2):307-16.

Appendix. Data of the test systems.

Table A.1. Data of the series system (*P*1).

Stage	$10^5\times \alpha_d$	β_d	$w_d v_d^2$	w_d	V	C	W
1	1.0	1.5	1	6	250	400	500
2	2.3	1.5	2	6			
3	0.3	1.5	3	8			
4	2.3	1.5	2	7			

Table A.2. Data of the series–parallel system (*P*2).

Stage	$10^5\times \alpha_d$	β_d	$w_d v_d^2$	w_d	V	C	W
1	2.500	1.5	2	3.5	180	175	100
2	1.450	1.5	4	4.0			
3	0.541	1.5	5	4.0			
4	0.541	1.5	8	3.5			
5	2.100	1.5	4	4.5			

Table A.3. Data of the complex (bridge) system (*P*3).

Stage	$10^5\times \alpha_d$	β_d	$w_d v_d^2$	w_d	V	C	W

1	2.330	1.5	1	7	180	175	200
2	1.450	1.5	2	8			
3	0.541	1.5	3	8			
4	8.050	1.5	4	6			
5	1.950	1.5	2	9			

Table A.4. Data of the fourth test system (*P*4).

Stage	$10^5\times\alpha_d$	β_d	$w_d v_d^2$	w_d	V	C	W	T
1	1.0	1.5	1	6	250	400	500	1000 *h*
2	2.3	1.5	2	6				
3	0.3	1.5	3	8				
4	2.3	1.5	2	7				