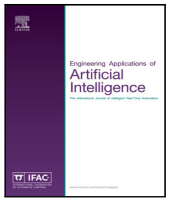
Engineering Applications of Artificial Intelligence 119 (2023) 105778

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



A novel dynamic opposite learning enhanced Jaya optimization method for high efficiency plate–fin heat exchanger design optimization

Lidong Zhang [a], Tianyu Hu [b], Linxin Zhang [c], Zhile Yang [b], Seán McLoone [d], Muhammad Ilyas Menhas [e], Yuanjun Guo [b,*]

[a] Northeast Electric Power University, Jilin 132012, China
[b] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, 518055, China
[c] Faculty of Data Science, City University of Macau, Taipa, 999078, Macau, China
[d] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Northern Ireland, UK
[e] Department of Electrical Engineering, Mirpur University of Science and Technology, Mirpur, Pakistan

ARTICLE INFO

Keywords:
Plate–fin heat exchangers
Jaya algorithm
Optimal design
Dynamic-opposite learning

ABSTRACT

A Plate–fin heat exchanger (PFHE) is a compact and efficient thermal device, whose performance strongly depends on its structural design. However, the design optimization of a PFHE is a mixed-integer optimization problem with a strong nonlinear characteristic, which presents significant challenges for existing optimization algorithms. Meta-heuristic algorithms (MAs) are competitive for solving complex non-linear optimization problems. In this paper, an improved dynamic-opposite learning Jaya (DOLJaya) method, the goal is to make the algorithm adaptable to each problem. The results of eighteen unimodal and multi-modal benchmarks and nine hybrid benchmarks demonstrate that the proposed DOLJaya has competitive robustness, efficiency and effectiveness for solving complex nonlinear problems compared to its popular counterparts. At the same time, we selected the optimization of the plate–fin heat exchanger as the industrial test benchmark for optimization, and the results of DOLJaya algorithm have been improved by a maximum average of 108.29% and 7.60% compared with the original Jaya, which are also satisfactory.

1. Introduction

A Plate–fin heat exchanger (PFHE) is a compact and featured heat exchanger design, which has been widely used for industrial and household applications. Its key characteristics are large heat transfer area, small volume and lightweight construction. Energy efficiency is the foremost issue in PFHE applications. At present, the majority of heat exchanger design methods are qualitative or trial-and-error in nature, and there is no unified or recognized design scheme (Wang and Li, 2016b) which has been widely adopted. To address this, PFHE optimal design problems have been proposed and solved using new methods such as heuristic-based optimization algorithms (Babu and Munawar, 2007; Xie et al., 2008). Meta-heuristic algorithms (MAs) are optimization techniques that are inspired by behaviours and connections in nature, especially those relating to biological populations. Some well-known MAs include genetic algorithm (GA) (Goldberg, 2006), particle swarm optimization (PSO) (Del Valle et al., 2008), differential evolution (DE) (Storn and Price, 1997), inclined planes system optimization (IPO) (Mozaffari et al., 2016), gravitational search algorithm (GSA) (Rashedi et al., 2009) and salp swarm algorithm (SSA) (Çelik et al., 2021). In addition, popular recently proposed MAs include ant lion optimizer (ALO) (Mirjalili, 2015a), moth-flame optimization algorithm (MFO) (Mirjalili, 2015b), whale optimization algorithm (WOA) (Mirjalili and Lewis, 2016), Jaya algorithm (Rao, 2016) and sine cosine algorithm (SCA) (Mirjalili, 2016). The evolution of MAs and their variants were developed over the years, like Micro GA, Cellular GA, NSGA, Contextual GA, Grouping GA, Quantum-inspired GA, Linkage learning GA, Island GA, non-dominated sorting genetic algorithm II (NSGA-II), Interactive GA, Jumping gene GA, Dynamic rule-based GA, Hierarchical cellular GA, NSGAIII, Tribe competition-based GA, Fluid GA, Block-based GA, etc. And the variants of DE like Self-adaptive DE, jDE, Chaotic DE, adaptive DE with optional external archive (JADE), ensemble of mutation strategies DE (EPSDE), Composite DE (CoDE), Multi-population DE, Adaptive Cauchy DE (ACDE), improved JADE, Extended adaptive Cauchy DE, jDErpo, Restart DE algorithm with Local search mutation, Colonial competitive DE, Memory-based DE, Stochastic Quasi-Gradient (SQG)-DE, Unified DE (UDE), Opposition-based Compound Sinusoidal DE (OCSinDE), etc. (Janga Reddy and Nagesh Kumar, 2020; Çelik, 2020b; Rashedi et al., 2010; Yılmaz et al., 2021) MAs have been employed to solve a number of energy-related problems, including heat exchanger optimization (Barros et al., 2018), damage detection (Tran-Ngoc et al., 2021;

* Corresponding author.
E-mail address: yj.guo@siat.ac.cn (Y. Guo).

https://doi.org/10.1016/j.engappai.2022.105778
Received 16 December 2021; Received in revised form 5 December 2022; Accepted 23 December 2022
Available online xxxx
0952-1976/

Nomenclature

ρ	Density, kg m^{-3}
T	temperature, K
ΔP	Pressure drop, N m^{-2}
ϵ	effectiveness
μ	Dynamic viscosity, N s^{-1} m^{-2}
A, A_{HT}	Heat transfer area, m^2
A_{ff}	Free flow area, m^2
C	Heat capacity rate, W K^{-1}
c	Cold stream
Cp	Specific heat of fluid, W kg^{-1} K^{-1}
D_H	Hydraulic diameter, m
e	Euler's number
f	Fanning friction factor
H	fin Height, m
h	Hot stream
i	Inlet
j	Colburn factor
L	Heat exchanger length, m
l	fin lance length, m
m	Mass flow rate of fluid, kg s^{-1}
max	Maximum
min	Minimum
N	Number of fin layers
n	Number of fins
N_s	Number of entropy generation units
NTU	Number of transfer units
Pr	Prandtl number
Q	Heat duty, W
Q^*	Desired/fixed heat duty, W
R	Specific gas constant, J kg^{-1} K^{-1}
Re	Reynolds number
s	Fin spacing, m
t	Fin thickness, m

Khatir et al., 2019, 2020), unit commitment problems (Mallipeddi and Suganthan, 2014; Yang et al., 2017, 2019a), environmental or economic dispatch problems (Qu et al., 2018; Yang et al., 2019b), automatic generation control of power system (Çelik, 2020a,b), combined cooling heating and power system optimization (Li et al., 2020), battery state-of-charge estimation (Guo et al., 2021; Liu et al., 2018) optimal siting and sizing of distributed generation (Saboori et al., 2017) and the optimization of heat exchanger networks (Mano et al., 2017).

Common heat exchanger design indexes include size, weight, cost and heat load. Two performance indexes are widely adopted: logarithmic mean temperature difference (LMTD) and effectiveness number of transfer units (ϵ-NTU) (Kakac et al., 2020). Selbaş et al. (2006) used GA to optimize the design of shell-and-tube heat exchanger (STHE), and selected the heat exchanger area as the optimization target. Babu and Munawar (2007) addressed the heat transfer area as the optimization target and adopted DE to optimize their STHE design. Fesanghary et al. (2009) used harmony search algorithm (HSA) with a GA to optimize the STHE design, with the total cost of heat exchanger operation selected as the target. Patel and Rao (2010) set the total annual cost as the optimization goal and adopted a PSO algorithm as the solver. Yousefi et al. (2012) used the imperialist competitive algorithm (ICA) to optimize the total annual cost and total weight of a PFHE design. Rao and Patel (2013) chose the TLBO algorithm to optimize the heat exchanger efficiency and total cost. Hadidi (2015) selected the total cost, heat exchange area, total pressure drop and the geometric size of the heat exchanger as the evaluation index, and used the biogeography-based optimization (BBO) algorithm for PFHE design. Wang and Li (2015) utilized improved multi-objective cuckoo search (IMOCS) for PFHE design optimization, with efficiency of pumping power and the total annual cost as their optimization goals. The NSGA-II is adopted in Wong et al. (2016) to optimize the heat recovery and total cost of a STHE. In 2021, Zhang et al. (2021b) used the elite and dynamic opposite learning enhanced sine cosine algorithm (EDOLSCA) to optimize PFHEs. Though numerous approaches have been adopted for solving optimal heat exchanger design problems, the complex evaluation index and constraints make it a very challenging optimization problem, and one where there is scope for more effective computational methods to be developed to achieve better solutions.

The Jaya algorithm (Rao, 2016) was first proposed by Rao and has been widely used in various engineering scenarios. Bhoye et al. (2016) used it to solve the combined economic emission dispatch problem in distributed energy resources management systems. Rao et al. (2016) applied Jaya to solve a surface grinding problem in a machining process and also to solve a STHE design optimization problem considering consistency, maintenance and fouling (Rao and Saroj, 2017). hlOther applications of the Jaya algorithm include optimal power flow in the integration of distributed generation (Warid et al., 2016), automatic generation control in multi-area interconnected power systems (Singh et al., 2017), environmental dispatch of distributed energy resources in a microgrid (Trivedi et al., 2016) and design optimization of truss structures taking sizing, layout and topology into account (Zhang et al., 2021a; Zhao et al., 2021; Kaveh et al., 2021; Degertekin and Tutar, 2022; Belagoune et al., 2022).

Building on the original standard Jaya algorithm, many variants have emerged in recent years to solve specific optimization problems. Rao and More (2017) proposed a self-adaptive Jaya algorithm. Huang et al. (2017) proposed a spline-guided Jaya algorithm (S-Jaya) to solve a maximum power point tracking (MPPT) problem. Farah and Belazi (2018) developed a chaotic Jaya algorithm (C-Jaya) and proved its advantage with regard to global convergence. Gao et al. (2018) proposed a discrete Jaya algorithm (DJaya) for solving a flexible job-shop rescheduling problem (FJRP), and demonstrated its efficacy through a comparative study with five other counterparts. Rao and Saroj (2019) proposed the elitist-based self-adaptive multi-population (SAMPE) Jaya algorithm (SAMPE-Jaya) and used it to solve the micro-channel heat sink design optimization problem. Yu et al. (2019) introduced the self-adaptive multi-population and Lévy flights Jaya algorithm (Jaya-SML) to solve economic load dispatch problems (ELDPs). In addition, there are also enhanced chaotic Jaya algorithm, self-learning discrete Jaya algorithm, etc., which have been intensively proposed in the past two years (Premkumar et al., 2021; Zhao et al., 2021). While many variations of the Jaya algorithm have been proposed, they are more suited to small scale and symmetrical constraints problems than to the challenging mixed-integer combination optimization problem that arises with PFHE design optimization, hence their effectiveness and robustness needs to be improved for this application.

The purpose of this study is to propose a new Jaya variant named DOLJaya that adopts the dynamic opposite learning (DOL) strategy first proposed by Xu et al. (2020) to solve the optimal PFHE design problem, with significantly improved stability, robustness and global search capability when compared to the original Jaya algorithm, and other competing MAs. The Jaya algorithm itself is a simple and effective solution strategy that can simultaneously approach the best and stay away from the worst solutions. However, the diversity of the algorithm solution is not adequately guaranteed. The DOL scheme is a dynamic solution generation method which can significantly enhance solution diversity, hence has the potential to improve the performance of the Jaya algorithm. The key contributions of this paper are as follows:

(1) A thermodynamic optimization model of PFHEs based on the NTU method is proposed, where fluid data and the constraints of two common design cases are considered.

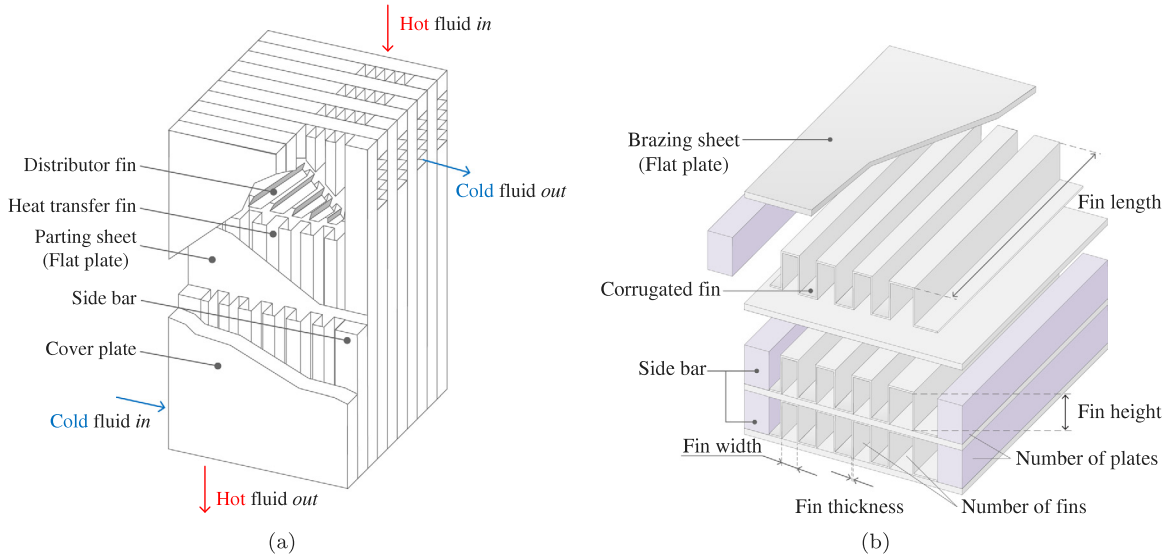


Fig. 1. Plate–fin heat exchanger and rectangular offset strip fin.

(2) A novel DOLJaya algorithm is proposed to improve on the exploitation ability of the original Jaya algorithm.
(3) Experimental studies have been conducted on numerical benchmark functions and the PFHE design problem to demonstrate the effectiveness of the proposed DOLJaya algorithm.

2. Problem formulation

The optimal design of a plate–fin heat exchanger is an important and difficult engineering optimization problem. It is a mixed-integer NP-hard problem and presents significant challenges for existing optimization methods. A typical two-stream plate–fin heat exchanger is shown in Fig. 1. The fin shape is considered as a straight fin in this paper. PFHE design optimization can be formulated as a multi-dimensional nonlinear optimization problem, where the fin geometry and fin density of the heat exchanger are the design (optimization) variables, and entropy generation units (Ns), number of transfer units (NTU) and heat transfer area (A_{HT}) are the heat exchanger performance indexes to be optimized.

2.1. Mathematical models for the plate–fin heat exchanger

Firstly, the heat duty of the heat exchanger is defined in (Yousefi et al., 2013):

$$Q = \epsilon C_{min}\left(T_{hi} - T_{ci}\right) \tag{1}$$

where T_{hi} and T_{ci} are the hot and cool fluid inlet temperatures, respectively, and C_{min} is the heat capacity rate, defined as:

$$C_{min} = min(m_h Cp_h, m_c Cp_c) \tag{2}$$

where subscripts h and c denote the hot and cool fluid, respectively, m is the fluid mass flow rate, and Cp is its specific heat value. Thus, C_{min} is the heat capacity rate of the fluid with the lowest heat capacity rate. The effectiveness ϵ is obtained from (Incropera et al., 2007),

$$\epsilon = 1 - e^{\left(\left(\frac{1}{C_r}\right)NTU^{0.22}\left(e^{\left(-C_r NTU^{0.78}\right)}-1\right)\right)} \tag{3}$$

where

$$C_r = C_{min}/C_{max}, \quad C_{max} = max(m_h Cp_h, m_c Cp_c) \tag{4}$$

The heat transfer coefficient is then expressed as:

$$\frac{1}{NTU} = C_{min}\left(\frac{A_{ffh}}{j_h Cp_h Pr_h^{-0.667} m_h A_h} + \frac{A_{ffc}}{j_c Cp_c Pr_c^{-0.667} m_c A_c}\right) \tag{5}$$

Here, Pr, j, A_{ff} and A are the Prandtl number, Colburn factor, fluid free flow area, and heat transfer area for the hot (subscript h) and cold (subscript c) fluids, respectively, and the other parameters are as previously defined. The fluid flow areas are given by:

$$A_{ffh} = \left(H_t - t_h\right)\left(1 - n_h t_h\right) L_c N_h \tag{6}$$

$$A_{ffc} = \left(H_c - t_c\right)\left(1 - n_c t_c\right) L_h N_c \tag{7}$$

where N_h and N_c are the number of hot and cold fin fluid layers, respectively, with

$$N_c = N_h + 1 \tag{8}$$

Parameters H, t, n and L are the fin height, fin thickness, number of fins, and the length of the heat exchanger, respectively. The heat transfer areas can be expressed as (Yousefi et al., 2012):

$$A_h = L_h L_c N_h \left(1 + \left(2n_h\left(H_h - t_h\right)\right)\right) \tag{9}$$

$$A_c = L_h L_c N_h \left(1 + \left(2n_c\left(H_c - t_c\right)\right)\right) \tag{10}$$

The Colburn factor, j, and Fanning friction factor, f, Joshi and Webb (1987), are given by

$$j = 0.53(Re)^{-0.5}\left(\frac{l}{D_H}\right)^{-0.15}\left(\frac{s}{H} - t\right)^{-0.14} \tag{11}$$

$$f = 8.12(Re)^{-0.74}\left(\frac{l}{D_H}\right)^{-0.41}\left(\frac{s}{H} - t\right)^{-0.02} \tag{12}$$

for laminar flow ($Re \le 1500$), and by

$$j = 0.21(Re)^{-0.4}\left(\frac{l}{D_H}\right)^{-0.24}\left(\frac{s}{H} - t\right)^{0.02} \tag{13}$$

$$f = 1.12(Re)^{-0.36}\left(\frac{l}{D_H}\right)^{-0.65}\left(\frac{s}{H} - t\right)^{0.17} \tag{14}$$

for turbulent flow($Re > 1500$), where

$$Re = \frac{m \cdot D_H}{A_{ff}\mu} \tag{15}$$

is the Reynolds number. The fin spacing, s, of the heat exchanger is:

$$s = (1/n - t) \tag{16}$$

and for the fin shape considered in this paper, the hydraulic diameter D_H of the heat exchanger can be expressed as:

$$D_H = \frac{4s(H-t)l}{2(sl+(H-t)l+t(H-t))+ts} \tag{17}$$

where l is the fin lance length. Finally, the pressure drop caused by friction between the hot and cold fluids and the heat exchanger walls can be calculated as:

$$\Delta P_h = \frac{2f_h L_h \left(\frac{m_h}{Aff_h}\right)^2}{\rho_h D_{Hh}} \tag{18}$$

$$\Delta P_c = \frac{2f_c L_c \left(\frac{m_c}{Aff_c}\right)^2}{\rho_c D_{Hc}} \tag{19}$$

where ρ_h and ρ_c are the fluid densities, D_{Hh} and D_{Hc} are the hydraulic diameters of the hot fluid and cold fluid sections of the heat exchanger, respectively, and the other parameters are as previously defined.

2.2. Plate–fin heat exchanger optimal design objective function

During the working process of a heat exchanger, the majority of heat exchangers have the characteristic of increased entropy due to the existence of friction. Thus, when optimizing a heat exchanger design, it can be considered that the smaller the entropy generation units are, the better the heat exchanger performs. In light of this, the objective function of entropy generation units is defined as follows (Bejan, 2013):

$$N_s = (1-\epsilon)\left[\frac{(T_{ci}-T_{hi})^2}{T_{ci}T_{hi}}\right] + \left(\frac{R_h}{Cp_h}\right)\left(\frac{\Delta P_h}{P_{hi}}\right) + \left(\frac{R_c}{Cp_c}\right)\left(\frac{\Delta P_c}{P_{ci}}\right) \tag{20}$$

where R_h and R_c, and P_{hi} and P_{ci} are the specific gas constants and inlet pressures of the hot and cold fluids, respectively.

The heat transfer area of the plate–fin heat exchanger determines its heat load and also affects the Reynolds number, which is directly related to the heat transfer efficiency. Therefore, the heat transfer area is also an important evaluation criteria for heat exchangers. The total heat transfer area of the two fluids is given by:

$$A_{HT} = A_h + A_c \tag{21}$$

2.3. Plate–fin heat exchanger application example

The design of two different gas–gas cross-flow plate–fin heat exchangers are considered as case studies, one rated at 160 kW (Case 1) and one at 1050 kW (Case 2). The differences between them lie in their heat duty and number of fin layers. The thermal load requirement is realized by a penalty function constraint, which can be expressed as:

$$g(X) \Rightarrow \xi(X) - Q^* = 0 \tag{22}$$

where Q^* is the desired heat duty, $g(X)$ is from Eq. (1) and $\xi(X)$ is the objective function. Here, X denotes the optimization variables, and $g(X)$ represents Eqs. (20) and (21) respectively in the following two cases. $\xi(X)$ is calculated by Eq. (1) and changes over the course of optimization iterations. Q^*, the desired heat load of the heat exchanger, is a fixed design parameter, as given in Table 1.

The size of the heat exchangers is limited to 1 m × 1 m. The optimization variables, X, are the heat exchanger size, fin thickness, number of fluid layers, fin density and fin length, width and height. The heat exchanger fluids are considered as ideal gases. The evaluation targets for the optimization are the entropy production units and the heat transfer area. Specific fluid characteristics and heat exchanger design constraints are given in Table 1.

3. Algorithm preliminaries

3.1. Jaya algorithm

The majority of heuristic optimization algorithms utilize specific control parameters to adjust their performance, such as the inertia weight in PSO and mutation and crossover probability in GA. If these parameters are not reasonably adjusted, the algorithm may converge prematurely or yield poor performance. Some algorithms, such as TLBO, have eliminated the need for such special parameters, with only more general parameters such as the population size and the number of iterations to be set. The Jaya algorithm is a new representative of this category of meta heuristic method.

The Jaya algorithm was first proposed by Rao in 2016 (Rao, 2016). It combines within a single stage the two stages of teachers and students in TLBO, and, similar to TLBO, does not need to adjust specific parameters. Jaya is designed to not only approach the best solution but also to actively move away from the worst solution. Its key update logic is defined as follows:

$$X'_{j,i,k} = X_{j,i,k} + r_{1,j,k}\left(X_{j,i,best} - \left|X_{j,i,k}\right|\right) \quad - r_{2,j,k}\left(X_{j,i,worst} - \left|X_{j,i,k}\right|\right) \tag{23}$$

where X is the variable, j is the dimension, k is the number of iterations, and i is the population size, $X_{j,i,k}$ is the value of the jth variable of the ith candidate in the kth iteration, r_1 and r_2 are random numbers distributed in the range of $[0, 1]$, $X_{j,i,best}$ and $X_{j,i,worst}$ represent the *best* and *worst* candidates for the jth variable in the kth iteration, and $|X_{j,i,k}|$ is the absolute value of the candidate. The term $+r_{1,j,k}(X_{j,i,best} - |X_{j,i,k}|)$ seeks to move variables towards the optimal solution, while the term $-r_{2,j,k}(X_{j,i,worst} - |X_{j,i,k}|)$ seeks to move them away from the worst solution.

3.2. Dynamic opposite learning

In order to improve the searching ability of meta-heuristic optimization algorithms, many learning strategies have been proposed. For population based optimization algorithms typical improvements include oppositional-based learning (OBL), quasi-opposite based learning (QOBL) (Rahnamayan et al., 2007), and quasi-reflection based learning (QRBL) (Ergezer et al., 2009).

The OBL strategy was first proposed by Rahnamayan et al. in 2008 (Rahnamayan et al., 2008) to speed up the convergence of the DE algorithm. The OBL strategy is based on using the opposite number to a given number X, that is:

$$X^O = a + b - X \tag{24}$$

where X^O is the opposite number of X, a and b are real numbers, $X \in [a, b]$. Letting X represent the candidate solution, j the jth variable in the problem, and D the dimension of the problem, Eq. (24) can be generalized to:

$$X_j^O = a_j + b_j - X_j, \quad j = 1 : D \tag{25}$$

However, OBL, QOBL and QRBL may be susceptible to local optima in their search space, which can significantly impact algorithm performance. To address this, Xu et al. (2020) proposed a dynamic opposite learning (DOL) based strategy to enhance diversity. The DOL strategy can be expressed as:

$$X^{DO} = X + wr_a(r_b X^O - X) \tag{26}$$

where r_a and r_b are random numbers in the range $[0, 1]$, and w is a positive weight to balance exploitation and exploration performance. The term $r_b X^O$ ensures the search space becomes asymmetric, which reduces the possibility of getting trapped in a local optimum. As a result, the search space becomes dynamic and changes with each iteration. In a similar fashion to Eq. (25), Eq. (26) can be generalized as:

$$X_j^{DO} = X_j + wr_{aj}(r_{bj} X_j^O - X_j), \quad j = 1 : D \tag{27}$$

Table 1
Design constraints and fluid data.

Parameters	Case 1, Q^* = 160 kW		Case 2, Q^* = 1000 kW	
	Hot fluid	Cold fluid	Hot fluid	Cold fluid
m (kg s^{-1})	0.8962	0.8296	1.66	2
T_i (K)	513	277	1173.15	473.15
P_i (Pa)	100 000	100 000	160 000	200 000
Cp (J kg^{-1} K^{-1})	1017.7	1011.8	1122	1073
ρ (kg m^{-3})	0.8196	0.9385	0.6296	0.9638
μ (N s m^{-2})	2.410E−05	2.182E−05	4.010E−05	3.360E−05
Pr	0.6878	0.6954	0.731	0.694
Parameters	Case 1, Q^* = 160 kW		Case 2, Q^* = 1000 kW	
	LowerBound	UpperBound	LowerBound	UpperBound
L_h (m)	0.1	1	0.1	1
L_c (m)	0.1	1	0.1	1
H (m)	0.002	0.01	0.002	0.01
n	100	1000	100	1000
t (m)	0.0001	0.0002	0.0001	0.0002
l (m)	0.001	0.01	0.001	0.01
N_h	1	10	1	200

4. DOL based Jaya algorithm

In this section the DOL strategy is combined with Jaya to arrive at a new algorithm, DOLJaya. Specifically, the DOL strategy is used to improve the population initialization step and generation jump step based on the Jaya principle.

4.1. DOL population initialization

Given a random initial population, oP, the DOL initialization step is given by:

$$oP_{ij}^{DO} = oP_{ij} + r_{1ij}(r_{2ij}(a_j + b_j - oP_{ij}) - oP_{ij}) \tag{28}$$

where i and j are indexes for the jth variable of the ith individual, a and b are the variable boundaries, and r_1 and r_2 are random numbers in the range $[0, 1]$. This enables the initial population to obtain better adaptability. The boundaries of the New population are checked and the $best$ and $worst$ individuals in the population are selected.

4.2. DOL generation jumping

The DOL strategy provides a mechanism for enhancing diversity. However, not every generation uses DOL policy updates. In the evolutionary process, a jumping rate (Jr) is introduced to control the option to adopt or ignore a DOL policy update. The value of Jr is within the range $[0, 1]$. During operation, a random number in the interval $[0, 1]$ is generated for each generation. If the random number is smaller than Jr, the DOL strategy is adopted. The generation jumping process for the DOL is as follows:

$$P_{ij}^{DO} = P_{ij} + wr_{3ij}(r_{4ij}(a_j + b_j - P_{ij}) - P_{ij}) \tag{29}$$

In order to reduce the scope of the search space at each iteration, the interval boundary can be updated as follows:

$$\begin{aligned} a_j &= min(P_{ij}) \\ b_j &= max(P_{ij}) \end{aligned} \tag{30}$$

Therefore, the $best$ and $worst$ individuals in the population can be selected. Parameters a_j and b_j represent the minimum and maximum values of the results of the j_{th} dimension in the current iteration of the population.

4.3. DOLJaya algorithm steps

The proposed Jaya variant involves several new steps, which are shown in pseudo code in Algorithm 1 and in the flow chart in Fig. 2.

4.4. Encoding and decoding

In the optimal PFHE design problem, it is important that the decision variables are appropriately encoded. Here, encoding is undertaken in accordance with Wang and Li (2016a) and Song and Cui (2019). There are 7 design variables in each step, and the evaluation functions are Eqs. (20) and (21). Eqs. (20) and (21) will be calculated as two application examples of the algorithm as two independent targets.

Firstly, for the two optimization goals under consideration, Eqs. (20) and (21), a population of N individuals is generated according to the algorithm step, with each individual encoding the seven design variables. Then, the PFHE design represented by each individual is evaluated and the best and worst designs selected, both of which are calculated using Eq. (23). Each design is checked to ensure that variables are not exceeding their boundary constraints. If the variable is out of bounds, it is randomly mapped back into the range of feasible values to keep it within bounds.

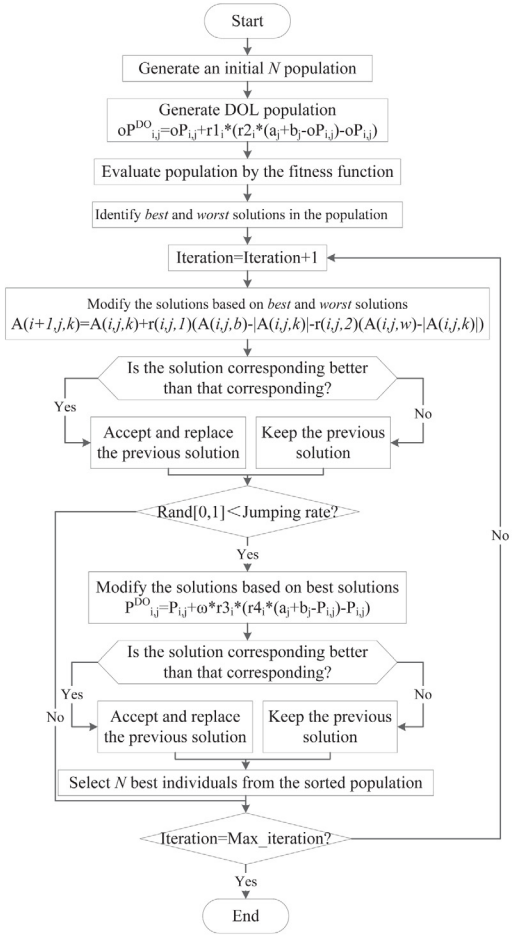
After evaluation, if the new design is better than the old one, the new design is retained. Otherwise, the old design is kept. If the jump condition is met, the best design is calculated using Eq. (29) and is retained for the next loop. Finally, when the whole evolutionary process has ended, the design with the best evaluation will be retained.

5. Experimental results and discussions

5.1. Benchmark test of CEC'14

In this section, 14 unimodal/multi-modal functions ($F1 - F14$) and 6 hybrid functions ($HF1 - HF6$) from CEC-2014 (Liang et al., 2013) are selected as benchmarks through which the proposed DOLJaya algorithm is compared with other competing algorithms. Unimodal functions are often used to test the search ability of algorithms, while multi-modal functions are used to test their exploration ability because the local optimal solutions can trap algorithms. The hybrid functions are a collection of unimodal and multi-modal functions which can be used to provide a more comprehensive evaluation of algorithm performance. The unimodal, multimodal and hybrid evaluation functions used are shown in Tables 2 and 3. In Table 3, N denotes the number of functions in the hybrid function, and P represents the mixing ratio of these functions. The composition of the mixing function used to generate the hybrid functions is given as follows:

$$F(x) = g_1\left(M_1 y_1\right) + \cdots + g_i\left(M_i y_i\right) + \cdots + g_N\left(M_N y_N\right) + F_x^* \tag{31}$$



(a)

Algorithm 1 The DOLJaya Algorithm

Randomly generate an initial population P;
for $i = 1; i \leq N; i++$ **do**
  $r1_i = rand(0,1), r2_i = rand(0,1)$;
  for $j = 1; j \leq D; j++$ **do**
    $P_{ij}^{DO} = P_{ij} + r1_i * (r2_i * (a_j + b_j - P_{ij}) - P_{ij})$;
    Check the boundaries;
  end for
end for
Select N number of the fittest individuals from $P \cup P^{DO}$;
Set G=0;
while G <Maximal evolutionary iteration **do**
  Evaluate all learners by the fitness function $f(.)$;
  if $G = 1$ **then**
    Sort individuals by the fitness value to get the best and worst population;
  else
    The *best* and *worst* solution is obtained by sorting the fitness values of the population
  end if
  for $i = 1; i \leq N; i++$ **do**
    Update the position of the individuals p according to the update mechanism Eq.23
    Check the boundaries;
    Evaluate the fitness values of the new individuals p' ;
    if $f(p)<f(p')$ **then**
      Replace p with p';
    end if
  end for
  if *rand* $>$ *Jr;* G++; back to 11
  if $rand \leq Jr$ **then**
    for $i = 1; i \leq N; i++$ **do**
      $r3_i = rand(0,1), r4_i = rand(0,1)$;
      for $j = 1; j \leq D; j++$ **do**
        $a_j = min(P_{ij}), b_j = max(P_{ij})$;
        $P_{ij}^{DO} = P_{ij} + w * r3_i * (r4_i * (a_j + b_j - P_{ij}) - P_{ij})$;
        Check the boundaries;
      end for
    end for
    Select N number of the fittest individuals from $P \cup P^{DO}$;
    $G++$;
  end if
end while

(b)

Fig. 2. The flowchart and pseudo code of the proposed DOLJaya.

$$y_1 = \left[Z_{s_1}, Z_{s_2}, \ldots, Z_{s_{n1}}\right]$$
$$y_2 = \left[Z_{s_{n1+1}}, Z_{s_{n1+2}}, \ldots, Z_{s_{n1+n2}}\right] \quad (32)$$
$$\ldots$$
$$y_N = \left[Z_{s_{\sum_{i=1}^{N-1} n_i+1}}, Z_{s_{\sum_{k=1}^{N-1} n_i+2}}, \ldots, Z_{s_D}\right]$$

$$Z = x - o_i \quad (33)$$

$$s = randperm(1 : D) \quad (34)$$

where $F(x)$ is the hybrid function, g_i is the ith basic construction function, and M is a rotation matrix, and

$$n_N = D - \sum_{i=1}^{N-1} n_i \quad (35)$$

where n_i is the dimension of the ith base function.

To validate the performance of DOLJaya, a number of alternative meta-heuristic algorithms are selected for comparison purposes, including several variants of Particle Swarm Optimization (PSO Kennedy and Eberhart, 1995, cfwPSO Clerc and Kennedy, 2002, cfPSO Clerc and Kennedy, 2002 and rlPSO Dong et al., 2018), several variants of Teacher-Learning Based Optimization (TLBO Rao et al., 2012, SLTLBO Zhile et al., 2014, and ETLBO Kadambur and Kotecha, 2015), Grey Wolf Optimization (GWO) (Mirjalili et al., 2014), and Jaya (Rao, 2016). The parameters of DOLJaya are set as follows: The jumping rate Jr is set as recommended in Rahnamayan et al. (2008). To determine a suitable choice for weight w, values in the range 1–15 were evaluated for four benchmark problems $F1$, $F3$, $F4$ and $F7$, with average performance computed over 10 independent runs of DOLJaya to account for the stochastic nature of the algorithm. The optimal value for each function was recorded and the average index (Ave index) of the results calculated. The average index is the average ranking of the algorithm among all algorithms. The results, which are presented in Table 5, reveal that the best performance is achieved when $w = 15$.

Table 2
Unimodal/multimodal functions.

Label	Formulas	Search Range	Dims
F1	$f1(z)=\sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}}z_i^2+f_{bias}, z=\mathrm{M}(x-\mathrm{o}),\ F1(z)=f1(z)+F_1^*$	$[-100,100]^D$	30
F2	$f2(z)=z_1^2+\sum_{i=2}^{D}(10^6)z_i^2+f_{bias}, z=\mathrm{M}(x-\mathrm{o}),\ F2(z)=f2(z)+F_2^*$	$[-100,100]^D$	30
F3	$f3(z)=(10^6)z_1^2+\sum_{i=2}^{D}z_i^2+f_{bias}, z=\mathrm{M}(x-\mathrm{o}),\ F3(z)=f3(z)+F_3^*$	$[-100,100]^D$	30
F4	$f4(z)=\sum_{i=1}^{D-1}\left(100\left(z_i^2-z_{i+1}\right)^2+\left(z_i-1\right)^2\right)+f_{bias}, z=\mathrm{M}(2.048(x-\mathrm{o})/100)+1,\ F4(z)=f4(z)+F_4^*$	$[-100,100]^D$	30
F5	$f5(z)=20-20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right)+e+f_{bias}, z=\mathrm{M}(x-\mathrm{o}),\ F5(z)=f5(z)+F_5^*$	$[-100,100]^D$	30
F6	$f6(z)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k}max\left[a^k cos\left(2\pi b^k(x_i+0.5)\right)\right]\right)-D\sum_{k=0}^{k}max\left[a^k cos\left(2\pi b^k\cdot 0.5\right)\right]+f_{bias}, a=0.5,\ b=3,\ kmax=20, z=\mathrm{M}(0.5(x-\mathrm{o})/100),\ F6(z)=f6(z)+F_6^*$	$[-100,100]^D$	30
F7	$f7(z)=\frac{1}{4000}\sum_{i=1}^{D}z_i^2-\prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right)+1+f_{bias}, z=\mathrm{M}(600(x-\mathrm{o})/100),\ F7(z)=f7(z)+F_7^*$	$[-100,100]^D$	30
F8	$f8(z)=\sum_{i=1}^{D}\left(z_i^2-10\cos(2\pi z_i)+10\right)+f_{bias}, z=\mathrm{M}(5.12(x-\mathrm{o})/100),\ F8(z)=f8(z)+F_8^*$	$[-100,100]^D$	30
F9	$f9(z)=\sum_{i=1}^{D}\left(z_i^2-10\cos(2\pi z_i)+10\right)+f_{bias}, z=\mathrm{M}(5.12(x-\mathrm{o})/100),\ F9(z)=f9(z)+F_9^*$	$[-100,100]^D$	30
F10	$f10(z)=418.9829\times D-\sum_{i=1}^{D}g(c_i)+f_{bias},\ c_i=z_i+4.2096874622755036e+002, z=\mathrm{M}(1000(x-\mathrm{o})/100),\ F10(z)=f10(z)+F_{10}^*$	$[-100,100]^D$	30
F11	$f11(z)=418.9829\times D-\sum_{i=1}^{D}g(c_i)+f_{bias},\ c_i=z_i+4.2096874622755036e+002, z=\mathrm{M}(1000(x-\mathrm{o})/100),\ F11(z)=f11(z)+F_{11}^*$	$[-100,100]^D$	30
F12	$f12(z)=\left\lvert\sum_{i=1}^{D}z_i^2-D\right\rvert^{0.25}+\left(0.5\sum_{i=1}^{D}z_i^2+\sum_{i=1}^{D}z_i\right)/D+0.5, z=\mathrm{M}(5(x-\mathrm{o})/100),\ F12(z)=f12(z)+F_{12}^*$	$[-100,100]^D$	30
F13	$f13(z)=\left\lvert\left(\sum_{i=1}^{D}z_i^2\right)^2-\left(\sum_{i=1}^{D}z_i\right)^2\right\rvert^{0.5}+\left(0.5\sum_{i=1}^{D}z_i^2+\sum_{i=1}^{D}z_i\right)/D+0.5, z=\mathrm{M}(5(x-\mathrm{o})/100),\ F13(z)=f13(z)+F_{13}^*$	$[-100,100]^D$	30
F14	$f14(z)=f_7(f_1(x_1,x_2))+f_7(f_1(x_2,x_3))+\cdots+f_7(f_1(x_{D-1},x_D))+f_7(f_1(x_D,x_1)),\ z=\mathrm{M}(5(x-\mathrm{o})/100)+1,\ F14(z)=f14(z)+F_{14}^*$	$[-100,100]^D$	30

Table 3
Hybrid functions.

Label	Formulas	Search Range	Dims
HF1	g1 : Modified Schwefel's Function $f9$ g2 : Rastrigin's Function $f8$ g3: High Conditioned Elliptic Function $f1$ $N=3,\ p=[0.3,0.3,0.4]$	$[-100,100]^D$	30
HF2	g1 : Bent Cigar Function $f2$ g2 : HGBat Function $f12$ g3: Rastrigin's Function $f8$ $N=3,\ p=[0.3,0.3,0.4]$	$[-100,100]^D$	30
HF3	g1 : Griewank's Function $f7$ g2 : Weierstrass Function $f6$ g3: Rosenbrock's Function $f4$ g4: Scaffer's F6 Function $f14$ $N=4,\ p=[0.2,0.2,0.3,0.3]$	$[-100,100]^D$	30
HF4	g1 : HGBat Function $f12$ g2 : Discus Function $f3$ g3: Expanded Griewank's plus Rosenbrock's Function $f13$ g4: Rastrigin's Function $f8$ $N=4,\ p=[0.2,0.2,0.3,0.3]$	$[-100,100]^D$	30
HF5	g1 : Scaffer's F6 Function $f14$ g2 : HGBat Function $f12$ g3: Rosenbrock's Function $f4$ g4: Modified Schwefel's Function $f9$ g5: High Conditioned Elliptic Function $f1$ $N=5,\ p=[0.1,0.2,0.2,0.2,0.3]$	$[-100,100]^D$	30
HF6	g1 : Katsuura Function $f10$ g2 : HappyCat Function $f11$ g3: Expanded Griewank's plus Rosenbrock's Function $f13$ g4: Modified Schwefel's Function $f9$ g5: Ackley's Function $f5$ $N=5,\ p=[0.1,0.2,0.2,0.2,0.3]$	$[-100,100]^D$	30

The settings of all algorithm control parameters are shown in Table 4. The software and hardware facilities for this experiment are: Matlab R2019a, 8-core 4.8 GHz processor, 32 GB RAM. The population size is 50 in all cases. For the unimodal and multi-modal benchmark functions, function evaluations (FES) is 300,000, for the hybrid benchmark functions, FES is 50,000, and for the actual engineering problem it is 50,000. The mean and standard deviation (Std) of the objective function $f(X)$ averaged over 10 repetitions of the algorithms are recorded for each benchmark function, while the best, worst, mean and stand deviation of the selected objective function are recorded for the PFHE design problems.

The results for the unimodal and multi-modal benchmark functions are shown in Table 6, and the average convergence results are shown in Fig. 3. It can be seen from these results that the DOLJaya algorithm performs the best in half of the unimodal benchmark functions and in the majority of the multi-modal benchmark functions. It is worth noting that DOLJaya is better than Jaya in all benchmark tests. It is most frequently the best performing algorithm (Best num) and has the highest

Table 4
Control parameters of all algorithms.

Algorithms	Parameters	Values
DOLJaya	Jr	0.3
PSO	C_1 and C_2	2
rlPSO		2
cfwPSO		2.05
cfPSO		2.05
cfwPSO	K	0.729
cfPSO		0.729
IPO	c_1	0.685
	c_2	0.653
	$shift_1$	527.392
	$shift_2$	380.811
	$scale_1$	0.423
	$scale_2$	0.571
GSA	G_0	100
	α	20

average ranking (Ave index) among all the algorithms considered in the table. Hence, DOLJaya has the best performance among all compared algorithms, which demonstrates that the DOL strategy greatly improves the performance of the original Jaya algorithm.

Table 7 and Fig. 4 show the results and convergence trend of DOLJaya for the hybrid benchmark functions. Among the six functions tested, DOLJaya yields the best performance for 5 of the 6 functions. It is only outperformed by GWO and rlPSO for HF2. This shows that DOLJaya has strong robustness and adaptability. In most hybrid functions, DOLJaya achieves faster convergence early in computation and is faster than the Jaya algorithm. DOLJaya continues to get better results when other algorithms do not change much. In can be seen that for real world applications, which mostly involve hybrid problems, DOLJaya has greater potential for consistent performance than the other algorithms.

Table 8 and Fig. 5 show the comparison results and convergence trend of DOLJaya, OLJaya and Jaya for four unimodal/multi-modal test functions $F1-F4$, $F10$ and $F15$. From these results, it is obvious that in most benchmark functions, the performance of DOLJaya algorithm is better than OLJaya and Jaya algorithm. Compared with other algorithms, DOLJaya can be improved by up to 38.67%. This demonstrates that the proposed DOL strategy with expanding asymmetric search spaces can effectively improve the possibility of converging to the best optimum, which can significantly achieve an excellent exploitation capability.

In order to compare the performance of OLJaya and DOLJAYA algorithms more comprehensively, we choose the composition benchmark functions, which use the hybrid functions as the basic functions. And the composition function is defined as follow:

$$CF(x)=\sum_{i=1}^{N}\left\{\omega_i^*\left[\lambda_i g_i(x)+bias_i\right]\right\}+F^* \tag{36}$$

where $CF(x)$ is the composition function, $bias_i$ defines which optimum is global optimum, λ_i is used to control each $g_i(x)$'s height. The weight value ω_i for each $g_i(x)$ is calculated as follow:

$$\omega_i=\frac{1}{\sqrt{\sum_{j=1}^{D}\left(x_j-o_{ij}\right)^2}}exp\left(-\frac{\sum_{j=1}^{D}\left(x_j-o_{ij}\right)^2}{2D\sigma_i^2}\right) \tag{37}$$

where σ_i is used to control each $g_i(x)$'s coverage range.

For hybrid and composition benchmark functions, DOLJaya algorithm also shows good performance, as show in Fig. 6 and Table 9. Among the hybrid test functions, DOLJaya performs better than OLJaya and Jaya for $HF1$, $HF2$ and $HF6$. It is only better than OLJaya for $HF4$, and better than Jaya algorithm for $HF5$. However, for the composition functions$CF6$, $CF7$ and $CF8$, the DOLJaya algorithm is

Table 5
The sensitivity analysis of w was performed using the benchmark function in the range of weights 0–15.

Weight	Mean				Ave index
	F1	F3	F4	F7	
$w = 1$	1.90E+08	8.29E+04	1.33E+08	4.17E+02	13
$w = 2$	**1.29E+08**	7.95E+04	1.26E+08	3.75E+07	6
$w = 3$	1.58E+08	7.57E+04	1.21E+08	3.59E+07	6
$w = 4$	1.61E+08	8.02E+04	1.35E+08	4.27E+07	11
$w = 5$	1.76E+08	8.32E+04	1.12E+08	3.76E+07	10
$w = 6$	1.36E+08	7.37E+04	1.65E+08	3.32E+07	6
$w = 7$	1.44E+08	8.68E+04	1.41E+08	3.67E+07	10
$w = 8$	1.86E+08	8.00E+04	1.22E+08	4.04E+07	10
$w = 9$	1.45E+08	8.05E+04	1.49E+08	4.07E+07	11
$w = 10$	1.84E+08	8.80E+04	1.45E+08	3.59E+07	12
$w = 11$	1.62E+08	7.80E+04	1.05E+08	**3.28E+07**	5
$w = 12$	1.44E+08	8.10E+04	**9.15E+07**	4.04E+07	7
$w = 13$	1.64E+08	8.23E+04	9.37E+07	3.76E+07	8
$w = 14$	1.38E+08	8.55E+04	9.61E+07	3.29E+07	6
$w = \mathbf{15}$	1.29E+08	**7.14E+04**	1.03E+08	3.29E+07	**2**

Table 6
The mean and standard value of unimodal/multi-modal test functions.

Algorithms	F1		F2		F3		F4	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
PSO	2.51E+08	1.14E+08	1.49E+10	2.77E+09	2.42E+04	8.91E+03	1.87E+09	4.26E+02
rlPSO	5.30E+07	2.69E+07	**1.15E+07**	**5.37E+06**	**8.14E+02**	**1.61E+03**	8.89E+02	1.39E+02
cfwPSO	2.99E+08	1.13E+08	1.81E+10	6.85E+09	3.65E+04	1.12E+04	3.19E+03	1.04E+03
cfPSO	2.30E+08	1.17E+08	1.82E+10	7.36E+09	2.95E+04	9.45E+03	1.86E+03	6.09E+02
GWO	5.37E+07	3.69E+07	2.72E+09	3.01E+09	2.85E+04	7.55E+03	6.89E+02	1.11E+02
TLBO	3.13E+08	1.63E+08	4.19E+10	1.08E+10	5.42E+04	1.25E+04	5.85E+03	2.23E+03
SLTLBO	3.68E+08	1.29E+08	3.27E+10	8.19E+09	5.42E+04	8.73E+03	4.61E+03	1.87E+03
ETLBO	4.67E+08	2.19E+08	3.46E+10	8.40E+09	6.58E+04	1.68E+04	4.34E+03	1.65E+03
Jaya	5.48E+06	4.50E+06	1.61E+09	9.01E+08	1.34E+04	9.07E+03	6.84E+02	7.93E+01
DOLJaya	**4.06E+06**	**2.57E+06**	3.79E+07	4.95E+07	2.45E+03	1.62E+03	**5.04E+02**	**2.43E+ 01**
Algorithms	F5		F6		F7		F8	
PSO	**2.03E+01**	1.15E−01	2.77E+01	3.41E+00	1.44E+02	2.82E+01	1.70E+02	3.35E+01
rlPSO	2.06E+01	1.97E−01	2.24E+01	2.59E+00	3.38E+00	1.15E+00	**5.95E+01**	1.74E+01
cfwPSO	2.09E+01	7.18E−02	2.91E+01	4.02E+00	1.73E+02	6.63E+01	1.60E+02	3.49E+01
cfPSO	2.04E+01	1.26E−01	2.81E+01	2.83E+00	1.77E+01	5.26E+01	1.44E+02	1.94E+01
GWO	2.10E+01	**4.37E−02**	1.40E+01	2.43E+00	4.20E+01	2.41E+01	9.03E+01	2.44E+01
TLBO	2.06E+01	8.96E−02	3.18E+01	2.22E+00	3.78E+02	9.63E+01	2.46E+02	1.44E+01
SLTLBO	2.06E+01	1.10E−01	3.03E+01	**1.94E+00**	3.43E+02	1.14E+02	2.46E+02	**1.33E+01**
ETLBO	2.11E+01	1.53E−01	3.59E+01	3.68E+00	3.67E+02	1.04E+02	2.60E+02	2.84E+01
Jaya	2.09E+01	5.04E−02	2.10E+01	3.30E+00	2.39E+01	8.84E+00	7.50E+02	3.49E+01
DOLJaya	2.09E+01	6.51E−02	**8.51E+00**	2.46E+00	**1.13E+00**	**8.24E−01**	7.45E+01	2.17E+01
Algorithms	F9		F10		F11		F12	
PSO	2.05E+02	1.54E+01	2.98E+03	8.20E+02	6.65E+03	**1.33E+03**	3.46E+00	3.78E−01
rlPSO	8.70E+01	2.49E+01	**1.18E+03**	4.58E+02	3.50E+03	5.09E+02	4.43E−01	1.19E−01
cfwPSO	2.03E+02	3.47E+01	2.46E+03	7.53E+02	5.18E+03	8.36E+02	3.81E+00	5.83E−01
cfPSO	1.60E+02	3.86E+01	3.79E+03	5.85E+02	4.20E+03	7.42E+02	3.34E+00	3.58E−01
GWO	1.00E+02	2.34E+01	2.04E+03	4.53E+02	**2.83E+03**	4.61E+02	6.62E−01	7.02E−01
TLBO	2.50E+02	2.38E+01	5.83E+03	**3.77E+02**	5.79E+03	3.80E+02	4.98E+00	4.25E−01
SLTLBO	2.77E+02	2.22E+01	5.76E+03	6.63E+02	5.53E+03	3.63E+02	4.40E+00	5.62E−01
ETLBO	2.80E+02	3.00E+01	6.52E+03	6.67E+02	6.79E+03	6.11E+02	4.85E+00	8.04E−01
Jaya	1.83E+02	1.45E+01	3.88E+03	8.78E+02	6.17E+03	4.68E+02	5.30E−01	**6.45E−02**
DOLJaya	**5.46E+01**	**1.28E+01**	2.04E+03	6.16E+02	5.03E+03	1.74E+03	**3.56E−01**	6.11E−02
Algorithms	F13		F14		Best num		Ave index	
PSO	6.42E+01	1.59E+01	4.01E+02	3.65E+02	1		5.21	
rlPSO	**2.70E−01**	2.51E+00	1.99E+01	6.26E+00	5		2.29	
cfwPSO	7.13E+01	1.85E+01	9.92E+02	8.23E+02	0		6.57	
cfPSO	6.40E+01	1.63E+01	4.61E+03	4.23E+03	0		5.86	
GWO	7.50E+00	6.92E+00	4.01E+01	4.09E+01	1		3.64	
TLBO	1.40E+02	3.38E+01	4.60E+04	3.76E+04	0		8.36	
SLTLBO	1.01E+02	4.10E+01	3.40E+04	2.17E+04	0		7.57	
ETLBO	1.29E+02	3.54E+01	3.97E+04	1.77E+04	0		9.21	
Jaya	1.55E+00	8.13E+00	4.30E+01	3.25E+01	0		4.29	
DOLJaya	2.89E−01	**1.76E−02**	**1.49E+01**	**1.62E+00**	**7**		**2.00**	

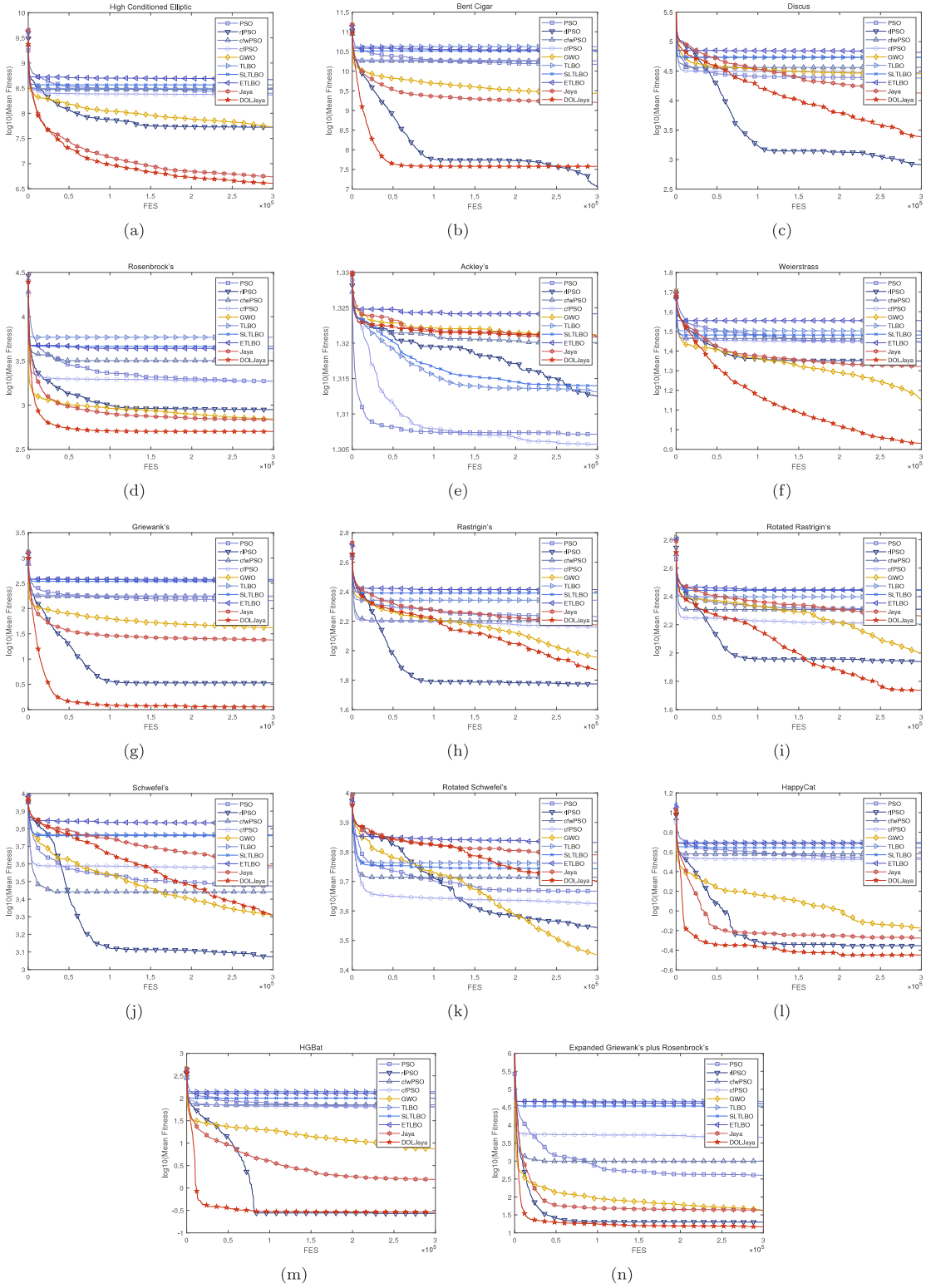


Fig. 3. The convergence trends of all algorithms on all unimodal and simple multimodal functions.

superior to the other two algorithms. It can be seen that DOLJaya has the potential to deal with comprehensive problems in real world.

To assess the statistical significance of the results obtained, the Wilcoxon rank-sum test (Wilcoxon, 1992) is used to compare the performance for the different algorithms. The results for a significance level of 0.05 and a 2-tailed hypothesis are presented in Table 7. In the table, '↑' means that DOLJaya is better, '←' means that the algorithm indicated by the arrow is better, and '=' means that there is no significant difference between the two algorithms. It can be seen from the results that DOLJaya is significantly outperforming Jaya and the other algorithms in most benchmarks. Hence, we can conclude that incorporating the DOL strategy improves the performance of Jaya

Table 7
The mean and standard value of hybrid test functions.

Algorithms	HF1		HF2		HF3		HF4	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
PSO	5.00E+06	3.54E+06	1.32E+06	3.06E+06	5.53E+01	2.83E+01	6.61E+03	4.20E+03
rlPSO	6.34E+05	8.02E+05	2.37E+03	2.11E+03	2.20E+01	2.02E+01	3.51E+02	2.55E+02
cfwPSO	4.46E+06	3.03E+06	2.93E+07	2.68E+07	9.73E+01	2.16E+01	1.72E+04	8.32E+03
cfPSO	4.17E+06	3.00E+06	1.71E+07	2.89E+07	1.04E+02	4.88E+01	1.15E+04	8.42E+03
GWO	1.58E+06	2.16E+06	**1.89E+03**	**2.08E+03**	6.89E+01	2.93E+01	4.14E+04	8.25E+03
TLBO	4.13E+06	5.63E+06	8.18E+07	7.99E+07	1.68E+02	7.41E+01	1.76E+04	1.21E+04
SLTLBO	3.97E+06	2.71E+06	3.55E+07	6.42E+07	1.82E+02	8.38E+01	1.56E+04	1.05E+04
ETLBO	1.29E+07	9.85E+06	3.93E+08	6.76E+08	2.13E+02	9.50E+01	2.64E+04	2.37E+04
Jaya	4.86E+04	4.00E+04	5.97E+04	2.57E+04	1.65E+01	2.02E+01	2.11E+02	6.15E+01
DOLJaya	**3.09E+04**	**1.79E+04**	1.48E+04	8.96E+03	**6.12E+00**	**8.03E−01**	**1.66E+02**	**1.85E+01**
Algorithms	HF5		HF6		Best num		Ave index	
PSO	3.54E+05	3.09E+05	6.45E+02	1.53E+02	0		5.67	
rlPSO	3.83E+04	2.76E+04	3.57E+02	1.50E+02	0		2.83	
cfwPSO	5.81E+05	4.89E+05	8.08E+02	1.66E+02	0		7.33	
cfPSO	9.03E+05	1.18E+06	5.95E+02	2.09E+02	0		6.67	
GWO	8.04E+05	2.15E+06	3.67E+02	1.47E+02	1		4.50	
TLBO	4.09E+05	5.74E+05	5.32E+02	2.01E+02	0		7.00	
SLTLBO	8.50E+05	8.59E+05	6.10E+02	1.52E+02	0		7.33	
ETLBO	2.27E+06	2.85E+06	8.42E+02	2.61E+02	0		10.00	
Jaya	1.03E+04	**3.86E+03**	3.20E+02	1.08E+02	0		2.33	
DOLJaya	**3.10E+03**	4.81E+03	**1.90E+02**	**7.36E+01**	**5**		**1.33**	

Table 8
The mean and standard value of unimodal/multi-modal test functions compared with OLJaya.

Algorithms	F1		F2		F3	
	Mean	Std	Mean	Std	Mean	Std
Jaya	2.32E+08	5.47E+07	2.18E+10	4.50E+09	**1.26E+05**	2.83E+04
OLJaya	2.58E+08	7.73E+07	**2.15E+10**	4.92E+09	1.39E+05	**1.31E+04**
DOLJaya	**1.85E+08**	**4.94E+07**	8.65E+09	**2.76E+09**	9.70E+04	1.58E+04
Algorithms	F4		F10		F15	
Jaya	1.74E+03	1.29E+03	**3.04E+03**	2.97E+03	2.92E+03	**2.40E+03**
OLJaya	2.09E+03	1.10E+03	3.22E+03	3.26E+03	1.20E+04	1.77E+04
DOLJaya	**4.04E+02**	**7.67E+02**	3.19E+03	**2.95E+03**	**9.73E+02**	2.94E+03

Table 9
The mean and standard value of hybrid test functions compared with OLJaya.

Algorithms	HF1		HF2		HF4		HF5	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Jaya	1.25E+07	7.01E+06	2.49E+08	9.65E+07	**3.38E+04**	**2.47E+04**	7.21E+06	5.66E+06
OLJaya	1.63E+07	8.30E+06	**2.42E+08**	**1.01E+08**	8.03E+04	6.45E+04	**2.96E+06**	**1.49E+06**
DOLJaya	**1.06E+07**	**4.06E+06**	7.48E+07	3.87E+07	5.38E+04	5.13E+04	4.42E+06	2.15E+06
Algorithms	HF6		CH6		CF7		CF8	
Jaya	3.43E+02	1.51E+03	2.88E+03	2.73E+03	7.76E+07	6.38E+07	4.57E+05	1.58E+05
OLJaya	2.64E+02	1.53E+03	3.07E+03	2.53E+03	1.13E+08	6.34E+07	4.14E+05	7.93E+04
DOLJaya	7.84E+01	**1.44E+03**	**1.77E+03**	**2.11E+03**	**6.27E+07**	**2.40E+07**	**2.11E+05**	**7.74E+04**

significantly, especially when applied to the hybrid benchmarks. This is because DOLJaya has a dynamic and asymmetric search space which increases search exploitation and yields excellent convergence performance. The consistent rapid convergence characteristic of DOLJaya is illustrated in Figs. 3 and 4 (see Table 10).

5.2. Benchmark test of CEC'17

In this section, in order to more comprehensively evaluate the performance of DOLJAYA algorithm, we have added three comparison algorithms IPO (Mozaffari et al., 2016), GSA (Rashedi et al., 2009) and SSA (Çelik et al., 2021), and selected unimodal/multi-modal functions $F1$, $F3$, $F4$ and $F9$, and Hybrid functions $HF3$, $HF5$, $HF9$ and $HF10$ in CEC'17 as benchmark functions, the benchmark functions for CEC'17 is similar to CEC'14, that each function has a shift data, but there are more hybrid functions in CEC'17 than CEC'14. Moreover, the mean and Std of each algorithm recorded have been run more than 10 times on average.

The results and the convergence trends are given in Table 11 and Fig. 7. For unimodal function $F1$ and multi-modal functions $F4$ and $F9$, DOLJaya is superior to most other algorithms, while for $F3$, the performance difference of all algorithms is small. Therefore, DOLJAYA has good stability in solving different problems, but its competitiveness is not as good as rlPSO for $F1$ and $F9$ in terms of asymmetric problems. In addition, the SSA algorithm shows better performance, which the speed of convergence is faster than DOLJaya.

Table 12 and Fig. 8 show the results of algorithms on hybrid functions. With the continuous increase of FES, the performance of DOLJaya is getting better and better, while most other algorithms have not changed at this time. Compared with the three new algorithms IPO, GSA and SSA, DOLJaya has better performance in most of the benchmark functions, which proves DOLJaya has a competitive exploration capability when compared with other algorithms. However, for $HF1$, SSA has a fast convergence speed, which means it has better exploration ability than DOLJaya.

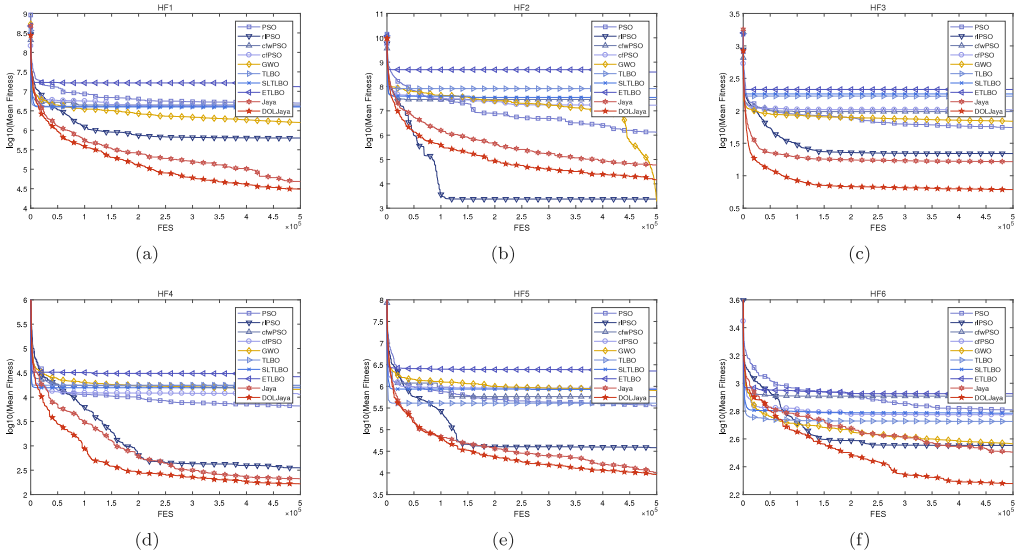


Fig. 4. The convergence trends of all algorithms on all hybrid functions.

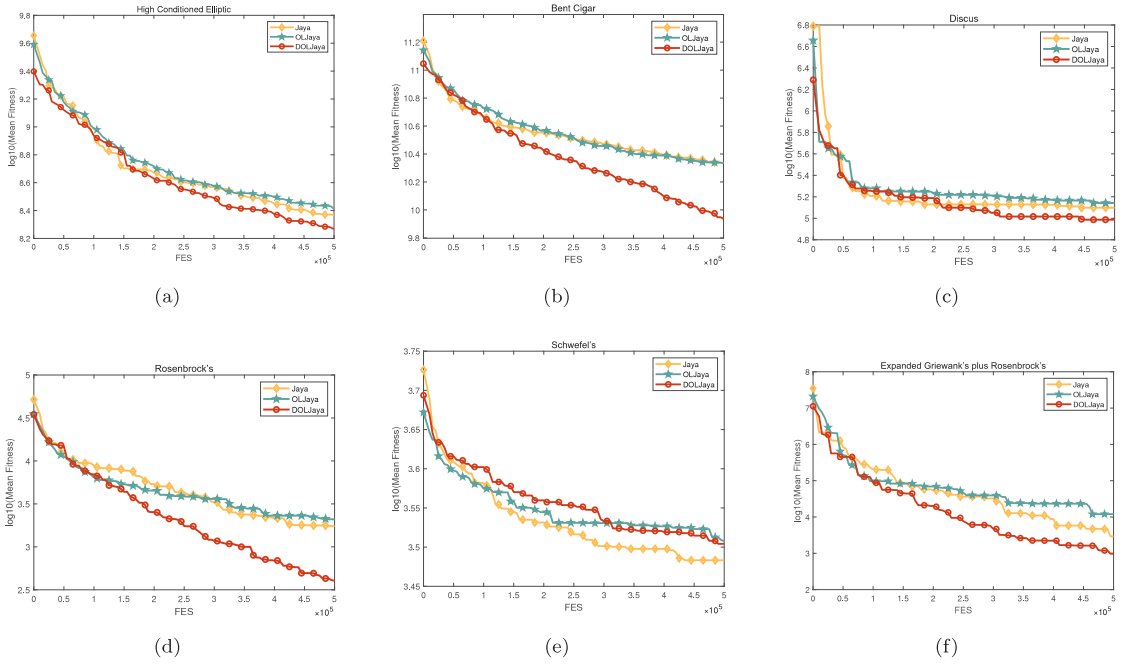


Fig. 5. The convergence trends of algorithms on unimodal/multi-modal test functions compared with OLJaya.

5.3. Application to PFHE design problems

In this section, DOLJaya is evaluated for plate–fin heat exchanger design optimization. The PFHE design problem and selected case studies have been described in detail in Section 2. To ensure independence, the optimization algorithm is repeated 10 times and the best, worst, average and standard deviation are recorded for each case study/objective function combination considered. The average of at each iteration over the 10 repetitions is plotted to illustrate the convergence trend. With two design case studies and two possible evaluation functions, (i.e. Eq. (20), objective function 1) and Eq. (21) (objective function 2), there are four sets of results. For comparisons purposes the performance of nine alternative optimization algorithms is also reported. The results are shown in Table 13. The convergence trend of objective function 1 is shown in Fig. 9, while the convergence trend of objective function 2 is shown in Fig. 10.

It can be concluded from Table 13 that DOLJaya yields the best overall performance with the best results in three of the four combinations considered, and only marginally outperformed by Jaya and SLTLBO in the fourth combination. Figs. 9 and 10 also illustrated that although the results are similar even in logarithmic coordinates, DOLJaya converges more quickly than other algorithms in most cases. This shows that DOLJaya has an advantage over other optimization algorithms for PFHE design, and suggests that it has great potential

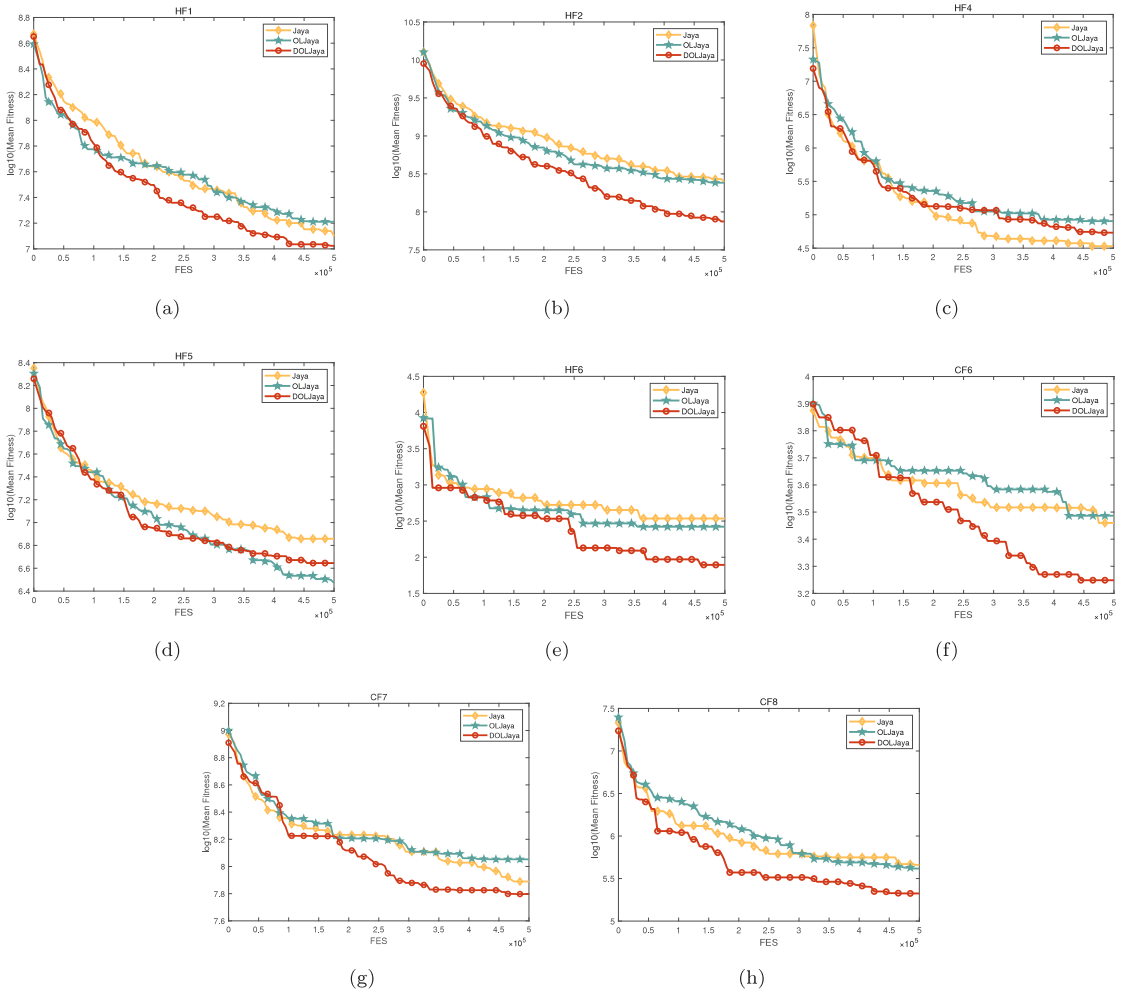


Fig. 6. The convergence trends of algorithms on hybrid functions and composition functions compared with OLJaya.

Table 10
Wilcoxon rank-sum test on unimodal/multi-modal/hybrid functions.

	DOLJaya											
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
PSO	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	=	↑
rlPSO	↑	=	←	↑	←	↑	↑	=	↑	←	↑	↑
cfwPSO	↑	↑	↑	↑	=	↑	↑	↑	↑	=	=	↑
cfPSO	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	=	↑
GWO	↑	↑	↑	↑	=	↑	↑	=	↑	=	←	↑
TLBO	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	=	↑
SLTLBO	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	=	↑
ETLBO	↑	↑	↑	↑	=	↑	↑	↑	↑	↑	↑	↑
Jaya	=	↑	↑	↑	=	↑	↑	↑	↑	↑	↑	↑
PSO	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	=	↑

	DOLJaya								Same	Better
	F13	F14	HF1	HF2	HF3	HF4	HF5	HF6		
PSO	↑	↑	↑	↑	↑	↑	↑	↑	1	18
rlPSO	=	↑	↑	↑	↑	↑	↑	↑	3	14
cfwPSO	↑	↑	↑	↑	↑	↑	↑	↑	3	17
cfPSO	↑	↑	↑	↑	↑	↑	↑	↑	1	18
GWO	=	↑	↑	↑	↑	↑	↑	↑	4	15
TLBO	↑	↑	↑	↑	↑	↑	↑	↑	1	18
SLTLBO	↑	↑	↑	↑	↑	↑	↑	↑	1	18
ETLBO	↑	↑	↑	↑	↑	↑	↑	↑	1	19
Jaya	=	↑	=	↑	↑	↑	=	↑	5	15

Table 11
The mean and standard value of unimodal/multi-modal test functions of CEC'17.

Algorithms	F1		F3		F4		F9	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
PSO	2.99E+10	3.49E+09	7.65E+04	**5.70E+03**	8.45E+03	2.26E+03	7.42E+03	2.22E+02
rlPSO	**4.10E+09**	**8.16E+08**	7.58E+04	1.78E+04	2.35E+02	6.83E+02	**2.65E+03**	4.96E+02
cfwPSO	3.80E+10	9.03E+09	1.64E+05	1.71E+04	8.77E+03	3.68E+03	1.10E+04	1.29E+03
cfPSO	1.73E+10	7.46E+09	7.26E+04	9.03E+03	3.15E+03	1.47E+03	4.29E+03	1.65E+03
GWO	6.96E+09	6.14E+09	1.03E+05	3.06E+04	1.99E+02	1.08E+03	4.27E+03	1.06E+03
TLBO	3.38E+10	6.54E+09	6.49E+04	7.27E+03	8.88E+03	5.08E+03	4.41E+03	1.02E+03
SLTLBO	3.36E+10	6.97E+09	**5.80E+04**	7.48E+03	7.33E+03	2.21E+03	8.68E+03	**1.22E+02**
ETLBO	3.61E+10	6.22E+09	9.86E+04	2.08E+04	9.53E+03	4.27E+03	8.78E+03	5.15E+02
Jaya	1.95E+10	3.06E+09	1.41E+05	2.55E+04	2.20E+03	8.44E+02	7.14E+03	5.48E+03
IPO	6.66E+10	4.28E+09	8.81E+04	1.14E+04	1.62E+04	2.02E+03	1.30E+04	1.07E+03
GSA	6.87E+10	4.35E+09	1.53E+05	8.79E+04	1.83E+04	4.37E+03	1.45E+04	1.08E+03
SSA	8.50E+09	4.84E+09	2.14E+05	1.07E+05	**4.44E+01**	1.12E+03	6.84E+03	7.56E+02
DOLJaya	5.19E+09	1.62E+09	9.83E+04	1.87E+04	8.56E+01	**4.77E+02**	2.84E+03	1.19E+03

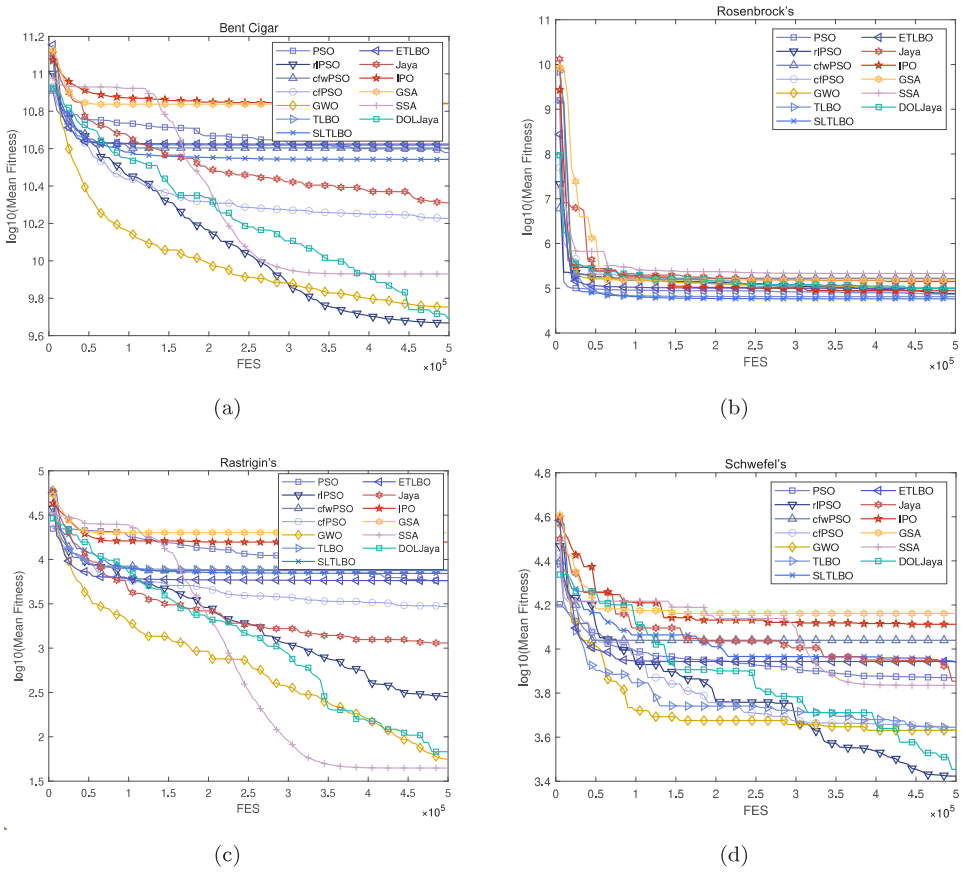


Fig. 7. The convergence trends of algorithms on unimodal/multi-modal test functions of CEC'17.

Table 12
The mean and standard value of hybrid functions of CEC'17.

Algorithms	HF3		HF5		HF9		HF10	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
PSO	6.03E+09	2.05E+09	1.42E+06	7.23E+05	3.81E+07	2.81E+07	1.32E+07	1.21E+07
rlPSO	4.34E+08	3.94E+08	**2.07E+04**	3.62E+04	**3.29E+05**	**5.88E+05**	**9.39E+03**	**1.08E+04**
cfwPSO	5.42E+09	1.55E+09	1.17E+06	7.52E+05	2.40E+07	7.66E+06	8.06E+07	5.02E+07
cfPSO	2.41E+09	6.87E+08	1.06E+06	1.40E+06	4.33E+06	7.06E+06	6.99E+07	1.44E+08
GWO	**2.25E+08**	2.94E+08	1.23E+06	9.49E+05	2.57E+06	2.42E+06	2.03E+06	2.96E+06
TLBO	4.80E+09	6.32E+08	2.78E+04	**2.84E+04**	6.69E+05	6.37E+05	2.78E+07	2.15E+07
SLTLBO	2.76E+09	9.67E+08	2.35E+04	3.62E+04	8.57E+05	9.05E+05	1.76E+06	1.17E+06
ETLBO	9.66E+09	1.90E+09	1.11E+05	1.22E+05	6.40E+06	1.14E+07	7.27E+07	1.17E+08
Jaya	1.56E+09	1.09E+09	1.62E+06	9.20E+05	7.84E+06	4.98E+06	2.59E+07	7.52E+06
IPO	1.81E+10	3.15E+09	1.44E+07	9.17E+06	4.41E+08	2.75E+08	2.22E+09	2.20E+08
GSA	1.53E+10	4.57E+09	1.75E+07	7.28E+06	1.68E+08	1.53E+08	1.23E+09	9.20E+08
SSA	3.85E+08	2.18E+08	2.38E+06	3.83E+06	2.88E+07	3.30E+07	2.34E+07	1.36E+07
DOLJaya	4.99E+08	**1.31E+08**	5.63E+05	3.87E+05	1.16E+07	4.91E+06	7.18E+06	1.98E+06

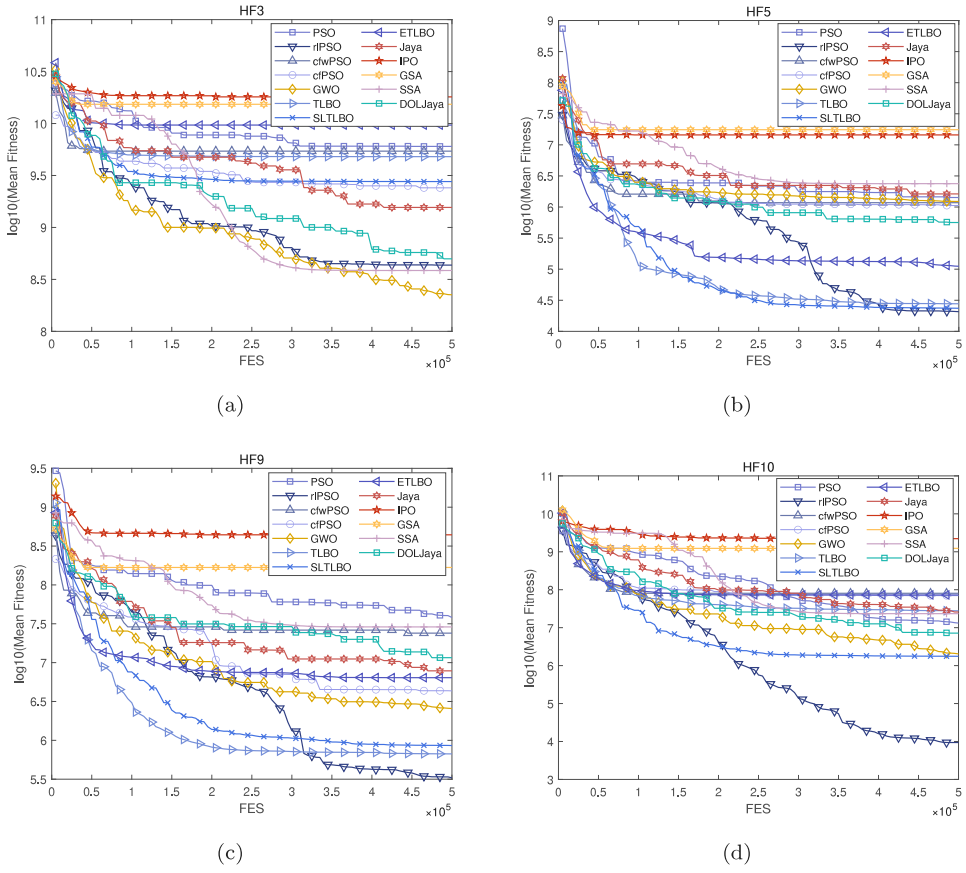


Fig. 8. The convergence trends of algorithms on hybrid functions of CEC'17.

Table 13
Results of the PFHE design problem.

Algorithms	Case study 1 and Objective function 1				Case study 1 and Objective function 2			
	Mean	Std	Best	Worst	Mean	Std	Best	Worst
PSO	8.564E−02	2.050E−03	8.371E−02	8.894E−02	1.865E+00	1.125E−01	1.721E+00	2.004E+00
rlPSO	8.450E−02	2.649E−03	8.110E−02	8.795E−02	2.004E+00	1.830E−01	1.743E+00	2.222E+00
cfwPSO	8.415E−02	5.539E−04	8.358E−02	8.486E−02	1.703E+00	1.965E−01	1.397E+00	1.869E+00
cfPSO	8.479E−02	2.670E−03	8.160E−02	8.743E−02	7.935E+00	6.899E−02	1.857E+00	2.033E+00
GWO	7.932E−02	4.134E−04	7.878E−02	7.972E−02	1.054E+00	5.864E−04	1.053E+00	1.054E+00
TLBO	8.287E−02	7.946E−03	8.014E−02	8.968E−02	1.196E+00	1.206E−01	1.057E+00	1.361E+00
SLTLBO	8.177E−02	1.946E−03	8.014E−02	8.362E−02	1.171E+00	1.010E−01	1.070E+00	1.285E+00
ETLBO	8.638E−02	4.653E−03	8.232E−02	9.259E−02	1.278E+00	1.950E−01	1.090E+00	1.562E+00
Jaya	8.049E−02	1.410E−03	7.924E−02	8.283E−02	1.103E+00	7.138E−02	1.051E+00	1.185E+00
DOLJaya	**7.863E−02**	**3.248E−04**	**7.830E−02**	**7.908E−02**	**1.051E+00**	**5.088E−16**	**1.051E+00**	**1.051E+00**
Algorithms	Case study 2 and Objective function 1				Case study 2 and Objective function 2			
	Mean	Std	Best	Worst	Mean	Std	Best	Worst
PSO	1.626E−01	2.013E−03	1.596E−01	1.649E−01	4.597E+00	5.937E-.1	4.002E+00	5.408E+00
rlPSO	1.626E−01	2.853E−03	1.597E−01	1.670E−01	4.176E+00	6.280E−01	3.297E+00	4.915E+00
cfwPSO	1.623E−01	2.067E−03	1.597E−01	1.649E−01	3.758E+00	3.885E−01	3.176E+00	4.210E+00
cfPSO	1.609E−01	1.184E−03	1.599E−01	1.622E−01	2.938E+00	3.871E−01	2.279E+00	3.201E+00
GWO	**1.591E−01**	3.840E−04	1.589E−01	1.589E−01	1.171E+00	1.790E−01	1.083E+00	1.491E+00
TLBO	1.604E−01	1.005E−03	1.591E−01	1.618E−01	1.313E+00	2.107E−01	1.075E+00	1.598E+00
SLTLBO	1.600E−01	1.343E−03	1.586E−01	1.622E−01	1.737E+00	2.680E−01	1.311E+00	2.038E+00
ETLBO	1.612E−01	2.667E−03	1.591E−01	1.656E−01	2.066E+00	6.067E−01	1.610E+00	3.111E+00
Jaya	1.598E−01	1.773E−03	**1.585E−01**	1.928E−01	1.310E+00	1.940E−01	1.073E+00	1.594E+00
DOLJaya	**1.591E−01**	**2.864E−04**	1.588E−01	**1.594E−01**	**1.068E+00**	**8.672E−13**	**1.068E+00**	**1.068E+00**

for other practical engineering optimization problems. Compared with the other algorithms, for Eqs. (20) and (21) problems solved in Section 5.3 of this paper, DOLJaya obtained a maximum average increase of 108.29%. Compared with the original Jaya, an increase of 7.60% is achieved. For the minimum value of the solution, the average increase is 87.01%, which means an 0.37% increase compared to the original algorithm.

As a conclusion, DOLJaya performs excellently both for the benchmark tests and the practical engineering problem, a consequence of its dynamic search ability, good balance between exploration and

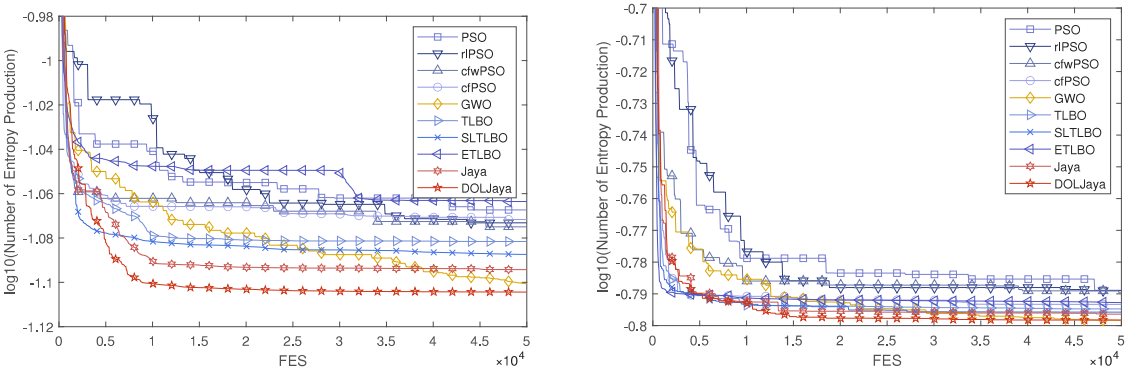


Fig. 9. Convergence results of PFHE design problems with objective function 1 for all the compared algorithms.

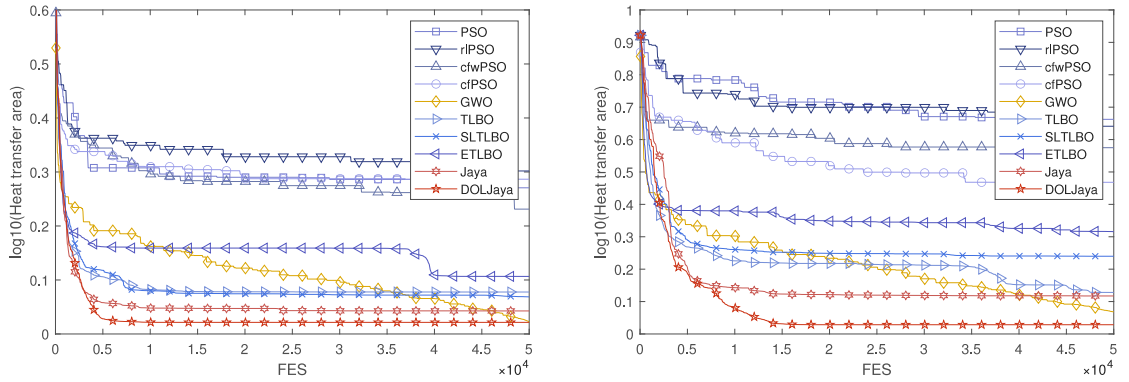


Fig. 10. Convergence results of PFHE design problems with objective function 2 for all the compared algorithms.

exploitation and robustness to changes in the optimization objectives and parameters.

6. Conclusion

In this paper, a new DOLJaya algorithm has been proposed for solving PFHE design problems. It involves combining the DOL strategy with the Jaya algorithm to improve convergence speed and stability. Based on comparing DOLJaya algorithm with Jaya algorithm, we introduce other eight improved algorithms for comparison to ensure fairness. Its performance was compared with nine other optimization algorithms for 28 benchmark optimization problems and four PFHE design case studies. Due to the dynamics and asymmetry characteristics of the DOL strategy, the convergence speed and accuracy of DOLJaya are substantially improved compared with the canonical Jaya algorithm. The proposed DOLJaya achieves accurate results in all PFHE design cases. In future work, the potential of DOL method should be further explored and applied to more practical engineering problems. In light of this, the DOL strategy and DOLJaya method are expected to become efficient and advanced tools especially for solving real-world complex optimization problems.

CRediT authorship contribution statement

Lidong Zhang: Conceptualization, Methodology, Validation, Resources, Supervision, Project administration, Funding acquisition. **Tianyu Hu:** Conceptualization, Methodology, Writing – original Draft, Writing – review & editing, Software, Validation, Formal analysis, data curation. **Linxin Zhang:** Conceptualization, Methodology, Writing – review & editing, Software, Validation, Data curation. **Zhile Yang:** Conceptualization, Methodology, Validation, Resources, Supervision, Writing – review & editing, Project administration, Funding acquisition. **Seán McLoone:** Conceptualization, Methodology, Writing – review & editing, Validation, Formal analysis, Data curation. **Muhammad Ilyas Menhas:** Conceptualization, Methodology, Validation, Writing – review & editing, Resources, Validation. **Yuanjun Guo:** Conceptualization, Methodology, Validation, Resources, Supervision, Writing – review & editing, Validation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This paper is financially supported by the CAS President's International Fellowship Initiative, China (2022VEA0013), National Science Foundation of China under grants 52077213 and 62003332, Natural Science Foundation of Guangdong, China (2018A-030310671), Project (IRT_17R19) supported by the Programme for Chang-jiang Scholars and Innovative Research Team in University, China.

References

Babu, B., Munawar, S., 2007. Differential evolution strategies for optimal design of shell-and-tube heat exchangers. Chem. Eng. Sci. 62 (14), 3720–3739.

Barros, J.J.C., Coira, M.L., de la Cruz López, M.P., del Caño Gochi, A., 2018. Sustainability optimisation of shell and tube heat exchanger, using a new integrated methodology. J. Clean. Prod. 200, 552–567.

Bejan, A., 2013. Entropy Generation Minimization: The Method of Thermodynamic Optimization of Finite-Size Systems and Finite-Time Processes. CRC Press.

Belagoune, S., Bali, N., Atif, K., Labdelaoui, H., 2022. A discrete chaotic Jaya algorithm for optimal preventive maintenance scheduling of power systems generators. Appl. Soft Comput. 119, 108608.

Bhoye, M., Pandya, M., Valvi, S., Trivedi, I.N., Jangir, P., Parmar, S.A., 2016. An emission constraint economic load dispatch problem solution with microgrid using JAYA algorithm. In: 2016 International Conference on Energy Efficient Technologies for Sustainability. ICEETS, IEEE, pp. 497–502.

Çelik, E., 2020a. Improved stochastic fractal search algorithm and modified cost function for automatic generation control of interconnected electric power systems. Eng. Appl. Artif. Intell. 88, 103407.

Çelik, E., 2020b. A powerful variant of symbiotic organisms search algorithm for global optimization. Eng. Appl. Artif. Intell. 87, 103294.

Çelik, E., Öztürk, N., Arya, Y., 2021. Advancement of the search process of salp swarm algorithm for global optimization problems. Expert Syst. Appl. 182, 115292.

Clerc, M., Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. 6 (1), 58–73.

Degertekin, S.O., Tutar, H., 2022. Optimized seismic design of planar and spatial steel frames using the hybrid learning based jaya algorithm. Adv. Eng. Softw. 171, 103172.

Del Valle, Y., Venayagamoorthy, G.K., Mohagheghi, S., Hernandez, J.-C., Harley, R.G., 2008. Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans. Evol. Comput. 12 (2), 171–195.

Dong, H., Zhang, H., Han, S., Li, X., Wang, X., 2018. Reverse-learning particle swarm optimization algorithm based on niching technology. In: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science. ICIS, IEEE, pp. 405–410.

Ergezer, M., Simon, D., Du, D., 2009. Oppositional biogeography-based optimization. In: 2009 IEEE International Conference on Systems, Man and Cybernetics. IEEE, pp. 1009–1014.

Farah, A., Belazi, A., 2018. A novel chaotic Jaya algorithm for unconstrained numerical optimization. Nonlinear Dynam. 93 (3), 1451–1480.

Fesanghary, M., Damangir, E., Soleimani, I., 2009. Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search algorithm. Appl. Therm. Eng. 29 (5–6), 1026–1031.

Gao, K., Yang, F., Zhou, M., Pan, Q., Suganthan, P.N., 2018. Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm. IEEE Trans. Cybern. 49 (5), 1944–1955.

Goldberg, D.E., 2006. Genetic Algorithms. Pearson Education India.

Guo, Y., Yang, Z., Liu, K., Zhang, Y., Feng, W., 2021. A compact and optimized neural network approach for battery state-of-charge estimation of energy storage system. Energy 219, 119529.

Hadidi, A., 2015. A robust approach for optimal design of plate fin heat exchangers using biogeography based optimization (BBO) algorithm. Appl. Energy 150, 196–210.

Huang, C., Wang, L., Yeung, R.S.-C., Zhang, Z., Chung, H.S.-H., Bensoussan, A., 2017. A prediction model-guided Jaya algorithm for the PV system maximum power point tracking. IEEE Trans. Sustain. Energy 9 (1), 45–55.

Incropera, F.P., Lavine, A.S., Bergman, T.L., DeWitt, D.P., 2007. Fundamentals of Heat and Mass Transfer. Wiley.

Janga Reddy, M., Nagesh Kumar, D., 2020. Evolutionary algorithms, swarm intelligence methods, and their applications in water resources engineering: a state-of-the-art review. H2Open J. 3 (1), 135–188.

Joshi, H.M., Webb, R.L., 1987. Heat transfer and friction in the offset stripfin heat exchanger. Int. J. Heat Mass Transfer 30 (1), 69–84.

Kadambur, R., Kotecha, P., 2015. Multi-level production planning in a petrochemical industry using elitist teaching–learning-based-optimization. Expert Syst. Appl. 42 (1), 628–641.

Kakac, S., Liu, H., Pramuanjaroenkij, A., 2020. Heat Exchangers: Selection, Rating, and Thermal Design. CRC Press.

Kaveh, A., Hosseini, S.M., Zaerreza, A., 2021. Improved Shuffled Jaya algorithm for sizing optimization of skeletal structures with discrete variables. In: Structures, Vol. 29. Elsevier, pp. 107–128.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4. IEEE, pp. 1942–1948.

Khatir, S., Boutchicha, D., Le Thanh, C., Tran-Ngoc, H., Nguyen, T., Abdel-Wahab, M., 2020. Improved ANN technique combined with Jaya algorithm for crack identification in plates using XIGA and experimental analysis. Theor. Appl. Fract. Mech. 107, 102554.

Khatir, S., Wahab, M.A., Boutchicha, D., Khatir, T., 2019. Structural health monitoring using modal strain energy damage indicator coupled with teaching-learning-based optimization algorithm and isogoemetric analysis. J. Sound Vib. 448, 230–246.

Li, L.-L., Liu, Y.-W., Tseng, M.-L., Lin, G.-Q., Ali, M.H., 2020. Reducing environmental pollution and fuel consumption using optimization algorithm to develop combined cooling heating and power system operation strategies. J. Clean. Prod. 247, 119082.

Liang, J.J., Qu, B.Y., Suganthan, P.N., 2013. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Vol. 635. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, p. 490.

Liu, K., Li, K., Peng, Q., Guo, Y., Zhang, L., 2018. Data-driven hybrid internal temperature estimation approach for battery thermal management. Complexity 2018, 9642892.

Mallipeddi, R., Suganthan, P.N., 2014. Unit commitment-a survey and comparison of conventional and nature inspired algorithms. Int. J. Bio-Inspired Comput. 6 (2), 71–90.

Mano, T.B., Jiménez, L., Ravagnani, M.A., 2017. Incorporating life cycle assessment eco-costs in the optimization of heat exchanger networks. J. Clean. Prod. 162, 1502–1517.

Mirjalili, S., 2015a. The ant lion optimizer. Adv. Eng. Softw. 83, 80–98.

Mirjalili, S., 2015b. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl.-Based Syst. 89, 228–249.

Mirjalili, S., 2016. SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.

Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.

Mozaffari, M.H., Abdy, H., Zahiri, S.H., 2016. IPO: an inclined planes system optimization algorithm. Comput. Inform. 35 (1), 222–240.

Patel, V., Rao, R., 2010. Design optimization of shell-and-tube heat exchanger using particle swarm optimization technique. Appl. Therm. Eng. 30 (11–12), 1417–1425.

Premkumar, M., Jangir, P., Sowmya, R., Elavarasan, R.M., Kumar, B.S., 2021. Enhanced chaotic JAYA algorithm for parameter estimation of photovoltaic cell/modules. ISA Trans. 116, 139–166.

Qu, B., Zhu, Y., Jiao, Y., Wu, M., Suganthan, P.N., Liang, J., 2018. A survey on multi-objective evolutionary algorithms for the solution of the environmental/economic dispatch problems. Swarm Evol. Comput. 38, 1–11.

Rahnamayan, S., Tizhoosh, H.R., Salama, M.M., 2007. Quasi-oppositional differential evolution. In: 2007 IEEE Congress on Evolutionary Computation. IEEE, pp. 2229–2236.

Rahnamayan, S., Tizhoosh, H.R., Salama, M.M., 2008. Opposition-based differential evolution. IEEE Trans. Evol. Comput. 12 (1), 64–79.

Rao, R., 2016. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int. J. Ind. Eng. Comput. 7 (1), 19–34.

Rao, R., More, K., 2017. Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm. Energy Convers. Manage. 140, 24–35.

Rao, R.V., Patel, V., 2013. Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm. Appl. Math. Model. 37 (3), 1147–1162.

Rao, R.V., Rai, D.P., Balic, J., 2016. Surface grinding process optimization using jaya algorithm. In: Computational Intelligence in Data Mining—Volume 2. Springer, pp. 487–495.

Rao, R.V., Saroj, A., 2017. Economic optimization of shell-and-tube heat exchanger using Jaya algorithm with maintenance consideration. Appl. Therm. Eng. 116, 473–487.

Rao, R.V., Saroj, A., 2019. An elitism-based self-adaptive multi-population Jaya algorithm and its applications. Soft Comput. 23 (12), 4383–4406.

Rao, R.V., Savsani, V.J., Vakharia, D., 2012. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Inform. Sci. 183 (1), 1–15.

Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Inform. Sci. 179 (13), 2232–2248.

Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S., 2010. BGSA: binary gravitational search algorithm. Nat. Comput. 9 (3), 727–745.

Saboori, H., Hemmati, R., Ghiasi, S.M.S., Dehghan, S., 2017. Energy storage planning in electric power distribution networks–A state-of-the-art review. Renew. Sustain. Energy Rev. 79, 1108–1121.

Selbaş, R., Kızılkan, Ö., Reppich, M., 2006. A new design approach for shell-and-tube heat exchangers using genetic algorithms from economic point of view. Chem. Eng. Process.: Process Intensif. 45 (4), 268–275.

Singh, S.P., Prakash, T., Singh, V., Babu, M.G., 2017. Analytic hierarchy process based automatic generation control of multi-area interconnected power system using Jaya algorithm. Eng. Appl. Artif. Intell. 60, 35–44.

Song, R., Cui, M., 2019. Single-and multi-objective optimization of a plate-fin heat exchanger with offset strip fins adopting the genetic algorithm. Appl. Therm. Eng. 159, 113881.

Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.

Tran-Ngoc, H., Khatir, S., Ho-Khac, H., De Roeck, G., Bui-Tien, T., Wahab, M.A., 2021. Efficient Artificial neural networks based on a hybrid metaheuristic optimization algorithm for damage detection in laminated composite structures. Compos. Struct. 262, 113339.

Trivedi, I.N., Purohit, S.N., Jangir, P., Bhoye, M.T., 2016. Environment dispatch of distributed energy resources in a microgrid using JAYA algorithm. In: 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics. AEEICB, IEEE, pp. 224–228.

Wang, Z., Li, Y., 2015. Irreversibility analysis for optimization design of plate fin heat exchangers using a multi-objective cuckoo search algorithm. Energy Convers. Manage. 101, 126–135.

Wang, Z., Li, Y., 2016a. A combined method for surface selection and layer pattern optimization of a multistream plate-fin heat exchanger. Appl. Energy 165, 815–827.

Wang, Z., Li, Y., 2016b. Layer pattern thermal design and optimization for multistream plate-fin heat exchangers—A review. Renew. Sustain. Energy Rev. 53, 500–514.

Warid, W., Hizam, H., Mariun, N., Abdul-Wahab, N.I., 2016. Optimal power flow using the Jaya algorithm. Energies 9 (9), 678.

Wilcoxon, F., 1992. Individual comparisons by ranking methods. In: Breakthroughs in Statistics. Springer, pp. 196–202.

Wong, J.Y., Sharma, S., Rangaiah, G., 2016. Design of shell-and-tube heat exchangers for multiple objectives using elitist non-dominated sorting genetic algorithm with termination criteria. Appl. Therm. Eng. 93, 888–899.

Xie, G.N., Sunden, B., Wang, Q.W., 2008. Optimization of compact heat exchangers by a genetic algorithm. In: Applied Thermal Engineering. Elsevier, pp. 895–906.

Xu, Y., Yang, Z., Li, X., Kang, H., Yang, X., 2020. Dynamic opposite learning enhanced teaching–learning-based optimization. Knowl.-Based Syst. 188, 104966.

Yang, Z., Li, K., Guo, Y., Feng, S., Niu, Q., Xue, Y., Foley, A., 2019a. A binary symmetric based hybrid meta-heuristic method for solving mixed integer unit commitment problem integrating with significant plug-in electric vehicles. Energy 170 (MAR.1), 889–905.

Yang, Z., Li, K., Niu, Q., Xue, Y., 2017. A comprehensive study of economic unit commitment of power systems integrating various renewable generations and plug-in electric vehicles. Energy Convers. Manage. 132 (1), 460–481.

Yang, Z., Liu, K., Fan, J., Guo, Y., Niu, Q., Zhang, J., 2019b. A novel binary/real-valued pigeon-inspired optimization for economic/environment unit commitment with renewables and plug-in vehicles. Sci. China 62, 070213:1–070213:3.

Yılmaz, Z.Y., Bal, G., Celik, E., Öztürk, N., Güvenç, U., Arya, Y., 2021. A new objective function design for optimization of secondary controllers in load frequency control. J. Fac. Eng. Archit. Gaz. 36 (4), 2053–2067.

Yousefi, M., Darus, A., Mohammadi, H., 2012. An imperialist competitive algorithm for optimal design of plate-fin heat exchangers. Int. J. Heat Mass Transfer 55 (11–12), 3178–3185.

Yousefi, M., Enayatifar, R., Darus, A.N., Abdullah, A.H., 2013. Optimization of plate-fin heat exchangers by an improved harmony search algorithm. Appl. Therm. Eng. 50 (1), 877–885.

Yu, J.-t., Kim, C.-H., Wadood, A., Khurshaid, T., Rhee, S.-B., 2019. Jaya algorithm with self-adaptive multi-population and Lévy flights for solving economic load dispatch problems. IEEE Access 7, 21372–21384.

Zhang, Y., Chi, A., Mirjalili, S., 2021a. Enhanced Jaya algorithm: A simple but efficient optimization method for constrained engineering design problems. Knowl.-Based Syst. 233, 107555.

Zhang, L., Hu, T., Yang, Z., Yang, D., Zhang, J., 2021b. Elite and dynamic opposite learning enhanced sine cosine algorithm for application to plat-fin heat exchangers design problem. Neural Comput. Appl. 1–14.

Zhao, F., Ma, R., Wang, L., 2021. A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system. IEEE Trans. Cybern..

Zhile, Y., Kang, L., Qun, N., Yusheng, X., Foley, A., 2014. A self-learning TLBO based dynamic economic/environmental dispatch considering multiple plug-in electric vehicle loads. J. Mod. Power Syst. Clean Energy 2 (4), 298–307.