Moth-Flame Optimization Algorithm: Theory, Literature Review, and Application in Optimal Nonlinear Feedback Control Design

Seyed Hamed Hashemi Mehne and Seyedali Mirjalili

Abstract A direct numerical method for optimal feedback control design of general nonlinear systems is presented in this chapter. The problem is generally infinite dimensional. In order to convert it to a finite dimensional optimization problem, a collocation type method is proposed. The collocation approach is based on approximating the control input function as a series of given base functions with unknown coefficients. Then, the optimal control problem is converted to the problem of finding a finite set of coefficients. To solve the resulting optimization problem, a new nature-inspired optimization paradigm known as Moth Flame Optimizer (MFO) is used. Validation and evaluating of accuracy of the method are performed via implementing it on some well known benchmark problems. Investigations presented in this chapter reveals the efficiency of the method and its benefits with respect to other numerical approaches. The chapter also consideres an in-depth literratur review and analysis of MFO.

1 Introduction

One of the fundamental and practical problems in control theory is to design controllers that minimize a predefined performance index estimating the success of the control inputs. This problem is known as optimal control problem. Exact or analytical solution may be found using necessary optimality conditions for linear or other simple systems. However, solving the equivalent boundary value problems arising

S. H. H. Mehne
Aerospace Research Institute, 1465774111 Tehran, Iran
e-mail: hmehne@ari.ac.ir

S. Mirjalili (✉)
School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia
e-mail: seyedali.mirjalili@griffithuni.edu.au

S. Mirjalili et al. (eds.), *Nature-Inspired Optimizers*, Studies in Computational Intelligence 811, https://doi.org/10.1007/978-3-030-12127-3_9

from optimality conditions is a formidable task in general, especially for non-linear systems. Therefore, approximate or numerical methods have been developed for solving optimal control problems by researchers. Due to the dynamic of the system and available tools, different methods exit which may be categorized in three classes as follow:

Approximating the Dynamical System: One of the popular procedures in finding the solution of optimal control problems is approximating or converting the underlying dynamical system with one or a set of simple dynamical systems with known or easy to find solutions. As example, a method for approximating nonlinear optimal feedback controllers for a class of general nonlinear systems was proposed in [1]. The method is based on successive linear time varying approximations. In [2], the underlying system was transformed into the Jurdjevic-Quinn type system using singular perturbation and zero dynamics theory. Then, with characteristic of this type of systems and Lyapunov theory, an optimal stabilization control law was derived. A method for approximate the stabilizing control law as the solution of another system of two partial differential equations was also presented in [3].

Indirect Methods: In the case that sufficient conditions of optimality such as Pontryagins maximum principle or Hamilton-Jacobi-Bellman equations are utilized, the method is called indirect. Using these conditions transfers the optimal control to a two point boundary value problem which may be easier to solve. For example of indirect methods, one may address [4], where a computational method for obtaining feedback optimal control for nonlinear systems based on the Hamilton-Jacobi-Bellman equation was reported. Using Pontryagins maximum principle in [5] a method for an optimal control problem of Mars landing in powered descent was derived. The method is based on reduction of the original problem to a two point boundary problem and real-time sampling optimal feedback control theory.

Indirect methods are usually computationally inexpensive and accurate. However, the available methods for solving the boundary value problem are dependent on initial guess of the solution that restricts the domain of convergence.

Direct Methods: With direct method we means an approach that transfers the optimal control problem to an optimization problem. It may be performed by collocation (or parametrization) and discretization (or time-marching). Some recent examples of direct methods are reviewed here. A direct method for designing feedback controllers that maximize the set of points that reach a given target set in specified finite time was given in [6]. The method is based on convex optimization in the form of finite dimensional linear programming problem. In [7] a numerical method for missile formation problem is introduced that is based on direct discretization of feedback control and using nonlinear programming. A numerical method using piece-wise control action was also reported in [8] for optimal control problems. Discretization of the control function leads to a nonlinear programming problem which has been solved by gradient based methods. A pseudo-spectral method was given in [9] where, state parametrization using Lagrange polynomials at special points converts the problem to an optimization one. Ease of implementation and wider convergence range are benefits of direct methods with respect to indirect ones.

Contribution of the Present Work: In this chapter, a numerical method for solving optimal control problems will be given. The control system is closed loop and optimal feedback controller for non-linear systems is under investigation. The method is in the category of direct methods based on collocation of the control. The first step is transferring the original problem to a finite dimensional optimization problem. This is performed by looking at the controller as a combination of same known functions and unknown coefficients. The resulting problem is a finite dimensional optimization problem. The second step is to solve this optimization problem by moth flame optimization algorithm that is an efficient nature-inspired optimizer. In comparison with discretization based methods like [10], the proposed method has lower dimension. Therefore, it is fast, accurate and with simple implementation. The method is tested on some benchmark problems in order to evaluation and validation. Results show that the proposed method is effective and accurate in practice.

Organization: The rest of this chapter is organized as follows: In Sect. 2, the MFO algorithm is reviewed and explained. Sect. 3 presents a litrature review of MFO. The optimal control problem is introduced and its application are noticed in Sect. 4 and MFO is tailored for optimal control problems with feedback control. Numerical examples and results are discussed in Sect. 5. Finally some conclusion remarks and future work suggestions are given in Sect. 6.

2 Moth Flame Optimization Method

2.1 Background and Motivation

The problem of finding the best situation among different choices is formulated as an optimization problem. Such a problem has objective function, decision variables and constraints as its pillars. Traditional methods for solving optimization problems such as Newton-Raphson, quasi-Newton, and conjugated gradient are based on gradient calculation. Theses methods are usually time consuming for complex problems in real world applications due to presence of partial derivatives. Therefore, search methods like golden search, Nelder-Mead and Tabu search have been developed which are just based on the objective function evaluations.

Among search methods, meta-heuristic optimization algorithms were developed imitating physical phenomena or nature behaviors. Meta-heuristic optimizers are popular in engineering applications because they are easy to implement, do not require gradient calculation, converge to global optima, and can be utilized in a wide range of problems covering different disciplines[11]. Some of the recent developed meta-heuristics are Whale Optimization [11], Dragonfly algorithm [12], Salp Swarm Algorithm [13], Colliding Bodies Optimization [14], and Thermal exchange optimization [15].

Moth-flame optimization algorithm is also a novel nature-inspired algorithm. This algorithm simulates the navigating mechanism of moths in nature known as trans-

verse orientation. Since its development, it has been used to solve different engineering optimization problems such as optimization of electrical power generation systems [16], estimation of model parameters in solar cells [17], estimating the weights and parameters in neural networks [18], image processing [19], forecasting on annual electricity consumption [20], optimizing the manufacturing processes [21], and bio-informatics [22, 23].

2.2 Explanation of MFO

The moth flame optimization method is introduced and explained in this section. For more details about this optimizer the reader is referred to [24].

2.3 Inspiration from Moths

As stated before, MFO is a nature-inspired optimization method proposed by S. Mirjalili in 2015. The method simulates the behavior of moths for navigating in night. They fly at night using the moon based on a mechanism called transverse orientation. Actually, the moths maintain a fixed angle to the moon when flying as shown in Fig. 1. Because of long distance from earth to moon (384,400 km), this mechanism helps them to fly in straight line. However, in the face of artificial lights such as electrical lamps or candles, using transverse orientation causes a spiral path to the light source. This is because of the short distance from moth to this sources of lights when maintaining the same angle. Being the base of the MFO method, this fact was depicted in Fig. 2. In the case of planar flying for one dimensional case, the mathematical equation of such a logarithmic spiral is as follows:

$$S = De^{bt}cos(2\pi t) + F \tag{1}$$

where, F is the position of flame, M is the position of moth, $D = |M - F|$ is the distance between moth and flame, and $t \in [-1, 1]$ is a time variable. The above equation constructs the path from moth to flame when t decreases from -1.5 to 1 as shown in Fig. 3. The constant b determines the rate of reaching to flame.

2.4 The MFO Algorithm

Let assume that the underlying optimization problem is d-dimensional dimensional, that is there exit d decision variables with desired optimum values. As the MFO is a population-base method, we need some search agents. A set of-n moths is the population of search agents, where each moth keeps a value of a d dimensional

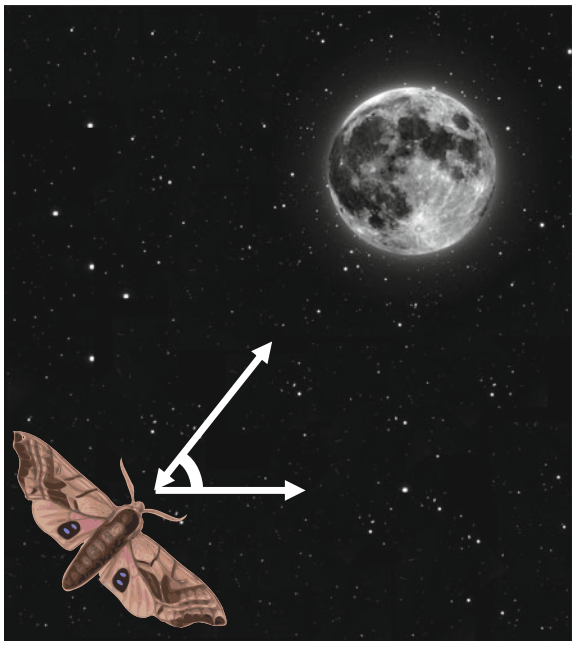

Fig. 1 Transverse orientation of moths

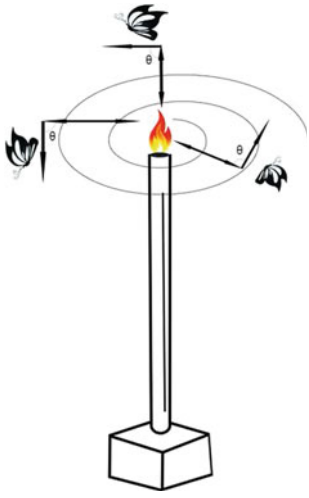

Fig. 2 Spiral path of flying around a close light source

vector as a candidate solution for the problem. In other words, moths are flying in the d dimensional space and their positions are possible solutions. Therefore, the set of-moths and positions may be described by the following $n \times d$ matrix:

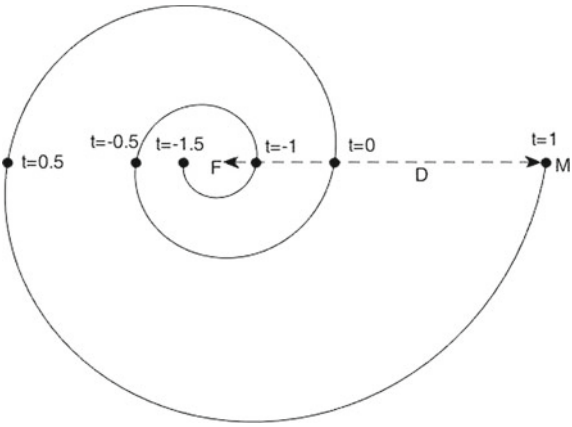

Fig. 3 A typical logarithmic spiral path around a flame with some positions with respect to t

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & \cdots & M_{1d} \\ M_{21} & M_{22} & \cdots & \cdots & M_{2d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{n1} & M_{n2} & \cdots & \cdots & M_{nd} \end{bmatrix} \tag{2}$$

All fitness values are also stored in a vector as:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{3}$$

where, each element shows the fitness value of the corresponding moth's position. There are also a set of flames indicating the best solution that every moths has found so far. Flames have a similar matrix representation as follows:

$$F = \begin{bmatrix} F_{11} & F_{12} & \cdots & \cdots & F_{1d} \\ F_{21} & F_{22} & \cdots & \cdots & F_{2d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n1} & F_{n2} & \cdots & \cdots & F_{nd} \end{bmatrix} \tag{4}$$

In the same way, the set of the corresponding best fitness values is stored to a vector as follows:

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{5}$$

Therefore, flames can be interpreted as flags or pins in the search space that are dropped by moths when searching. A nominal moth flies and searches around a flame for better solution. It updates the flam position when such a better solution is found.

From mathematical viewpoint, the MFO consists of three parts as:

$$MFO = (I, P, T) \tag{6}$$

where, $I : \emptyset \rightarrow \{M, OM\}$ is a function that generates a random population of moths and corresponding fitness values at the beginning. $P : M \rightarrow M$ is the update function that receives the current M matrix and updates it each iteration. $T : M \rightarrow \{True, False\}$ is a termination function that returns *True* when the termination condition happens and returns *False* when termination doesn't attain.

Therefore, the MFO is expressed in brief as follows:

$M = I()$; **while** $T(M)$ *is equal to False*
  $M = P(M)$;
end

The function I generates random initial solutions and calculates the objective function values. The following method is utilized as default:

for $i = 1 : n$
 for $j = 1 : d$
 $M_{ij} = (ub_i - lb_i) * rand() + lb(i)$
 end
end

where, ub and lb are two vectors including upper and lower bound for decision variables as follow:

$$ub = [ub_1, ub_2, \ldots, ub_d] \tag{7}$$

$$lb = [lb_1, lb_2, \ldots, lb_d] \tag{8}$$

After function I generates the initial solution, the iteration can be started and the function P is run in iterations until T function announces the termination. The P function is the main part of the MFO that moves moths around the search space and updates the position of flames. The position of i-th moth with respect to j-th flame is updates based on the following equations:

$$M_i = S(M_i, F_j) = D_{ij} e^{bt} cos(2\pi t) + F_j \tag{9}$$

$$D_{ij} = |M_i - F_j| \tag{10}$$

As stated before, this is a logarithmic spiral from moth to the flame. Figure 4 shows four different possible points in a typical spiral in one dimension (dashed black lines). These positions can be chosen in the next iteration as moth position (horizontal line) around the flame (vertical line) depending the random variables. Therefore, a moth can explore and exploit the search space. Exploration in general encourages the candidate solutions to change its moving direction. Consequently, the exploration helps methods to prevent from trapping in local optima. Here, the exploration occurs when the new position is outside the space between the moth and flam as can be seen in the arrows labeled by 1, 3, and 4. The exploitation phase tends to improve the quality of solutions by searching locally around the obtained promising solutions in the exploration milestone. Exploitation in MFO happens when the new position lies inside the space between the moth and flame as can be observed in the arrow labeled by 2. It is to be noted that each moth can update its position with respect to only one flame at each iteration. After updating the list of flames, they are sorted based on their fitness values from the best to the worst. Then the moths update their positions according to corresponding flame. This means that for example the first moth is updated with respect to the flame with best fitness value, and so on. More over, to prevent form degrading the exploitation of the best promising solution, an adapting mechanism was proposed in [24]. This mechanism decreases the number of flames during iteration based on the following relation:

$$\text{Flame number} = \textit{round}\left(N - l * \frac{N-1}{T}\right) \tag{11}$$

where, l is the iteration counter, T is the maximum number of iterations, and N is the maximum number (or initial number) of flames. Now, by the above mentioned description, the algorithm of function P is as follows (Fig. 5):

Update flame number using Eq. (9.11)
$OM = FitnessFunction(M)$;
if *iteration* $==1$
  $F = sort(M)$;
  $OF = sort(OM)$;
else
  $F = sort(M_{t-1}, M_t)$;
  $OF = sort(M_{t-1}, M_t)$;
end
for $i = 1 : n$
  for $j = 1 : d$
    Update r and t
    Calculate D_{ij} *using Eq.*(9.10) *with respect to the corresponding moth*
    Update M_{ij} *using Eq.*(9.9) *with respectto the corresponding moth*
  end
end

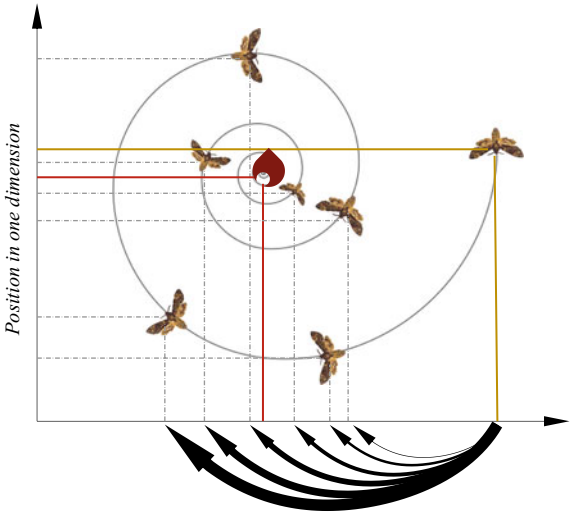

Fig. 4 Some of the possible positions that can be reached by a moth with respect to a flame using the logarithmic spiral [24]

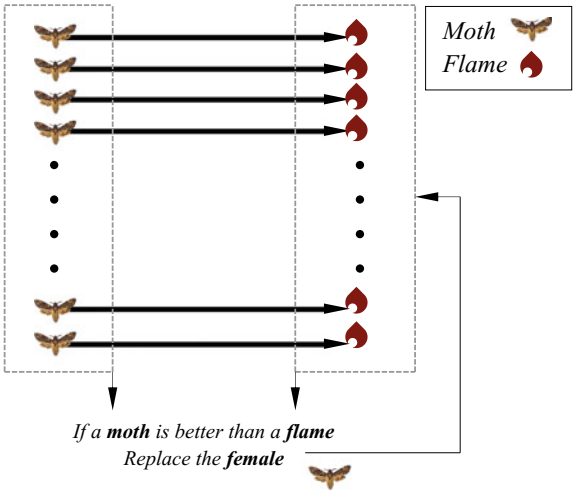

Fig. 5 Moth-Flame assignment [24]

3 Literature Review of MFO

This section provides the literature review of the MFO algorithm. It first covers different variants of this algorithm to solve problems of different types. Then, the applications of this algorithm in diverse fields are given.

3.1 Variants

This original version of MFO has been designed to solve problems with continuous variables. To solve binary problems with this algorithm, Reddy et al. [25] employed a sigmoid function to map continuous domain to binary domain. Both continuous and binary MFO can solve single-objective problems. This motivated the attempt of several researchers to developed the multi-objective version of this algorithm. For instance, a non-dominated sorting multi-objective MFO is proposed in [26]. In a similar work, an archive is used to require MFO to solve multi-objective problems [27]. Another variant of MFO was proposed in [28, 29], in which several penalty functions were used to solve constrained problems using MFO.

There are also several works in the literature to improve exploration and/or exploitation of MFO. In [30], an opposition-based learning version of MFO was propsoed by considering the opposite location of moths during the optimization process. There are three works that use chaotic maps to tune different controlling paramters of MFO to improve its performance as well [31–33]. Several Levy flights are also used to mostly boost exploration and local optima avoidance of the MFO algorithm [34, 35].

Another direction that several authors have taken is to hybridize MFO with other algorithms. There are different ways to hybridize two algorithms in the field of evolutionary computation. Most of them are used to improve the performance of MFO as follows:

- Hybrid MFO and Simulated Annealing [36]
- Hybrid MFO and Particle Swarm Optimization [37–39]
- Hybrid MFO and Gravitational Search Algorithm [40]
- Hybrid MFO and Firefly Algorithm [41]

Another popular area that MFO has been largely is Machine Learning. The problem of training different machine learning techniques is a challenging optimization problem. Gradient-based algorithms normally fail to solve such problems since they suffer from local optima stagnation and they require gradiant information of the problem. For instance, the MFO algorithm is used to find the optimal values for the main parameters of Support Vector Machine in [42]. This algorithm has been also used to train NN including finding optimal values for the connections weights, biases, and number of hidden nodes [43–46].

3.2 Applications

MFO has been applied to a large number of applications in diverse fields. The following list shows some of them:

- Breast cancer detection [47, 48]
- Optimal sizing and placement of capacitors in radial distribution systems [49]

- AGC system design [50–53]
- Clustering for Internet of Things [54, 55]
- Automatic test generation technique for software testing [56, 57]
- Dynamic economic load dispatch problem [58, 59]
- Geographic atrophy segmentation for SD-OCT images [60]
- Power dispatch problems [61–64]
- Optimal reactive power dispatch problem[65]
- Optimal allocation of shunt capacitor banks in radial distribution systems [66]
- PID controller for optimal control of active magnetic bearing System [67]
- Devices allocation in power systems [68]
- Optimal allocation of distributed generations and capacitor banks in distribution grids [69]
- Profit maximization with integration of wind farm [70]
- Dynamic performance enhancement for wind energy conversion [71]
- Web service composition in cloud computing [72]
- Image segmentation (kidney images, satellite images, etc.) [73–76]
- Optimization of water resources [77]
- Antenna array design [78]
- Optimal power tracking [79]
- Accuracy control of contactless laser sensor system [80]
- Optimal placement and sizing of distributed generation units for power loss Reduction [81]
- Range image registration [82]
- PID controller design [83]
- Alzheimers disease diagnosis [84]
- Optimal bidding strategy under transmission congestion in deregulated power market [85]
- Production planning in petrochemical industries [86]
- Terrorism prediction [87]
- Optical network unit placement in Fiber-Wireless (FiWi) access network [88]
- Hybrid power generation systems [89]
- Arabic handwritten letter recognition [90]
- Forecasting [91]
- Feature selection [92]

4 MFO for Optimal Control Problems

In this section, the form of the underlying optimal control problem is introduced. Then the method of collocation is explained and the way of implementing the MFO algorithm is discussed.

4.1 Problem Definition

The problem of designing controllers to optimally stabilize an equilibrium of a control system is a problem in the field of control with many real world applications. When the control law is based on knowledge of the system response, the controller called feedback. Feedback control is preferred over open loop solutions due to robustness to disturbances and reduction in computational time [93]. Some applications that have been reported in literature are rotor speed control in wind turbines [94], optimal stabilization and attitude tracking of satellites [2], coordinate control of turbine boilers [95], and optimal chemotherapy administration for the cancer treatment [1]. The problem of designing such a of control law is investigated in sequel.

Let us revisit the general form of a non-linear optimal control problem in which we intend to minimize a cost functional as

$$J(u) = h(x(T_f)) + \int_0^{T_f} f(x, u)dt \tag{12}$$

where, the control input $u \in \mathbb{R}^m$ and state vector $x \in \mathbb{R}^n$ are related to each other by the following dynamical system and initial condition:

$$\dot{x} = g(t, x, u), \quad x(0) = x_0 \tag{13}$$

Some of special and practical cases of this problem are linear quadratic cost function as:

$$J(u) = \frac{1}{2}x(T_f)^T Hx(T_f) + \frac{1}{2}\int_0^{T_f} (x(t)^T Qx(t) + u(t)^T Ru(t))dt \in \mathbb{R} \tag{14}$$

where, $H, Q \in \mathbb{R}^{n\times n}$ are positive semidefinite matrices and $R \in \mathbb{R}^{m\times m}$ is a positive definite matrix.

Linear systems in which,

$$\dot{x} = Ax(t) + Bu(t) \tag{15}$$

where, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are given and may be constant in invariant systems or variable with time in time varying systems.

Affine non-linear systems that has the following form:

$$\dot{x} = G(x(t)) + g(x(t))u(t) \tag{16}$$

The control values may be bounded as:

$$u_l \leq u(t) \leq u_b \quad \forall t \in [0, T_f] \tag{17}$$

where, u_l and u_b are constant vectors with the same dimension as $u(t)$.

4.2 Collocation

In the case of searching for feedback optimal control, the control can be considered as a function of the state, that is $u = u(x)$. For control parametrization we need a set of independent functions as base functions. They may be monomials, Chebyshev polynomials, geometric or exponential functions. Let

$$\Phi(x) = \left[\, \phi_1(x)\ \phi_2(x)\ \cdots\ \phi_d(x) \,\right]^T \tag{18}$$

is such a base. Then, the parametrized control function is a linear combination of the elements of this base. Then, a typical control function has the following form:

$$u^d(x) = \sum_{i=1}^{d} c_i \phi_i(x) = c\Phi(x)^T \tag{19}$$

where, $c = [c_1, c_2, \ldots, c_d]$ is a real value vector of coefficients determining the control. In other words, every selection of these coefficients constructs a control function. As the control is selected, the corresponding state will be obtained by solving Eq. (13) numerically for this control input. By this transformation, the problem of looking for the best control function is reduced to the problem of looking in the following subset of d-dimensional Euclidean space:

$$C = \{c = (c_1, c_2, \ldots, c_d) \in \mathbb{R}^n \mid u_l \leq u^d(x) \leq u_b\} \tag{20}$$

On the other hand, the objective function will be dependent implicitly on c and therefore, the original problem is transfered to an optimization problem. In the proposed method, the MFO is utilized to solve this optimization problem. Figure 6 describes the method as a block diagram showing the position of MFO in generating the vector c of coefficient, managing the iterations and stopping condition. In the next section the method will be evaluated via practical and standard case studies.

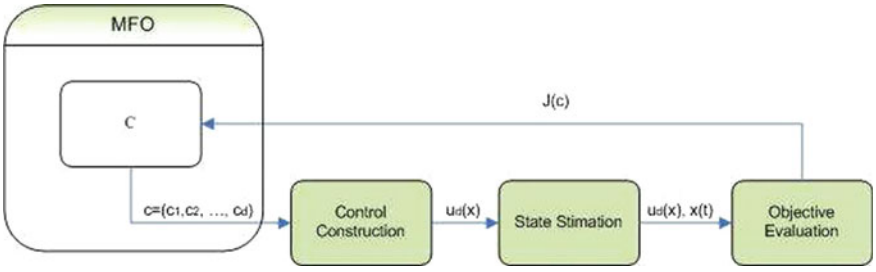

Fig. 6 Flowchart of the proposed method

5 Numerical Examples

In this section, the method of Sect. 4 is implemented on some benchmark optimal control problems to evaluate performance, accuracy and efficiency of this method.

5.1 Example 1. Rotational/Translational Actuator

Rotational/Translational actuator (RTAC) is a well known nonlinear benchmark problem in the field of feedback control. It has been used to validated and evaluate the numerical and analytical methods for deriving control laws in literature. As depicted schematically in Fig. 7, RTAC is a mechanical system consisting of a translational oscillator with an eccentric rotational proof-mass actuated by a DC motor. The rotational actuator is used to stabilize the translational motion. It was originally proposed to represent a simplified model of a dual spin spacecraft. Due to different application of this system, it has been studied as an individual system in literature. For example, in [96] a least square method was applied to estimating the parameters of RTAC system and by using experimental data, a linear system representation for RTAC was introduced using system identification. Dynamical analysis and designing of stabilizing controls of RTAC system in the case that the installation plane is not horizontal was reported in [97]. The RTAC system is also used to check the new methods in control. For example, it was used as a benchmark problem in [98] for evaluating a fuzzy reinforced learning based controller. In the present work, the proposed method is applied on this system for evaluation.

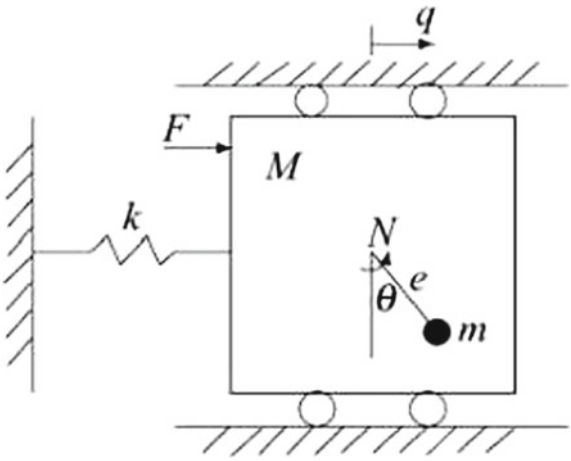

Fig. 7 Translational oscillator with rotational actuator [96]

The RTAC motion in non-dimensional form is represented by the following four dimensional nonlinear system of equations [99]:

$$\dot{x}_1 = x_2 \tag{21}$$

$$\dot{x}_2 = \frac{-x_1 + \varepsilon x_4^2 \sin x_3}{1 - \varepsilon^2 \cos^2 x_3} + \frac{-\varepsilon \cos x_3}{1 - \varepsilon^2 \cos^2 x_3} u \tag{22}$$

$$\dot{x}_3 = x_4 \tag{23}$$

$$\dot{x}_4 = \frac{\varepsilon \cos x_3 (x_1 - \varepsilon x_4^2 \sin x_3)}{1 - \varepsilon^2 \cos^2 x_3} + \frac{1}{1 - \varepsilon^2 \cos^2 x_3} u \tag{24}$$

where, disturbance is ignored and
x_1 and x_2 are normalized position and velocity of the platform.
x_3 and x_4 are the angular position and velocity of the rotating mass.
u is the non-dimensional control torque.

The problem was solved by the proposed method and using MFO. The base functions for parametrization were chosen similar to [100]. They are x_1, x_2, x_3 and x_4 with pairwise combinations of degree lower than 3. The MFO parameters, including number of iterations, b, and number of moths are respectively 500, 1, and 30. Table 1 shows the base functions and the resulting coefficient for each of them in the optimal combination. The resulting feedback control function was shown in Fig. 8. As expected it has an oscillating form which damps with time. Using the control function of Fig. 8 as a control input to the system (21)–(24) and solving equations numerically by Runge-Kutta method, the system responses were found as depicted in Fig. 9. It is clear that the resulting control stabilizes all the states as desired. This shows that the proposed method works well in this case and the results are valid.

Table 1 Base functions and their resulting optimal coefficients for Example 1

i	$\phi(i)$	c_i	i	$\phi(i)$	c_i
1	x_1	5.0000	10	x_2x_3	5.0000
2	x_2	−1.1159	11	x_2x_4	−3.2979
3	x_3	−1.1902	12	x_3^2	4.1480
4	x_4	−4.0401	13	x_3x_4	4.3387
5	x_1^2	1.0678	14	x_4^2	−0.1579
6	x_1x_2	5.0000	15	x_1^3	−5.0000
7	x_1x_3	−5.0000	16	x_2^3	−5.0000
8	x_1x_4	−0.9885	17	x_3^3	−5.0000
9	x_2^2	−3.7699	18	x_4^3	5.0000

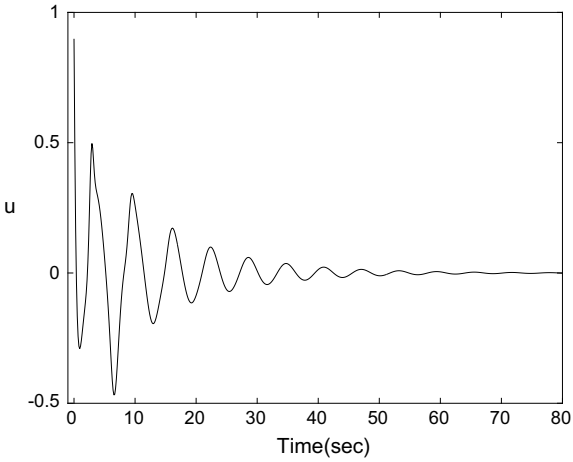

Fig. 8 The resulting optimal feedback control of the RTAC system

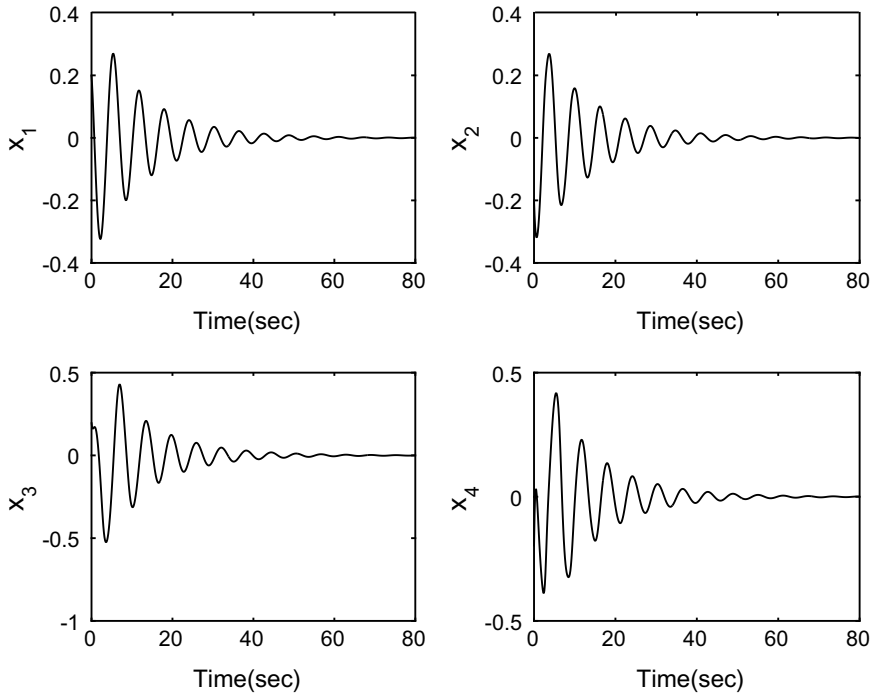

Fig. 9 The response of the RTAC system to the resulting control function

5.2 Example 2. F-8 Aircraft

The equations of F-8 dynamics in certain conditions is descried by the following control system [101]:

$$\dot{x}_1 = -0.877x_1 + x_3 - 0.088x_1x_3 + 0.47x_1^2 - 0.019x_2^2 - x_1^2x_3 + 3.846x_1^3 - 0.215u + 0.28x_1^2u + 0.47x_1u^2 + 0.63u^3 \quad (25)$$

$$\dot{x}_2 = x_3 \quad (26)$$

$$\dot{x}_3 = -4.208x_1 - 0.396x_3 - 0.47x_1^2 - 3.564x_1^3 - 20.967u + 6.265x_1^2u + 46x_1u^2 + 61.4u^3 \quad (27)$$

where, x_1 denotes the angle of attack (rad), x_2 is the pitch angle (rad), and x_3 shows the pitch rate (rad/s). The control input u is related to the elevator angle (rad). The aim of this example is to provide a control law that stabilizes an initial disturbance in the angle of attack in addition to minimizing LQR performance index as EQ. (14). The problem has been solved by the proposed method with data given from [101] such as $T_f = 5(sec)$, $H = 0.1I_3$, $Q = 0.01I_{3\times3}$, and $R = 1$. Initial condition indicating the disturbance is $x(0) = (0.56, 0, 0)$. For parametrization, 12 different forms of monomials were utilized. Number of iterations was 100, b were set to 1, and the number of moths was 30. Table 2 shows the base functions and resulting optimal coefficients. The control function and the response of the system of the first channel have been depicted in Fig. 10. It turns out from the figure that the control function is successful in stabilizing the angle of attack and elimination of the initial disturbances. The optimal performance index was $J^* = 0.035$ and results are agreed with the results of [101]. Therefore, the proposed method has the ability and accuracy to derive control laws in practical, complex, and nonlinear systems.

Table 2 Base functions and their resulting optimal coefficients for Example 2

i	$\phi(i)$	c_i	i	$\phi(i)$	c_i
1	x_1	0.1103	10	x_2^2	−0.2600
2	x_2	0.1441	11	x_2x_3	−0.2600
3	x_3	−0.0574	12	x_3^2	0.0800
4	x_1^2	0.2600	14	x_1^3	0.2600
5	x_1x_2	−0.1271	15	x_2^3	−0.2600
6	x_1x_3	0.2600	16	x_3^3	0.1174

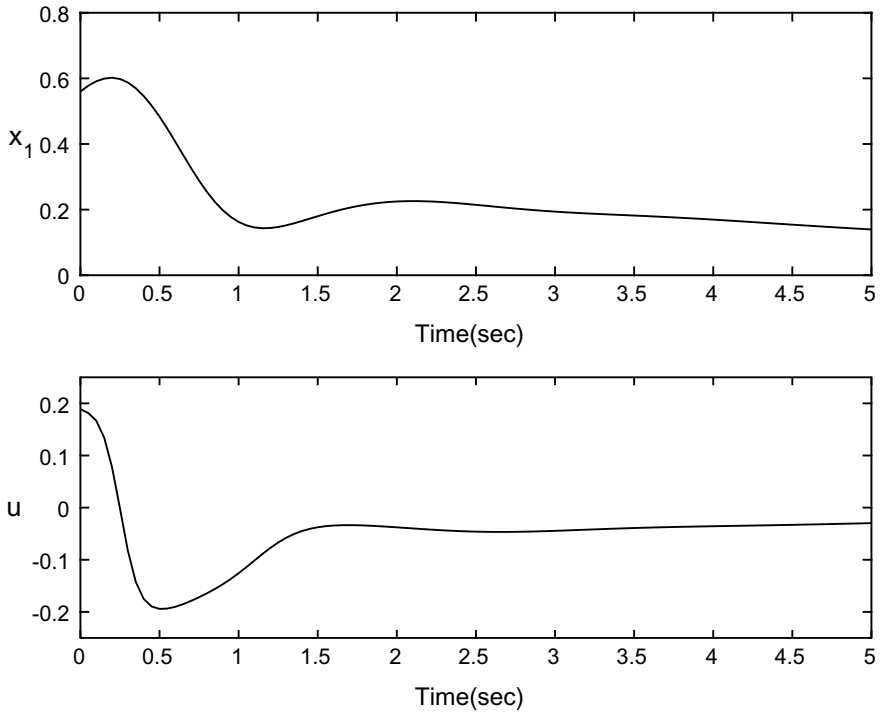

Fig. 10 The resulting optimal control and the angle of attack in Example 2

6 Conclusion

A direct numerical method for obtaining optimal feed back controls was presented. The method uses control parametrization and benefits from moth-flame optimization algorithm features. Control parametrization decreases the size of the optimization problem in comparison with discretization in the time domain. Therefore, the resulting method has lower computational complexity, it is easy to implement, fast, and doesn't care about system nonlinearities. It has been implemented and tested on two practical problems. Results show that the method could derive stabilizing controls in optimum situations. For further works, trying it on GPU structures is suggested.

Acknowledgements Authors would like to thank Mr. Farhad Karimzadeh for performing some of the graphical tasks.

References

1. Unal, C., & Salamci, M. U. (2018). Drug administration in cancer treatment via optimal nonlinear state feedback gain matrix design. *IFAC Papersonline*, *50*, 9979–9984.
2. Zhang, B., Liu, K., & Xiang, J. (2013). A stabilized optimal nonlinear feedback control for satellite attitude tracking. *Aerospace Science and Technology*, *27*, 17–24.
3. Mylvaganam, T., & Sassano, M. (2017). Approximate optimal control via measurement feedback for a class of nonlinear systems. *IFAC Papersonline*, *50*, 15391–15396.
4. Zhu, J. (2017). A feedback optimal control by Hamilton-Jacobi-Bellman equation. *European Journal of Control*, *37*, 70–74.
5. Zheng, Y., & Cui, H. (2015). Optimal nonlinear feedback guidance algorithm for Mars powered descent. *Aerospace Science and Technology*, *45*, 359–366.
6. Majumdar, A., Vasudevan, R., Tobenkin, M. M., & Tedrake, R. (2014). Convex Optimization of nonlinear feedback controllers via occupation measures. *The International Journal of Robotics Research*, *33*, 1209–1230.
7. Yun-jie, W., Futao, Z., & Chuang, S. (2017). Optimal discretization of feedback control in missile formation. *Aerospace Science and Technology*, *67*, 456–472.
8. Armaoua, A., & Ataei, A. (2014). Piece-wise constant predictive feedback control of nonlinear systems. *Journal of Process Control*, *24*, 326–335.
9. Xiao-Jun, T., Jian-Li, W., & Kai, C. (2015). A Chebyshev-Gauss pseudospectral method for solving optimal control problems. *Acta Automatica Sinica*, *41*, 1778–1787.
10. Mehne, S. H. H., & Mirjalili, S. (2018). A parallel numerical method for solving optimal control problems based on whale optimization algorithm. *Knowl-Based System*, *151*, 114–123.
11. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.
12. Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, *27*, 1053–1073.
13. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163–191.
14. Kaveh, A., & Mahdavi, V. R. (2014). Colliding bodies optimization: A novel meta-heuristic method. *Computers Structures*, *139*, 18–27.
15. Kaveh, A., & Dadras, A. (2017). A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software*, *110*, 69–84.
16. Mohamed, A. A., Mohamed, Y. S., El-Gaafary, A. A. M., & Hemeida, A. M. (2017). Optimal power flow using moth swarm algorithm. *Electric Power Systems Research*, *142*, 190–206.
17. Allam, D., Yousri, D. A., & Eteiba, M. B. (2016). Parameters extraction of the three diode model for the multi-crystalline solar cell/module using Moth-flame optimization algorithm. *Energ Convers*, *123*, 535–54.
18. Yamany, W., Fawzy, M., Tharwat, A., & Hassanien, A. E. (2016). Moth-flame optimization for training Multi-Layer Perceptrons. In *2015 11th International Computer Engineering Conference*. https://doi.org/10.1109/ICENCO.2015.7416360.
19. Abd El Azizab, M., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and Moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications*, *83*, 242–256.
20. Zhao, H., Zhao, H., & Guo, S. (2016). Using GM (1,1) Optimized by MFO with rolling mechanism to forecast the electricity consumption of inner mongolia. *Applied Sciences, 6*. https://doi.org/10.3390/app6010020.
21. Yildiz, B. S., & Yildiz, A. R. (2017). Moth-flame optimization algorithm to determine optimal machining parameters in manufacturing processes. *Materials Testing*, *59*, 425–429.
22. Chitsaz, H., & Aminisharifabad, M. (2015). Exact learning of rna energy parameters from structure. *Journal of Computational Biology*, *22*(6), 463–473.

23. Aminisharifabad, M., Yang, Q. & Wu, X. (2018). A penalized Autologistic regression with application for modeling the microstructure of dual-phase high strength steel. *Journal of Quality Technology*. in-press.
24. Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl-Based System*, *89*, 228–249.
25. Reddy, S., Panwar, L. K., Panigrahi, B. K., & Kumar, R. (2018). Solution to unit commitment in power system operation planning using binary coded modified moth flame optimization algorithm (BMMFOA): A flame selection based computational technique. *Journal of Computational Science, 25*, 298–317. Multi-objective MFO.
26. Savsani, V., & Tawhid, M. A. (2017). Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems. *Engineering Applications of Artificial Intelligence*, *63*, 20–32.
27. Nanda, S. J. (2016, September). Multi-objective moth flame optimization. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 2470–2476). IEEE.
28. Jangir, N., Pandya, M. H., Trivedi, I. N., Bhesdadiya, R. H., Jangir, P., & Kumar, A. (2016, March). Moth-Flame Optimization algorithm for solving real challenging constrained engineering optimization problems. In *Electrical, Electronics and Computer Science (SCEECS), 2016 IEEE Students' Conference on* (pp. 1–5). IEEE.
29. Bhesdadiya, R. H., Trivedi, I. N., Jangir, P., & Jangir, N. (2018). Moth-flame optimizer method for solving constrained engineering optimization problems. In *Advances in Computer and Computational Sciences* (pp. 61–68). Springer, Singapore.
30. Apinantanakon, W., & Sunat, K. (2017, July). OMFO: A new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems. In *International Conference on Computing and Information Technology* pp. 22–31). Springer, Cham.
31. Emary, E., & Zawbaa, H. M. (2016). Impact of chaos functions on modern swarm optimizers. *PloS One*, *11*(7), e0158738.
32. Wang, M., Chen, H., Yang, B., Zhao, X., Hu, L., & Cai, Z., et al. (2017). Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. *Neurocomputing, 267*, 69–84.
33. Guvenc, U., Duman, S., & Hnsloglu, Y. (2017, July). Chaotic moth swarm algorithm. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)* (pp. 90–95). IEEE.
34. Li, Z., Zhou, Y., Zhang, S., & Song, J. (2016). Lvy-flight moth-flame algorithm for function optimization and engineering design problems. *Mathematical Problems in Engineering*.
35. Trivedi, I. N., Bhesdadiya, R. H., Pandya, M. H., Jangir, N., Jangir, P., & Ladumor, D. Implementation of meta-heuristic levy flight moth-flame optimizer for solving real challenging constrained engineering optimization problems.
36. Sayed, G. I., & Hassanien, A. E. (2018). A hybrid SA-MFO algorithm for function optimization and engineering design problems. *Complex & Intelligent Systems*, 1–18.
37. Bhesdadiya, R. H., Trivedi, I. N., Jangir, P., Kumar, A., Jangir, N., & Totlani, R. (2017). A novel hybrid approach particle swarm optimizer with moth-flame optimizer algorithm. In *Advances in Computer and Computational Sciences* (pp. 569-577). Springer, Singapore.
38. Anfal, M., & Abdelhafid, H. (2017). Optimal placement of PMUs in algerian network using a hybrid particle SwarmMoth flame optimizer (PSO-MFO). *Electrotehnica, Electronica, Automatica*, *65*(3).
39. Jangir, P. (2017). Optimal power flow using a hybrid particle Swarm optimizer with moth flame optimizer. *Global Journal of Research In Engineering*.
40. Sarma, A., Bhutani, A., & Goel, L. (2017, September). Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality. In *Intelligent Systems Conference (IntelliSys), 2017* (pp. 52–60). IEEE.
41. Zhang, L., Mistry, K., Neoh, S. C., & Lim, C. P. (2016). Intelligent facial emotion recognition using moth-firefly optimization. *Knowledge-Based Systems*, *111*, 248–267.

42. Li, C., Li, S., & Liu, Y. (2016). A least squares support vector machine model optimized by moth-flame optimization algorithm for annual power load forecasting. *Applied Intelligence*, *45*(4), 1166–1178.
43. Yamany, W., Fawzy, M., Tharwat, A., & Hassanien, A. E. (2015, December). Moth-flame optimization for training multi-layer perceptrons. In *Computer Engineering Conference (ICENCO), 2015 11th International* (pp. 267–272). IEEE.
44. Faris, H., Aljarah, I., & Mirjalili, S. (2017). Evolving radial basis function networks using MothFlame optimizer. In *Handbook of Neural Computation* (pp. 537–550).
45. Dosdoru, A. T., Boru, A., Gken, M., zalc, M., & Gken, T. (2018). Assessment of hybrid artificial neural networks and Metaheuristics for stock market forecasting. *ukurova niversitesi Sosyal Bilimler Enstits Dergisi, 27*(1), 63–78.
46. Kaur, N., Rattan, M., & Gill, S. S. (2018). Performance optimization of Broadwell-Y shaped transistor using artificial neural network and Moth-flame optimization technique. *Majlesi Journal of Electrical Engineering*, *12*(1), 61–69.
47. Sayed, G. I., Soliman, M., & Hassanien, A. E. (2016). Bio-inspired swarm techniques for thermogram breast cancer detection. In *Medical Imaging in Clinical Applications* (pp. 487–506). Springer, Cham.
48. Sayed, G. I., & Hassanien, A. E. (2017). Moth-flame swarm optimization with neutrosophic sets for automatic mitosis detection in breast cancer histology images. *Applied Intelligence*, *47*(2), 397–408.
49. Diab, A. A. Z., & Rezk, H. Optimal sizing and placement of capacitors in radial distribution systems based on Grey Wolf, Dragonfly and MothFlame optimization algorithms. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 1–20.
50. Mohanty, B. (2018). Performance analysis of moth flame optimization algorithm for AGC system. *International Journal of Modelling and Simulation*, 1–15.
51. Mohanty, B., Acharyulu, B. V. S., & Hota, P. K. (2018). Mothflame optimization algorithm optimized dualmode controller for multiarea hybrid sources AGC system. *Optimal Control Applications and Methods*, *39*(2), 720–734.
52. Barisal, A. K., & Lal, D. K. (2018). Application of moth flame optimization algorithm for AGC of multi-area interconnected power systems. *International Journal of Energy Optimization and Engineering (IJEOE)*, *7*(1), 22–49.
53. Lal, D. K., Bhoi, K. K., & Barisal, A. K. (2016, October). Performance evaluation of MFO algorithm for AGC of a multi area power system. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)* (pp. 903–908). IEEE.
54. Reddy, M. P. K., & Babu, M. R. (2017). A hybrid cluster head selection model for internet of things. *Cluster Computing*, 1–13.
55. Yang, X., Luo, Q., Zhang, J., Wu, X., & Zhou, Y. (2017, August). Moth Swarm algorithm for clustering analysis. In *International Conference on Intelligent Computing* (pp. 503–514). Springer, Cham.
56. Metwally, A. S., Hosam, E., Hassan, M. M., & Rashad, S. M. (2016, October). WAP: A novel automatic test generation technique based on moth flame optimization. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 59–64). IEEE.
57. Sharma, R., & Saha, A. (2017). Optimal test sequence generation in state based testing using moth flame optimization algorithm. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1–13.
58. Bhadoria, A., Kamboj, V. K., Sharma, M., & Bath, S. K. (2018). A solution to non-convex/convex and dynamic economic load dispatch problem using moth flame optimizer. *INAE Letters*, *3*(2), 65–86.
59. Trivedi, I. N., Kumar, A., Ranpariya, A. H., & Jangir, P. (2016, April). Economic load dispatch problem with ramp rate limits and prohibited operating zones solve using Levy Flight Moth-Flame optimizer. In *2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS)* (pp. 442–447). IEEE.
60. Huang, Y., Ji, Z., Chen, Q., & Niu, S. (2017, September). Geographic atrophy segmentation for SD-OCT images by MFO algorithm and affinity diffusion. In *International Conference on Intelligent Science and Big Data Engineering* (pp. 473–484). Springer, Cham.

61. Mei, R. N. S., Sulaiman, M. H., Daniyal, H., & Mustaffa, Z. (2018). Application of Moth-flame optimizer and ant lion optimizer to solve optimal reactive power dispatch problems. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10*(1–2), 105–110.
62. Elsakaan, A. A., El-Sehiemy, R. A. A., Kaddah, S. S., & Elsaid, M. I. (2018). Economic power dispatch with emission constraint and valve point loading effect using moth flame optimization algorithm. In *Advanced Engineering Forum* (Vol. 28, pp. 139–149). Trans Tech Publications.
63. Trivedi, I. N., Parmar, S. A., Pandya, M. H., Jangir, P., Ladumor, D., & Bhoye, M. T. Optimal active and reactive power dispatch problem solution using Moth-Flame optimizer.
64. Anbarasan, P., & Jayabarathi, T. (2017). Optimal reactive power dispatch using Moth-flame optimization algorithm. *International Journal of Applied Engineering Research*, *12*(13), 3690–3701.
65. Sulaiman, M. H., Mustaffa, Z., Aliman, O., Daniyal, H., & Mohamed, M. R. (2016). Application of moth-flame optimization algorithm for solving optimal reactive power dispatch problem.
66. Upper, N., Hemeida, A. M., & Ibrahim, A. A. (2017, December). Moth-flame algorithm and loss sensitivity factor for optimal allocation of shunt capacitor banks in radial distribution systems. In *Power Systems Conference (MEPCON), 2017 Nineteenth International Middle East* (pp. 851–856). IEEE.
67. Dhyani, A., Panda, M. K., & Jha, B. (2018). Moth-flame optimization-based fuzzy-PID controller for optimal control of active magnetic bearing system. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 1–13.
68. Saurav, S., Gupta, V. K., & Mishra, S. K. (2017, March). Moth-flame optimization based algorithm for FACTS devices allocation in a power system. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1–7). IEEE.
69. Tolba, M. A., Diab, A. A. Z., Tulsky, V. N., & Abdelaziz, A. Y. (2018). LVCI approach for optimal allocation of distributed generations and capacitor banks in distribution grids based on mothflame optimization algorithm. *Electrical Engineering*, 1–26.
70. Gope, S., Dawn, S., Goswami, A. K., & Tiwari, P. K. (2016, November). Profit maximization with integration of wind farm in contingency constraint deregulated power market using Moth flame optimization algorithm. In *Region 10 Conference (TENCON), 2016 IEEE* (pp. 1462–1466). IEEE.
71. Ebrahim, M. A., Becherif, M., & Abdelaziz, A. Y. (2018). Dynamic performance enhancement for wind energy conversion system using Moth-flame optimization based blade pitch controller. *Sustainable Energy Technologies and Assessments*, *27*, 206–212.
72. GhobaeiArani, M., Rahmanian, A. A., Souri, A., & Rahmani, A. M. A mothflame optimization algorithm for web service composition in cloud computing: Simulation and verification. Software: Practice and Experience.
73. Khairuzzaman, A. K. M., & Chaudhury, S. (2017). Moth-flame optimization algorithm based multilevel thresholding for image segmentation. *International Journal of Applied Metaheuristic Computing (IJAMC)*, *8*(4), 58–83.
74. Said, S., Mostafa, A., Houssein, E. H., Hassanien, A. E., & Hefny, H. (2017, September). Moth-flame optimization based segmentation for MRI liver images. In *International Conference on Advanced Intelligent Systems and Informatics* (pp. 320–330). Springer, Cham.
75. Muangkote, N., Sunat, K., & Chiewchanwattana, S. (2016, July). Multilevel thresholding for satellite image segmentation with moth-flame based optimization. In *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)* (pp. 1–6). IEEE.
76. El Aziz, M. A., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and Moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications*, *83*, 242–256.
77. Li, W. K., Wang, W. L., & Li, L. (2018). Optimization of water resources utilization by multi-objective moth-flame algorithm. *Water Resources Management*, 1–14.

78. Das, A., Mandal, D., Ghoshal, S. P., & Kar, R. (2018). Concentric circular antenna array synthesis for side lobe suppression using moth flame optimization. *AEU-International Journal of Electronics and Communications*, *86*, 177–184.
79. Huang, L. N., Yang, B., Zhang, X. S., Yin, L. F., Yu, T., & Fang, Z. H. (2017). Optimal power tracking of doubly fed induction generator-based wind turbine using swarm mothflame optimizer. *Transactions of the Institute of Measurement and Control*, 0142331217712091.
80. Pathak, V. K., & Singh, A. K. (2017). Accuracy control of contactless laser sensor system using whale optimization algorithm and moth-flame optimization. *tm-Technisches Messen, 84*(11), 734–746.
81. Das, A., & Srivastava, L. Optimal placement and sizing of distributed generation units for power loss reduction using Moth-flame optimization algorithm.
82. Zou, L., Ge, B., & Chen, L. (2018). Range image registration based on hash map and moth-flame optimization. *Journal of Electronic Imaging*, *27*(2), 023015.
83. Sahu, P. C., Prusty, R. C., & Panda, S. (2017, April). MFO algorithm based fuzzy-PID controller in automatic generation control of multi-area system. In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)* (pp. 1–6). IEEE.
84. Sayed, G. I., Hassanien, A. E., Nassef, T. M., & Pan, J. S. (2016, November). Alzheimers disease diagnosis based on Moth flame optimization. In *International Conference on Genetic and Evolutionary Computing* (pp. 298–305). Springer, Cham.
85. Gope, S., Dawn, S., Goswami, A. K., & Tiwari, P. K. (2016, November). Moth Flame optimization based optimal bidding strategy under transmission congestion in deregulated power market. In *Region 10 Conference (TENCON), 2016 IEEE* (pp. 617–621). IEEE.
86. Chauhan, S. S., & Kotecha, P. (2016, November). Single level production planning in petrochemical industries using Moth-flame optimization. In *Region 10 Conference (TENCON), 2016 IEEE* (pp. 263–266). IEEE.
87. Soliman, G. M., Khorshid, M. M., & Abou-El-Enien, T. H. (2016). Modified moth-flame optimization algorithms for terrorism prediction. *International Journal of Application or Innovation in Engineering and Management*, *5*, 47–58.
88. Singh, P., & Prakash, S. (2017). Optical network unit placement in Fiber-Wireless (FiWi) access network by Moth-Flame optimization algorithm. *Optical Fiber Technology*, *36*, 403–411.
89. Mekhamer, S. F., Abdelaziz, A. Y., Badr, M. A. L., & Algabalawy, M. A. (2015). Optimal multi-criteria design of hybrid power generation systems: A new contribution. *International Journal of Computer Applications*, *129*(2), 13–24.
90. Ewees, A. A., Sahlol, A. T., & Amasha, M. A. (2017, May). A Bio-inspired moth-flame optimization algorithm for Arabic handwritten letter recognition. In *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)* (pp. 154–159). IEEE.
91. Zhao, H., Zhao, H., & Guo, S. (2016). Using GM (1, 1) optimized by MFO with rolling mechanism to forecast the electricity consumption of inner mongolia. *Applied Sciences*, *6*(1), 20.
92. Zawbaa, H. M., Emary, E., Parv, B., & Sharawi, M. (2016, July). Feature selection approach based on moth-flame optimization algorithm. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 4612–4617). IEEE.
93. Patil, D., Mulla, A., Chakraborty, D., & Pillai, H. (2015). Computation of feedback control for time optimal state transfer using Groebner basis. *Systems & Control Letters*, *79*, 1–7.
94. Jabbari Asl, H., & Yoon, J. (2016). Power capture optimization of variable-speed wind turbines using an output feedback controller. *Renewable Energy*, *86*, 517–525.
95. Zhou, H., Chen, C., Lai, J., Lu, X., Deng, Q., Gao, X., et al. (2018). Affine nonlinear control for an ultra-supercritical coal fired once-through boiler-turbine unit. *Energy*, *153*, 638–649.
96. Tavakoli, M., Taghirad, H. D., & Abrishamchian, M. (2005). Identification and robust H control of the rotational/translational actuator system. *International Journal of Control, Automation*, *3*, 387–396.

97. Gao, B., & Ye, F. (2014). Dynamical analysis and stabilizing control of inclined rotational translational actuator systems. *Journal of Applied Mathematics*,. https://doi.org/10.1155/2014/598384.
98. Kumar, A., & Sharma, R. (2017). Fuzzy lyapunov reinforcement learning for non linear systems. *ISA Transactions*, *67*, 151–159.
99. Bupp, R. T., Bernstein, D. S., & Coppola, V. T. (1998). A benchmark problem for nonlinear control design. *International Journal Robust Nonlinear Control*, *8*, 307–310.
100. Luo, B, Wu, H. N., Huang, T, & Derong Liu, D. Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design. *Automatica, 50*, 3281–3290.
101. Cimen, T., & Banks, S. P. (2004). Global optimal feedback control for general nonlinear systems with nonquadratic performance criteria. *Systems Control Letters*, *53*, 327–346.

Particle Swarm Optimization: Theory, Literature Review, and Application in Airfoil Design

Seyedali Mirjalili, Jin Song Dong, Andrew Lewis and Ali Safa Sadiq

Abstract The Particle Swarm Optimization (PSO) is one of the most well-regarded algorithms in the literature of meta-heuristics. This algorithm mimics the navigation and foraging behaviour of birds in nature. Despite the simple mathematical model, it has been widely used in diverse fields of studies to solve optimization problems. There is a tremendous number of theoretical works on this algorithm too that has led to a large number of variants, improvements, and hybrids. This chapter covers the inspirations, mathematical equations, and the main algorithm of this technique. Its performance is tested and analyzed on a challenging real-world problem in the field of aerospace engineering.

1 Introduction

Meta-heuristics are high-level techniques using heuristics to solve a wide range of problems. As opposed to heuristics, they do not need any problem-dependent heuristic information. One of the main advantages of such techniques is the fact that they make few assumptions about the problem and attempt to consider them as a black box [1]. To search for optimal solutions, they sample the search space of the problem that is too large to be searched entirely.

S. Mirjalili (✉) · J. Song Dong · A. Lewis
Institute for Integrated and Intelligent Systems, Griffith University,
Nathan, Brisbane, QLD 4111, Australia
e-mail: seyedali.mirjalili@griffithuni.edu.au

J. Song Dong
Department of Computer Science, School of Computing, National University of Singapore,
Singapore, Singapore
e-mail: j.dong@griffith.edu.au

A. Lewis
e-mail: a.lewis@griffith.edu.au

A. S. Sadiq
School of Information Technology, Monash University, 47500 Bandar Sunway, Malaysia
e-mail: ali.safaa@monash.edu

S. Mirjalili et al. (eds.), *Nature-Inspired Optimizers*, Studies in Computational Intelligence 811, https://doi.org/10.1007/978-3-030-12127-3_10

Meta-heuristics belong to the family of Soft Computing techniques, in which solutions can be partially true or inaccurate. This comes at the cost of unreliability in finding the best solution for a given problem. The extremely large size of some problems and failure of exact methods justify the need for such problem solving techniques since they find optimal or near optimal solutions in a reasonable time.

The field of meta-heuristics has seen a large number of algorithms mostly inspired from nature. Some of the most popular ones are Simulated Annealing (SA) [2] mimicking the annealing process in physics, Genetic Algorithms (GA) [3] inspired from the theory of evolution, Differential Evolution (DE) [4] incorporating the concepts of evolution in differential equations, Ant Colony Optimization (ACO) [5] inspired from the swarm intelligence of ants, and Particle Swarm Optimization (PSO) [6] that mimics navigation of birds.

Some of the algorithms belong to the family of swarm intelligence techniques. In this area, the main inspiration is the collective behaviour of insects, animals, or any other creatures that leads to fascinating global behaviours. There is no centralized control unit for decision making and the local interactions between individual/individual and individuals/environment lead to a global decision making.

The PSO algorithm is one of the most well-regarded swarm intelligence techniques that has been widely used in both science and industry [7, 8]. The main purpose of this chapter is to present the preliminaries, essential definitions, mathematical models, and algorithms of this technique.

2 Particle Swarm Optimization

The PSO algorithm has been inspired from the flocking behavior of birds in nature. In this algorithm, each particle is considered to be a solution for a given optimization problem. It is made of two vectors: position and velocity. The position vector includes the values for each of the variables in the problem. If the problem has two parameters, for instance, the particles will have position vectors with two dimensions. Each particle will then be able to move in an n-dimensional search space where n is the number of variables. To update the position of particles, the second vector (velocity) is considered. This vector defines the magnitude and direction of step size for each dimension and each particle independently.

The location of particles is updated in each step of optimization using the following equation:

$$\overrightarrow{X_i(t+1)} = \overrightarrow{X_i(t)} + \overrightarrow{V_i(t+1)} \tag{1}$$

where $\overrightarrow{X_i(t)}$ shows the position of ith particle at tth iteration and $V_i(t)$ shows the velocity of ith particle at tth iteration.

This equation shows that the position updating is simple and the main component is the velocity vector. The velocity vector is defined as follows:

$$\overrightarrow{V_i(t+1)} = w\overrightarrow{V_i(t)} + c_1 r_1\left(\overrightarrow{P_i(t)} - \overrightarrow{x_i(t)}\right) + c_2 r_2\left(\overrightarrow{G(t)} - \overrightarrow{x_i(t)}\right) \quad (2)$$

where $\overrightarrow{X_i(t)}$ shows the position of ith particle at tth iteration, $\overrightarrow{V_i(t)}$ shows the velocity of ith particle at tth iteration, w is the inertial weight, c_1 shows the individual coefficient, c_2 signifies the social coefficient, r_1, r_2 are random numbers in [0, 1], $\overrightarrow{P_i(t)}$ is the best solution obtained by the ith particle until tth iteration, and $\overrightarrow{G(t)}$ shows the best solution found by all particles (entire swarm) until tth iteration.

Equation 2 shows that the velocity vector is made of three components. The first component, $w\overrightarrow{V_i(t)}$ maintains the tendency towards the current velocity. This component is multiplied by an inertial parameter (w). The larger the value of this parameter, the higher the tendency to maintain the previous velocity. The second component, $c_1 r_1\left(\overrightarrow{P_i(t)} - \overrightarrow{x_i(t)}\right)$ simulates the individual intelligence of a bird by memorizing and using the best solution obtained so far by each of the particles. The vector $\overrightarrow{P_i(t)}$ is updated in each iteration in case the ith particle finds a better solution. The impact of this component on the final value of the velocity can be increased or decreased by changing c_1. This parameter is multiplied by a random number in [0, 1] to provide randomized behaviors since PSO is a stochastic optimization technique. Overall, the second component maintains a tendency towards the best solution that a particle found so far, the so called "personal best".

The third component, $c_2 r_2\left(\overrightarrow{G(t)} - \overrightarrow{x_i(t)}\right)$ mimics the social intelligence of a flock of birds, in which the best solution obtained by all particles is saved in $\overrightarrow{G(t)}$ and used in this component. This means that considering the best solution found by the swarm gravitates all particles toward one point. The impact of this component can be tuned using c_2 as well.

With these three components, the next position of a particle can be defined. An example is shown in Fig. 1. This figure shows that each particle considers the previous

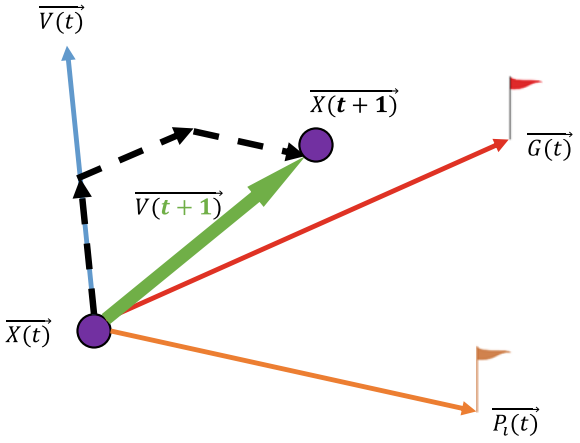

Fig. 1 In PSO, each particle considers the previous velocity, the personal best, and the global best to define the current velocity and update its position

velocity, the personal best, and the global best to define the current velocity and update its position. The next position greatly depends on the random numbers generated using r_1 and r_2 as well. In Fig. 1, it is assumed that the particle considers 50% of the current velocity, 40% of the global best, and 40% of the personal best. In reality, however, c_1 and c_2 are multiplied by random numbers.

To see the area that a particle can move into, the values of w, c_1, and c_2 should be known. In the most well-regarded version of PSO, w linearly decreases from 0.9 to 0.4 proportional to the number of iterations. The parameters c_1, and c_2 are both set to 2 as well. Since it is difficult to show the possible locations of particles while varying the inertial weight, the snapshots of the possible locations when $w = 1$, $w = 0.5$, and $w = 0$ are shown in this subsection. The possible new positions of the particle with this configuration are visualized in Fig. 2.

The vectors toward the personal best and global best can be from 0 to double the size of their distance. This is due to the multiplication of both c_1 and c_2 by a random number in [0, 1]. Therefore, the range of these two parameters can be any number in the interval of [0, 2]. When the value is equal to 0, the particle does not consider the component. When the value is equal to 2, it considers twice the component. This is

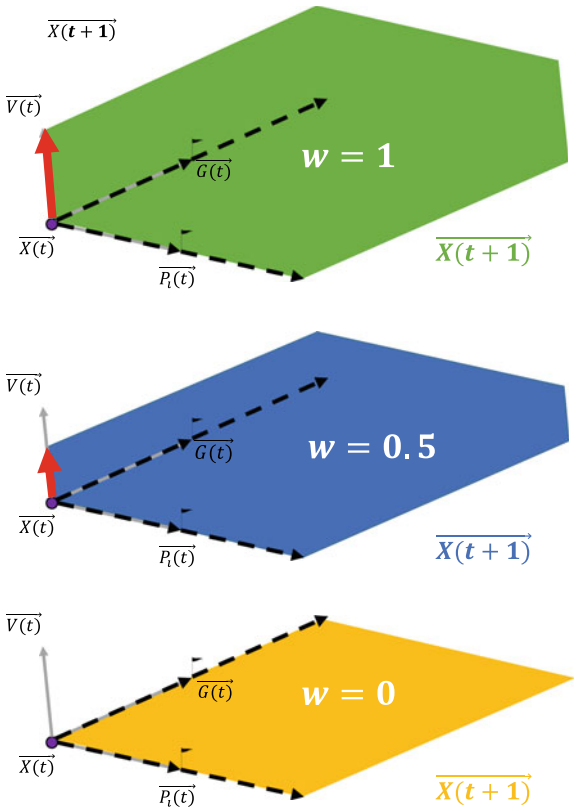

Fig. 2 The impact of the inertial weight (w) on the velocity vector in PSO

why the vectors are doubled in Fig. 2. The lower bound and upper bound of c_1 and c_2 do not change, so the maximum and minimum distance towards personal best and global best are equal as shown in Fig. 2. The inertial weight, however, is changed and its impact is evident in this figure.

Figure 2 shows that when the inertial weight is equal to 0, the next position of the particle is between its current location and the locations of personal and global bests. This causes exploitation and local search in the PSO model since the algorithm searches locally inside the area defined by the current position, personal best, and global best. In fact, the local search is purely done when c_1 and c_2 are both set to 1. An example is given in Fig. 3.

The exploration and global search increase proportionally to the values of all these parameters. This is shown in Fig. 4. It can be seen that the particle tends to go beyond the area between itself and the global or local bests. This leads to exploration and global search. It should be noted that the personal and global bests are updated constantly as well. So, the shaded area in Fig. 3, which shows the next position area, keeps changing as well.

The PSO algorithm uses these simple concepts to look for the global optimum of a given optimization problem. It starts with a random population of solutions. It then iteratively goes through the following steps until the end condition is met:

1. Calculating the objective value of all particles

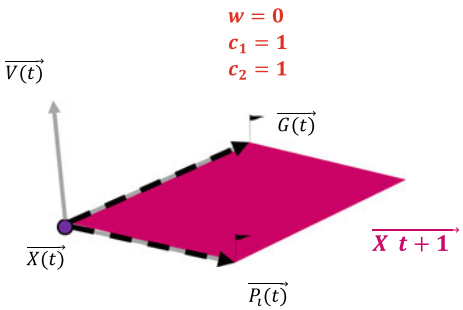

Fig. 3 When $w = 0$, $c_1 = 1$ and $c_2 = 1$, the exploitation and local search is at the maximum level in PSO

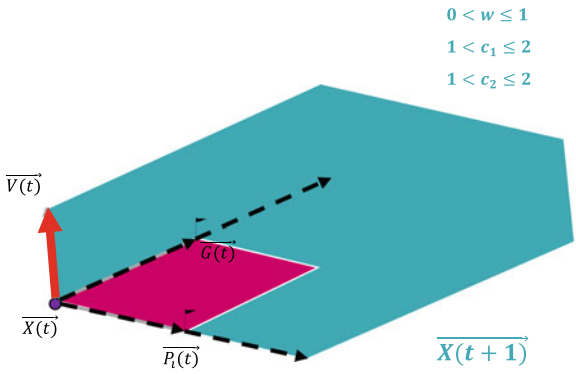

Fig. 4 The exploration and global search increases proportional to the values w, c_1 and c_2

2. Updating w, c_1 and c_2. If c_1 and c_2 are constant, then only w is updated
3. Updating the personal best and the global best
4. Calculating the velocity vector for each particle using Eq. 1
5. Calculate the next position for each particle using Eq. 2

At the end of the optimization process, the best solution obtained by the entire swam will be returned as the final optimum for the optimization problem.

The PSO has been modified in a large number of works in the literature. There are also different variants of this algorithm too.

To solve binary problems, transfer functions have been widely integrated into PSO [9]. Such functions use the velocity values to flip a binary bit. The larger value of the velocity vector, the higher probability of flipping the binary bit [10, 11].

To solve multi-objective problems, several multi-objective variants of PSO have been proposed in the literature. The most popular technique was proposed by Coello et al. [12], in which an archive was employed to store and improve non-dominated solutions (Pareto optimal solutions) during the optimization process. In each iteration, one solution from the archive was chosen as the global best since all the solutions in the archive are considered a solution for a multi-objective optimization problem. To improve diversity of the solutions obtained, the archive was equipped with a grid mechanism [13]. There are also methods such as crowding distance to improve diversity as well [14]. Another popular technique is the use of non-dominated sorting in the PSO to rank and improve non-dominated solutions obtained so far [15]. There are also algorithms that aggregate objectives in the literature [16].

To handle constraints, there are many functions in the literature [17, 18]. Such functions are mostly integrated into the objective function to penalize the particles that violate the constraints. Some of these functions penalize the particles in a similar manner regardless of the level of violation. In this case, such functions can be called barrier functions. There are also functions that penalize particles proportional to their level of violation. Such functions are essential when solving problems with dominated infeasible regions since PSO deals with a large number of infeasible solutions in each iteration [19].

To solve problems with dynamically changing objectives, a dynamic version of PSO has been proposed in [20]. The main idea is to prevent PSO from converging towards a solution to track the changes in the objective function(s) [21]. To solve problems with uncertainties, robust PSOs [22, 23] and robust MOPSOs [24, 25] have been proposed in the literature as well.

3 Results

3.1 Exploration and Exploitation in PSO

In this subsection, the statements about the impact of the main controlling parameters (w, c_1 and c_2) on the performance of PSO have been investigated experimentally.

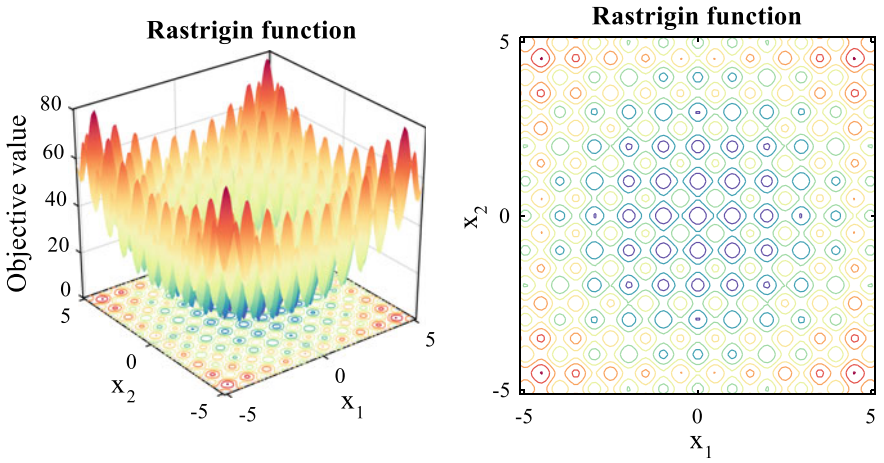

Fig. 5 Rastrigin test function used in the experiment

We test PSO while changing the parameters in a case study. Data visualization in a space with more than three dimensions is difficult to perform and analyze. Therefore, this section solves a 2-dimensional version of the Rastrigin test function as shown in Fig. 5.

The mathematical formulation of this test function is as follows:

$$f(\overrightarrow{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - Acos(2\pi x_i)\right] \tag{3}$$

where $A = 10$ and $x_i \in [-5.12, 5.12]$. The global optimum of this test function is right at the origin with the optimal value of 0.

Figure 5 shows that this function has a large number of local solutions and the search space is very challenging. To observe the exploratory and exploitative behaviour of PSO, the history of sampled points is visualized. If the solutions are widely spread around the search space, it is a good indication of high exploration. If the solutions scatter around just a region of the search space, it shows that the exploitation is high.

In the following sub-section, PSO is run with four particles and 500 iterations while changing the controlling parameters independently. To observe the exploratory and exploitative behaviours of PSO, the search history, convergence curve (GBEST in each iteration), average objective of all particles, fluctuations in the first dimension, and average distance between all particles are presented in multiple figures. Note that in the search history, the sampled points change their colours from cold to warm proportional to the number of iterations.

3.1.1 Investigating the Impact of the Inertial Weight (w)

In the first experiment, the value of the inertial weight is set to 1, 0.5, and 0. It also adaptively decreases from 0.9 to 0.2 proportional to the number of iterations.

The results in Fig. 6 show that when the inertial weight is equal to 0, the exploration is at the lowest level. This can be seen in the search history (the first column), average objective (the third column), fluctuation (the fourth column), and average distance

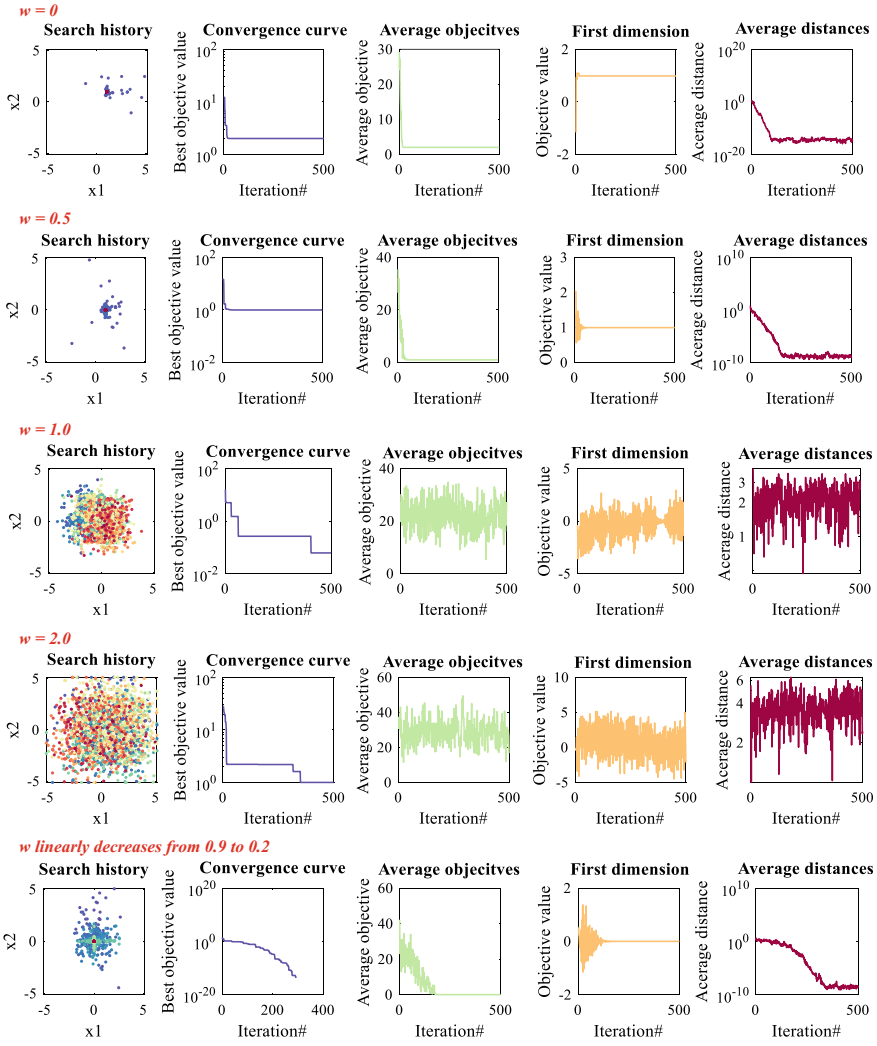

Fig. 6 The performance of PSO when the inertial weight is set to 1, 0.5, and 0. It also adaptively decreases from 0.9 to 0.2 proportionally to the number of iterations

between the particles (the fifth column). In addition, the exploitation is very high when using this value for the inertial weight. This configuration can be recommended for linear problems.

Figure 6 shows that exploration increases proportional to the value of the inertia weight. Considering the third and the fourth rows in this figure, it can be seen the search history covers wide areas. Since there are sampled points with warm color, it shows that particles keep exploring even in the final iteration. The fluctuations are substantial in the average objective, first dimension and the average distances too. These are all evidence of very high exploration when $w \geq 1$.

Neither of the above behaviours is enough to solve challenging problems. Both exploration and exploitation should be balanced. This can be done by adaptively changing the inertia weight. The last row in Fig. 6 shows that the algorithm first explores the search space (as the sampled points with cold colour show) and then exploits it. This allows finding an accurate estimation of the global optimum around iteration 300 as the convergence shows. The fluctuations in the average objectives, first dimension, and average distances are also reasonable, in that the changes taper away as the iteration count increases.

3.1.2 Investigating the Impact of the Cognitive Component (c_1)

In this subsection, different values (0, 1, 2, 3, and 4) are used for the cognitive constant in the PSO algorithm. The results are provided in Fig. 7. This figure shows that exploitation is very high when the cognitive constant is equal to 0. This is because all particles are pulled towards the global optimum since the impacts of the personal bests are skipped with this value. The average distance between particles is the most interesting curve to check here. The average distance between all particles is gradually decreasing, showing consistent movement of the particles towards each other and the global best.

On the other hand, the exploration increases proportional to the value of the cognitive constant as Fig. 7 shows. The reason why the algorithm finds the global optimum in most of the cases is due to the use of adaptive inertial weight in this experiment and the simplicity of the test function. For more challenging problems, it will be more difficult for PSO to find the global optimum. The exploration is very high when $c_1 = 3$ or $c_1 = 4$. In fact, the convergence and exploitation are interrupted when using $c_1 = 4$. This is because the particles almost exclusively gravitate toward their personal bests.

Figure 7 indicates that exploration and exploitation balances best at $c_2 = 2$, which is the reason why most works in the literature uses this value for the cognitive constant.

3.1.3 Investigating the Impact of the Social Component (c_2)

In this subsection, different values (0, 1, 2, 3, and 4) are used for the social constant in the PSO algorithm. The results are provided in Fig. 8.

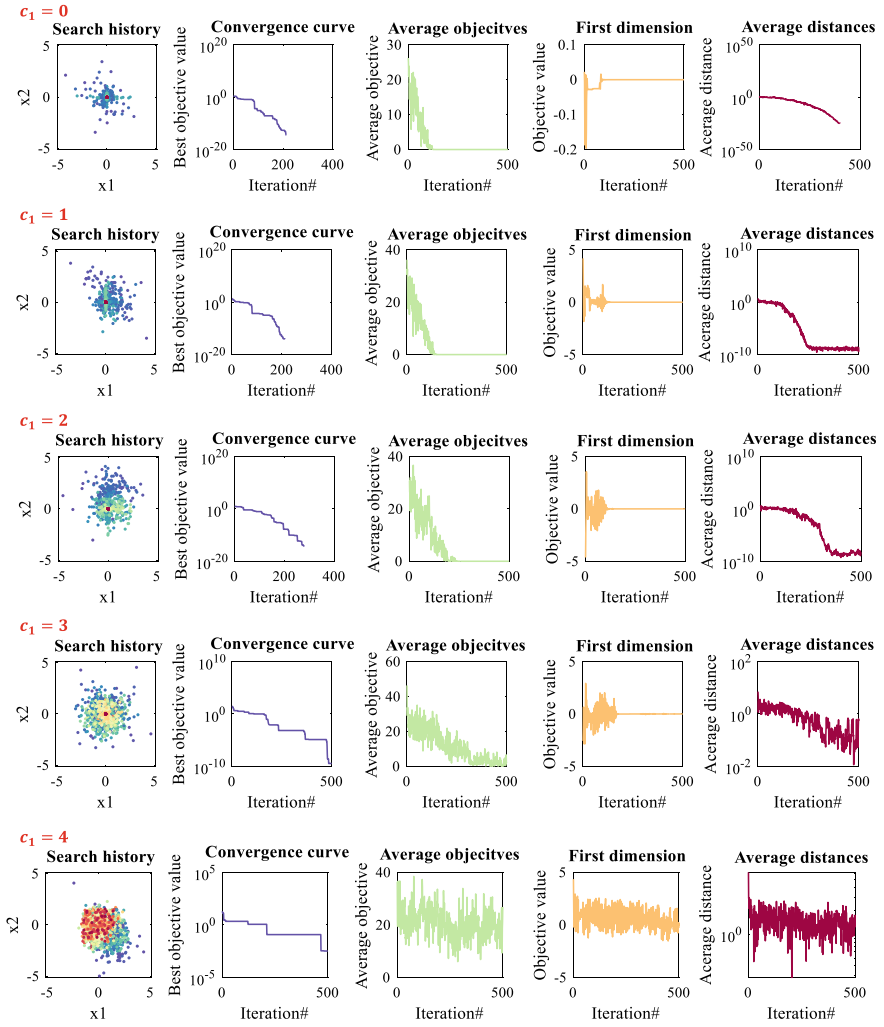

Fig. 7 The performance of PSO when the cognitive constant is set to 0, 1, 2, 3, and 4

An interesting pattern can be observed when $c_2 = 0$, in which four particles converged towards independent points. This is due to the fact that the global best is not considered when $c_2 = 0$, so each particle searches around its personal best independently. In this case, there is no systematic convergence towards one solution at the end.

As the social constant increases, particles start to use the global best, and the entire swarm is more directed. A reasonable balance between local and global search is observed when using $c_2 = 2$. For larger values, the impact from the global best dominates, and the PSO algorithm behaves more randomly.

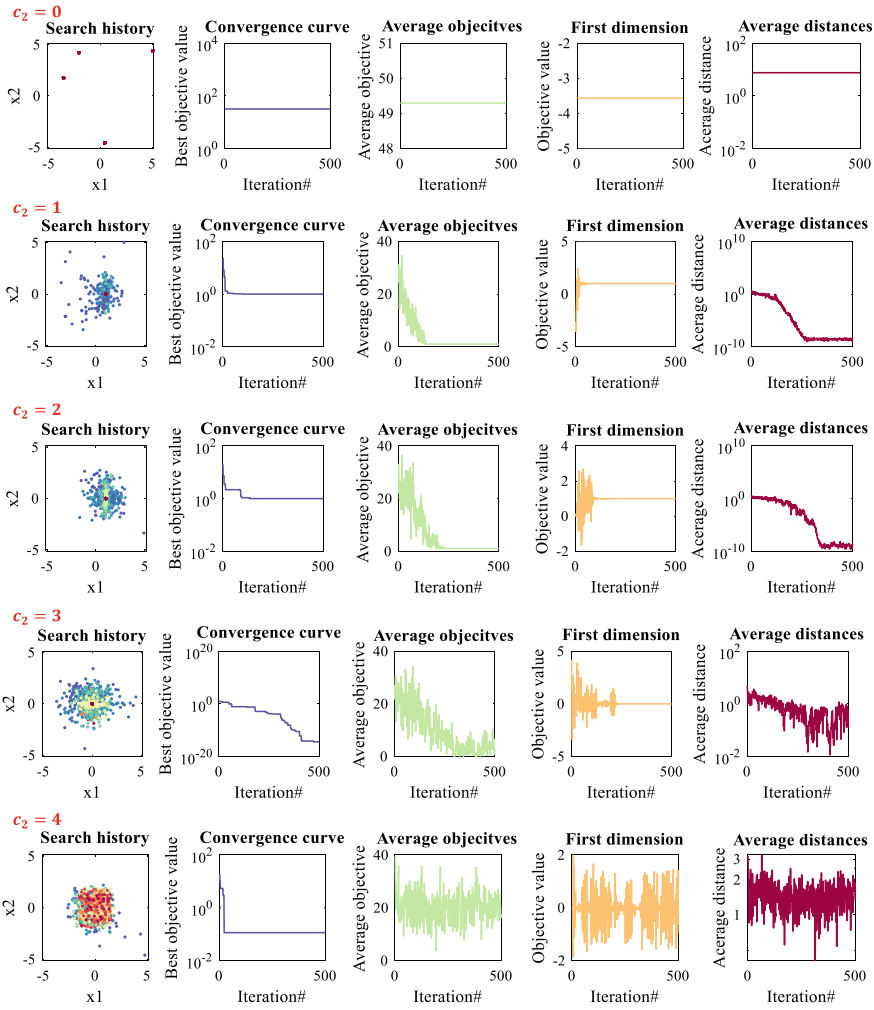

Fig. 8 The performance of PSO when the social constant is set to 0, 1, 2, 3, and 4

Overall, the results so far have shown that the best configuration is to linearly decrease the inertia weight while setting c_1 and c_2 to 2. This configuration will be used when solving the real-world problem in the next section.

3.2 2D Airfoil Design Using PSO

The problem of 2D airfoil design deals with designing the cross section of an aircraft wing considering two objectives: maximizing lift and minimizing drag. Since the main focus of this book and chapter is on single-objective optimization, the problem is solved considering the latter objective.

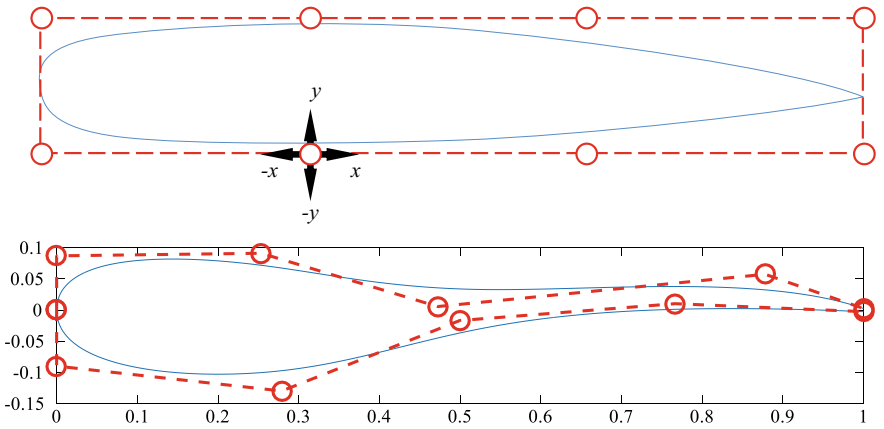

Fig. 9 B-spline method with controlling points to define the shape of an airfoil

Several variants of this problem have been solved in the literature. They mostly use different methods and parameters to define the shape of the airfoil. This subsection uses the B-spline method as shown in Fig. 9. This figure shows that there are eight controlling parameters that can be moved in two dimensions. As such, the total number of variables is equal to 16.

The problem of 2D airfoil design is formulated as follows:

$$\text{Maximize: } C_l\left(\overrightarrow{X}, \overrightarrow{Y}\right) \tag{4}$$

$$\text{Minimize: } C_d\left(\overrightarrow{X}, \overrightarrow{Y}\right) \tag{5}$$

$$\text{Subject to: } -1 \leq \overrightarrow{X}, \overrightarrow{Y} \leq 1 \tag{6}$$

where C_d is the coefficient of drag, $\overrightarrow{X} = \{x_1, x_2, ..., x_8\}$ and $\overrightarrow{Y} = \{y_1, y_2, ..., y_8\}$

There are a large number of constraints in this problem. Since the details of the problem is outside the scope of this book, interested readers may refer to [26]. To calculate the coefficient of drag a free-ware called XFoil is employed [27].

The above problem is multi-objective, but the experiments described in this chapter optimizes each of the objectives independently. In addition, $\frac{C_l}{C_d}$ is maximized, which should have a tendency to maximize the lift and minimize the drag. All three versions of this problem are solved by the PSO algorithm using 10 particles and 150 iterations. PSO is required to stop in case of exceeding the maximum number of iterations or not getting a better result in more than 100 iterations. The results are shown in Fig. 10. This figure shows that the objective function (C_l/C_d) is increased

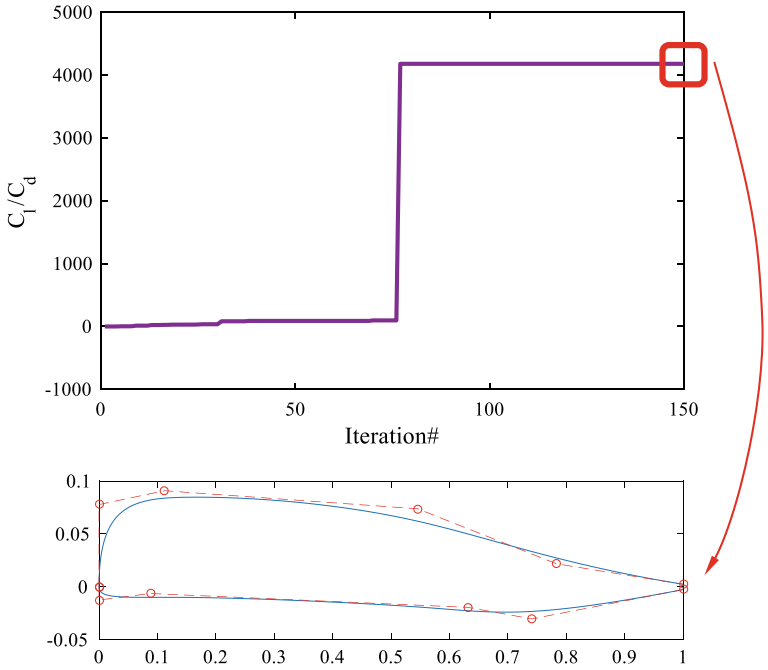

Fig. 10 The convergence curve and final optimal airfoil when maximizing C_l/C_d

significantly only once. This shows that PSO is not better than a random search in this problem, which is due to the large number of infeasible particles in each iteration that hinder the search mechanism of PSO. The convergence curve shows the improvement in the objective value, yet it does not show the shapes of airfoils. To see how the random population of PSO looks like, Fig. 11 is given. This figure shows 10 particles in the initial population.

It can be seen in this figure that almost all of the initial shapes are not smooth and do not provide lift. Most of them even violate constraints, so they are considered infeasible. Two popular methods to alleviate these drawbacks are limiting the range of parameters and starting with initial feasible population. This is done at the end of this chapter.

This chapter also optimizes both of the objectives independently. The same experimental settings compared to the previous experiments are used to find an optimal design for the 2D airfoil. The convergence of PSO and the best solution found are shown in Figs. 12 and 13. The convergence of the PSO algorithm when maximizing C_l is similar to that shown in Fig. 10 when maximizing C_l/C_d. There is a big incline at a point during the iterations and there is no significant improvement afterwards. In fact, there is no improvement for 100 iterations, which is the reason why the

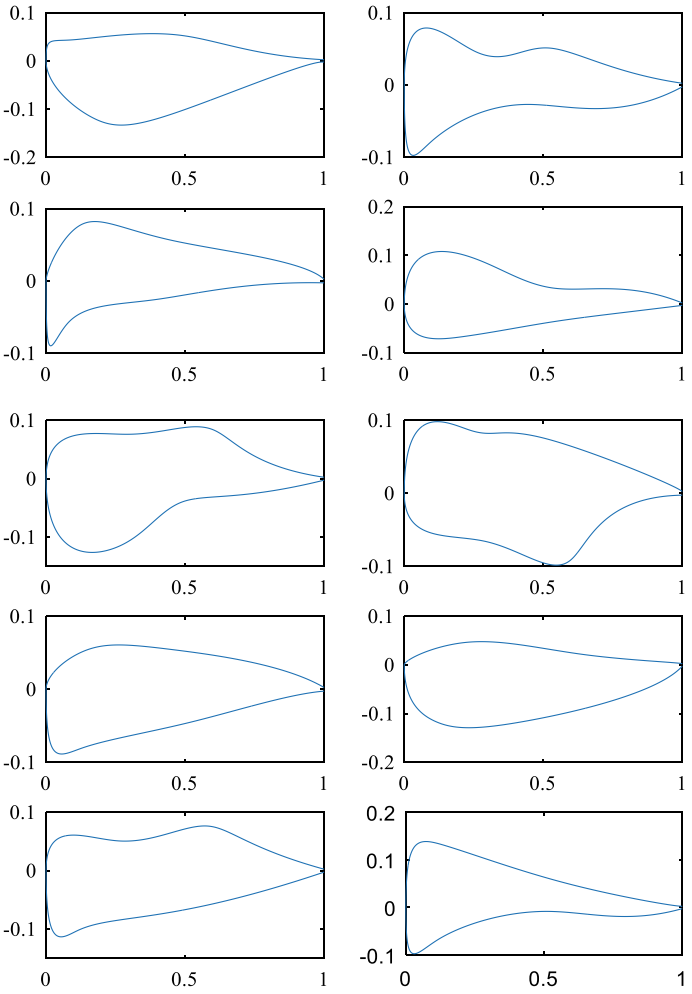

Fig. 11 Ten initial airfoils repressed by each particle in the first population of PSO

algorithm stopped before hitting the 150th iteration. This is due to the same reason mentioned when maximizing C_l/C_d.

The convergence of the PSO algorithm when minimizing C_d is different from those on other objectives. Figure 13 shows that the best solution keeps improving over the course of iteration. Since the algorithm has not been changed, this shows that the search space of the drag objective function is less challenging than that of the lift. Any airoilf shape can give drag, but not all aoifoil shapes provide lift. The results on the other two objectives show that once the lift is involved in the objective function, it becomes more difficult for the algorithm to maintain a consistent improvement

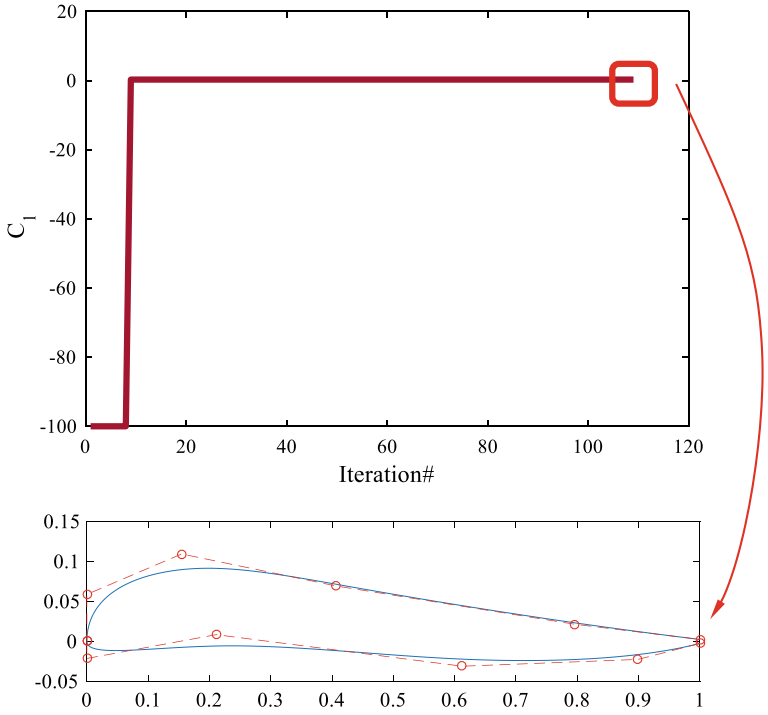

Fig. 12 The convergence curve and final optimal airfoil when maximizing C_l

and steady convergence rate. This is due to the large number of infeasible solutions in each iteration. The accuracy of PSO cannot be improved with fine tuning since the main problem is the failure of PSO mechanisms when there are not at least two feasible solutions in the population.

To see how much the performance of PSO can be improved when considering the lift objective function, the range of variables are bounded significantly and some feasible solutions are added in the first population. The results are shown in Fig. 14. This figure shows that the PSO algorithm shows consistent improvement for such a highly-constraint problem when incorporating the changes.

Taken together, the results of this chapter showed that the PSO algorithm is very beneficial in finding an optimal shape for a 2D airfoil. The experiments on the test functions also showed that tuning the parameters of PSO should be done carefully since using inappropriate values might significantly degrade its performance and change the exploratory and exploitative behaviours.

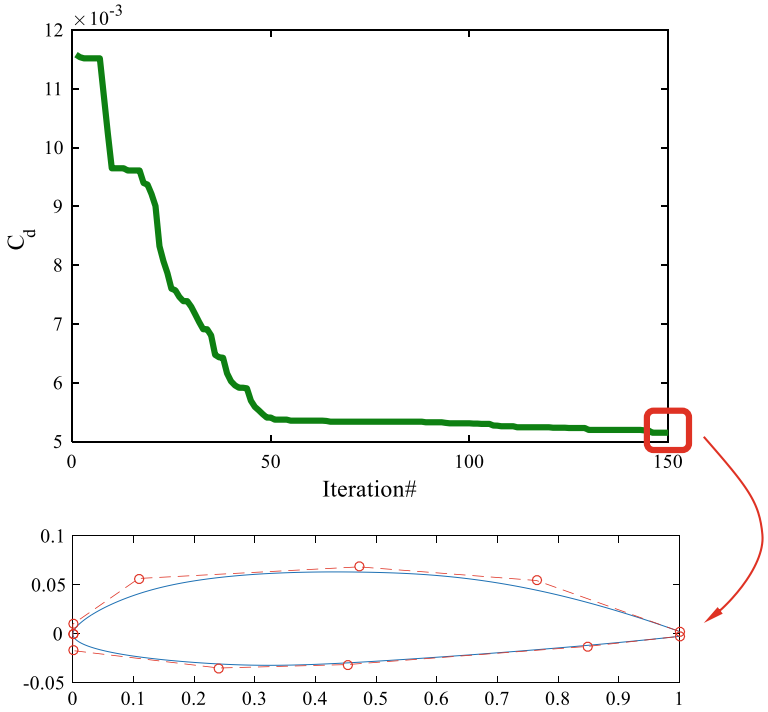

Fig. 13 The convergence curve and final optimal airfoil when minimizing C_d

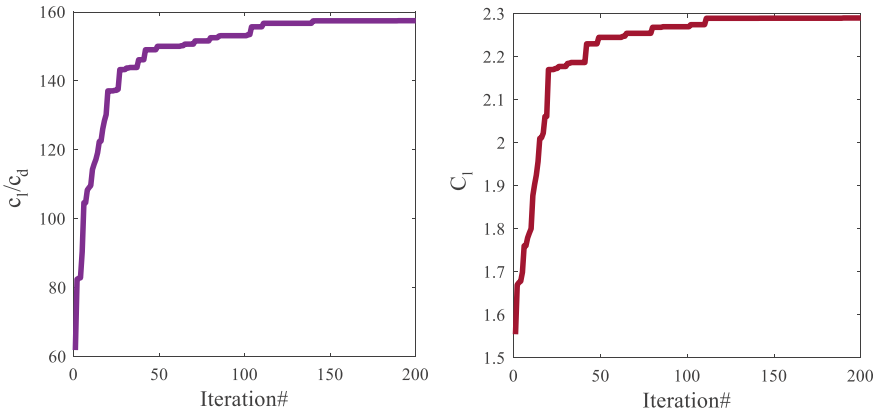

Fig. 14 The convergence curve and final optimal airfoil (maximizing C_l or C_l/C_d) when starting with some feasible solutions and limiting the parameters' bounds

4 Conclusion

This chapter first presented the inspirations and mathematical models of the PSO algorithm. The impact of the parameters on the particle movement and position updating equations was then discussed theoretically. After providing a brief literature review covering different variants, the PSO algorithm was applied to challenging benchmark functions. A large number of figures were used to observe the performance of the PSO algorithm including the search history, convergence curve, average objective of all particles, fluctuation in one variable, and average distance between all particles. The experiments were conducted while changing the values for the main controlling parameters of PSO: inertial weight (w), cognitive constant (c_1), and social constant (c_2).

It was observed that the best balance between exploration and exploitation can be achieved when linearly decreasing the inertial weight. Also, the best value to achieve a similar balance is to set both cognitive and social component to 2. After extensive experiments on the benchmark functions, the chapter employs the PSO algorithm to find an optimal shape for a 2D aifoil considering three different objectives. The results showed that PSO is able to find an optimal design to maximize or minimize any of the objectives.

References

1. Faris, H., Aljarah, I., Al-Betar, M. A., & Mirjalili, S. (2017). Grey wolf optimizer: A review of recent variants and applications. *Neural Computing and Applications*, 1–23.
2. Kirkpatrick, S., Gelatt, C. D.,& Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.
3. Holland, J. H. (1992). Genetic algorithms. *Scientific American*, *267*(1), 66–73.
4. Storn, R., & Price, K. (1997). Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.
5. Dorigo, M., & Birattari, M. (2011). Ant colony optimization. In *Encyclopedia of machine learning* (pp. 36–39). Boston: Springer.
6. Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning* (pp. 760–766). Boston: Springer.
7. Chitsaz, H., & Aminisharifabad, M. (2015). Exact learning of rna energy parameters from structure. *Journal of Computational Biology*, *22*(6), 463–473.
8. Aminisharifabad, M., Yang, Q., & Wu, X. (2018). A penalized autologistic regression with application for modeling the microstructure of dual-phase high strength steel. *Journal of Quality Technology* (in-press).
9. Clerc, M. (2004). Discrete particle swarm optimization, illustrated by the traveling salesman problem. In *New optimization techniques in engineering* (pp. 219–239). Berlin: Springer.
10. Mirjalili, S., & Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, *9*, 1–14.
11. Saremi, S., Mirjalili, S., & Lewis, A. (2015). How important is a transfer function in discrete heuristic algorithms. *Neural Computing and Applications*, *26*(3), 625–640.
12. Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279.

13. Mostaghim, S., & Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS 2003* (pp. 26–33). IEEE.
14. Sierra, M. R., & Coello, C. A. C. (2005). Improving PSO-based multi-objective optimization using crowding, mutation and-dominance. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 505–519). Berlin: Springer.
15. Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation Conference* (pp. 37–48)
16. Reyes-Sierra, M., & Coello, C. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, *2*(3), 287–308.
17. Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for constrained optimization problems. *Intelligent TechnologiesTheory and Application: New Trends in Intelligent Technologies*, *76*(1), 214–220.
18. Pulido, G. T., & Coello, C. A. C. (2004). A constraint-handling mechanism for particle swarm optimization. In *IEEE congress on evolutionary computation* (Vol. 2, pp. 1396–1403).
19. Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11–12), 1245–1287.
20. Eberhart, R. C., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress on Evolutionary Computation, 2001* (Vol. 1, pp. 94–100). IEEE.
21. Blackwell, T., & Branke, J. (2004). Multi-swarm optimization in dynamic environments. In *Workshops on Applications of Evolutionary Computation* (pp. 489–500). Berlin: Springer.
22. Luan, F., Choi, J. H., & Jung, H. K. (2012). A particle swarm optimization algorithm with novel expected fitness evaluation for robust optimization problems. *IEEE Transactions on Magnetics*, *48*(2), 331–334.
23. Mirjalili, S., Lewis, A., & Mostaghim, S. (2015). Confidence measure: A novel metric for robust meta-heuristic optimisation algorithms. *Information Sciences*, *317*, 114–142.
24. Mirjalili, S., Lewis, A., & Dong, J. S. (2018). Confidence-based robust optimisation using multi-objective meta-heuristics. *Swarm and Evolutionary Computation*.
25. Ono, S., Yoshitake, Y., & Nakayama, S. (2009). Robust optimization using multi-objective particle swarm optimization. *Artificial Life and Robotics*, *14*(2), 174.
26. Mirjalili, S., Lewis, A., & Mirjalili, S A M. (2015) Multi-objective optimisation of marine propellers. *Procedia Computer Science*, *51*, 2247–2256.
27. Drela, M. (1989). XFOIL: An analysis and design system for low Reynolds number airfoils. In *Low Reynolds number aerodynamics* (pp. 1–12). Springer, Berlin, Heidelberg.