Chapter 5
Advancements in the Sine Cosine Algorithm

In the last few decades, the development and advancement of meta-heuristic algorithms have become the focus of the research community as these algorithms face various challenges like, balance between exploration and exploitation, tuning of parameters, getting trapped in local optima, and very slow convergence rate. Sine Cosine Algorithm (SCA) also faces similar kinds of challenges and sometimes fails to perform effectively in finding the global optimal solution. Sine and Cosine are trigonometric operators with a 90° phase shift from each other. The range of sine and cosine functions lies in the range $[-1, 1]$. Sine and cosine functions in the position update equation of SCA help solutions to perform search procedure. However, in some situations, SCA promotes similar solutions in the search space, which results in the loss of diversity in the population, and the search process is susceptible to trapping in the region of local optimum [1]. Motivated by these challenges, SCA has been modified to improve its capability and efficiency in several ways. Several strategies have been employed to alter the basic version of SCA [2], aiming to enhance its effectiveness and optimization capabilities. In this chapter, we will discuss about these modifications and strategies, which have been incorporated into the sine cosine algorithm (SCA) in past few years. Apart from this, we will briefly describe the applications of the modified versions of SCA.

The modifications and ensemble of new strategies into the SCA algorithm include—modification in the update mechanism, change in the parameters involved, the introduction of elitism, the introduction of new operators, the introduction of an encoding scheme, the introduction of several statistical distributions for random number generations, etc. For the sake of brevity, we will briefly describe about these modifications and developments in the following manner,

1. Modifications in the position update mechanism
2. Opposition-based learning (OBL) in SCA
3. Quantum-inspired SCA
4. Hybridization of SCA with other meta-heuristics.

© The Author(s) 2023

J. C. Bansal et al., *Sine Cosine Algorithm for Optimization*,
SpringerBriefs in Computational Intelligence,
https://doi.org/10.1007/978-981-19-9722-8_5

5.1 Modifications in the Position Update Mechanism

The position update mechanism or position update operator can be considered as the core of any population-based meta-heuristic algorithm. The movement of the search agents in the search space is controlled by the position update mechanism. It is responsible for updating the current position of the search agents in an intelligent stochastic manner. In the literature of SCA, various modifications in the position update mechanism have been proposed to modify SCA in different ways.

Long et al. [1] proposed an improved version of the SCA (ISCA) for solving high-dimensional problems. This approach is inspired by the integration of the inertia weight (w) in the particle swarm optimizer (PSO) [3]. In this approach, the position update equation is modified by including the concept of inertia weight coefficient (w) to speed up the convergence and prevent local optima entrapment. Furthermore, a new nonlinearly decreasing conversion parameter based on the Gaussian function is introduced to keep the fine-tune balance between SCA's exploration and exploitation phases. The suggested modifications in the position update equation is given by Eq. (5.1).

$$X_{ij}^{t+1} = \begin{cases} w(t) \cdot X_{ij}^{t} + r_1 \cdot \sin(r_2) \times \left| r_3 \cdot P_{gj}^{t} - X_{ij}^{t} \right| & \text{if } r_4 < 0.5 \\ w(t) \cdot X_{ij}^{t} + r_1 \cdot \cos(r_2) \times \left| r_3 \cdot P_{gj}^{t} - X_{ij}^{t} \right| & \text{if } r_4 \geq 0.5 \end{cases} \tag{5.1}$$

where $w \in [0, 1]$ is the inertia weight coefficient. The value of w is linearly decreased from the initial value (w_s) to the final value (w_e) according to the following equation:

$$w(t+1) = w_e + (w_s - w_e) \times \frac{(T - t)}{t} \tag{5.2}$$

where T denotes the maximum number of iterations, and t is the current iteration number.

Along with the introduction of the weight coefficient, Long et al. [1] proposed modifications in the control parameter r_1. The control parameter r_1 is the critical control parameter in the SCA algorithm which helps in controlling the exploration and exploitation phase of the algorithm by controlling the step size. The linearly decreasing value of r_1 helps the algorithm in choosing large step sizes in the initial phase and small step sizes at later phases of the optimization process [2]. However, the linearly decreasing value of r_1 might restrict its convergence rate and accuracy. Long et al. [1] presented a new nonlinearly decreasing strategy for control parameter r_1 based on the Gaussian function, mentioned in Eq. (5.3).

$$r_1(t) = a_e + (a_s - a_e) \times \exp\left(\frac{-t^2}{(m \times T)^2} \right) \tag{5.3}$$

where t indicates the current iteration, T indicates the maximum number of iterations, m is the nonlinear modulation index, and a_s and a_e are the initial and final values of constant a, respectively.

Suid et al. [4] proposed modifications in its update position and in the control parameter r_1 by utilizing the mean of the best search agent's position and the position of the current search agent. In this approach, each agent updates its position dimension-wise with respect to the average of its current position and the best search agent's position to avoid premature convergence. The modified position update equation is given in Eq. (5.4).

$$X_{ij}^{t+1} = \begin{cases} \frac{X_{ij}^t + P_{gj}^t}{2} + r_1 \cdot \sin(r_2) \times \left| r_3 \cdot P_{gj}^t - X_{ij}^t \right| & \text{if } r_4 < 0.5 \\ \frac{X_{ij}^t + P_{gj}^t}{2} + r_1 \cdot \cos(r_2) \times \left| r_3 \cdot P_{gj}^t - X_{ij}^t \right| & \text{if } r_4 \geq 0.5 \end{cases} \tag{5.4}$$

The control parameter r_1 is updated using a nonlinear decreasing mechanism, instead of the linearly decreasing mechanism, as mentioned in Eq. (5.5).

$$r_1 = b \cdot \left(1 - \left(\frac{t}{T}\right)^{\alpha}\right)^{\beta} \tag{5.5}$$

where b is the constant parameter ($a = 2$), T denotes the number of maximum iteration, t is the current iteration, and both α and β are positive real numbers.

Kumar et al. [5] proposed Weibull Pareto sine cosine optimization algorithm (WPSCO), a modification in the sine cosine algorithm (SCA) to solve the peak power detection problem in solar PV panels. In WPSCO, Weibull and Pareto distributions functions are integrated with the SCA algorithm in the position update equation, which improves the convergence rate and enhances the exploitation of the search spaces [5]. In the first stage, the SCA is applied to find the optimal place for all variables (see Eq. 5.6).

$$\Phi = \begin{cases} X_{ij}^t + r_1 \cdot \sin(r_2) \times \left| r_3 \cdot P_{gj}^t - X_{ij}^t \right| & \text{if } r_4 < 0.5 \\ X_{ij}^t + r_1 \cdot \cos(r_2) \times \left| r_3 \cdot P_{gj}^t - X_{ij}^t \right| & \text{if } r_4 \geq 0.5 \end{cases} \tag{5.6}$$

In the second stage, the positions of all variables are analyzed by the Weibull and Pareto distribution function, and the worst regions of the search space are filtered. The position update mechanism for the second stage is given in Eq. (5.7).

$$X_{ij}^{t+1} = \Phi \times \left[1 + \Omega \times \left\{\frac{t}{T} \times W(1, \varepsilon) + \left(1 - \frac{t}{T}\right) \times P(0, \varepsilon, 0)\right\}\right] \tag{5.7}$$

where Ω is the inertia constant. $W(1, \varepsilon)$ is Weibull random number. $P(0, \varepsilon, 0)$ is Pareto random number. ε is the error modulation index described as:

$$\varepsilon = \frac{\left| P_{gj}^{t} - P_{gj}^{t-1} \right|}{\rho_{\max j}^{t}} \tag{5.8}$$

where

$$\rho_{\max j}^{t} = \begin{cases} 1 & \text{if } t = 1 \\ \left| P_{gj}^{t} - P_{gj}^{t-1} \right| & \text{if } \rho_{\max}^{t-1} < \left| P_{gj}^{t} - P_{gj}^{t-1} \right| \quad \text{if } t = 1 \\ \rho_{\max j}^{t-1} & \text{otherwise} \end{cases} \tag{5.9}$$

For maximum power point tracking (MPPT) of partially shaded PV system, Kumar et al. [6] proposed another variant of SCA, called Cauchy and Gaussian sine cosine optimization (CGSCO) algorithm. The CGSCO algorithm combines the Cauchy density [7] and Gaussian distribution function (GCF) [8] with the sine cosine algorithm (SCA). In the proposed method, firstly initial population is updated using the position update mechanism of the SCA algorithm, and then Cauchy and Gaussian mutation mechanisms are employed on the updated population matrix Φ at every iteration. The Cauchy-Gauss mutation mechanism is given in Eq. (5.10).

$$X_{\text{new}} = \Phi + [1 + \delta \times \{\eta \times N(0, 1) + (1 - \eta) \times C(0, 1)\}] \tag{5.10}$$

where $N(0, 1)$ and $C(0, 1)$ are Gaussian and Cauchy random numbers, δ is an inertia constant, $\eta = \frac{t}{T}$, t is the current iteration, and T is the maximum number of iterations. The Cauchy density function enhances the global exploration ability and prevents the algorithm from trapping into the region of local minima. And, the Gaussian distribution function increases the local exploitation capabilities to enhance the rate of convergence of the proposed CGSCO algorithm.

In the position update mechanism, random components are drawn from different distributions. For example, normal distribution, Gaussian distribution, or Cauchy distribution play a very important role in managing the stochasticity of the underlying meta-heuristic algorithm. These random components are responsible for the movement in the search agent's position in the search space by deciding direction and step lengths randomly. In simpler terms, position update mechanisms can be considered as random walks followed by the agents or particles in the search space. A Lévy flight is a specific class of random walk in which the step lengths have a heavy-tailed probability distribution, that is, agents will take a large step sizes occasionally, which in turn improves the exploration capabilities of the underlying algorithm and helps the search agents in escaping local optimal regions of the search space [9].

Inspired by the concept of Lévy flight, Attia et al. [10] proposed a modified sine cosine technique for solving the optimal power flow (OPF) problem by embedding Lévy flight into the position update mechanism of the sine cosine algorithm (SCA). The introduction of Lévy flights in the position update mechanism enhances the global search capabilities of the algorithm, and prevents the agents from being trapped in the regions of local optima. In addition, a fine-tuning capability (i.e., adaptive

tuning of population size strategy) is utilized, in which the size of the population is updated in the following manner:

if

$f_{\min}(t) < \{f_{\min}(t-1), f_{\min}(t-2), f_{\min}(t-3), f_{\min}(t-4)\}$, here t is iteration counter.

Then,

population size = Number of search agents in the $(t-1)$th iteration$\times(1-\alpha)$

else

population size does not change

α is a constant whose value is taken to be 0.05. This adaptive strategy for the population size provides a fast convergence rate to the proposed algorithm.

Similarly, inspired by the concept of Lévy flights, Qu et al. [11] proposed another SCA variant involving Lévy flight. For maintaining a better balance between the exploration and exploitation capabilities of the algorithm, the method of exponentially decreasing conversion (see Eq. 5.11) was applied to the control parameter r_1, and the method of linearly decreasing inertia weight (see Eq. 5.12) was adopted on w. This helps in achieving a smooth transition from global exploration to local development.

$$r_1 = b \cdot \mathrm{e}^{\frac{t}{T}} \tag{5.11}$$

$$w = w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{t}{T} \tag{5.12}$$

Here, T is the maximum number of iterations, t is the current iteration. $w_{\max}$ and $w_{\min}$ denote the maximum and minimum value of the weight parameter, respectively.

Along with the adaptive control parameter strategy, a random neighborhood search strategy is employed, in which a random solution in the vicinity of the optimal solution is used in the position update equation. This allows the algorithm to quickly jump out of the local optimum and increases the diversity of the population. The modified position update equation is mentioned in Eq. (5.13).

$$X_{ij}^{t+1} = \begin{cases} w \cdot X_{ij}^{t} + r_1 \cdot \sin(r_2) \times \mid r_3 \cdot P_{gj}^{t} \times (1 + \lambda \cdot \mathrm{rand}(-1,1)) - X_{ij}^{t} \mid & \text{if } r_4 < 0.5 \\ w \cdot X_{ij}^{t} + r_1 \cdot \cos(r_2) \times \mid r_3 \cdot P_{gj}^{t} \times (1 + \lambda \cdot \mathrm{rand}(-1,1)) - X_{ij}^{t} \mid & \text{if } r_4 \geq 0.5 \end{cases} \tag{5.13}$$

where r_1 and w are the same as mentioned in Eqs. (5.12) and (5.13), respectively. λ is a constant parameter, and r_2, r_3, and r_4 are control parameters.

A self-adapting greedy Lévy mutation strategy is applied to perturb the optimal solution to enhance the local exploitation ability of the algorithm, and to eliminate the defect of low efficiency in a later period [11]. The optimal solution is updated using the following equation:

$$P_{gj}^{t+1} = P_{gj}^{t} + \eta(j) \cdot L \cdot P_{gj}^{t} \tag{5.14}$$

Here, L is a random number drawn from Lévy distribution. P_{gj} is the optimal solution, g is a solution index, and j denotes the dimension at iteration counter t. $\eta(j)$ is the coefficient of self-adapting variation defined in Eq. (5.15).

$$\eta(j) = \mathrm{e}^{\left(-\varepsilon \cdot \frac{t}{T}\right)\left(1-\frac{r(j)}{r_{\max}(j)}\right)} \tag{5.15}$$

Here in Eq. (5.15), ε is a control parameter whose value is chosen to be 30. $r(j)$ denotes the adjusted optimal solution's position given by Eq. (5.16), and $r_{\max}$ denotes the difference between the maximum and minimum value of all the solutions in dimension j (see Eq. 5.17)

$$r(j) = P_{gj}^{t} - \frac{1}{N} \cdot \sum_{i=1}^{N} X_{i,j}^{t} \tag{5.16}$$

$$r_{\max} = \max\left(X_{:,j}^{t}\right) - \min\left(X_{:,j}^{t}\right) \tag{5.17}$$

where N is the population size, t is the current iteration, and T denotes the maximum iteration.

5.2 Opposition-Based Learning Inspired Sine Cosine Algorithm

In this section, we briefly discuss about the concept of opposition-based learning in the sine cosine algorithm. Opposition-based learning (OBL) is a search strategy proposed by Tizhoosh [12] for machine learning applications. It takes into consideration the opposite position of solutions in the search space to increase the chance of finding better solutions in the search space. For a given population, say X, the opposition-based population $\overline{X}$ is calculated in a given manner. Suppose $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ is a solution in X, then $\overline{X_i}$ is calculated using the following equation:

$$\overline{x_{ij}} = u_j + l_j - x_{ij}, \; i = 1, 2 \ldots N; \; j = 1, 2 \ldots D \tag{5.18}$$

where u_j and l_j are the upper and lower bounds of jth dimension, respectively.

The concept of OBL increases the chances of better exploration in the search space and utilizing the opposite positions of solutions in the search space helps in generating a more refined population. For instance, suppose X is a randomly generated population of size N. Using the concept of OBL, a new population $\overline{X}$ is generated using X. Now, there are $2N$ solutions in the search space, and out of these $2N$ solutions, N solutions are selected on the basis of fitness value. That is, fitness values of X and $\overline{X}$ are calculated, and N number of solutions with better fitness values is retained in the population, and the rest of the solutions are eliminated or deleted. For the sake of brevity, two modifications of SCA algorithm using the concept of opposition-based learning (OBL) are discussed below.

Elaziz et al. [13] proposed opposition-based sine cosine algorithm (OBSCA). The authors combined the opposition-based learning strategy with SCA in both the initialization phase and updating phase. In the initialization phase, a randomly generated population (say, X) containing N solutions is initialized, and the concept of OBL is employed to generate the opposition-based population (say, $\overline{X}$). The fitness values of both X and $\overline{X}$ are calculated and N better solutions are retained for the updating phase. In the updating phase, the population is updated using the SCA algorithm, and the opposition-based learning is employed in the updated population. The fitness values of both the population are calculated, and N better solutions are retained for the next iterations, and the rest of the solutions are eliminated. The iterative process is repeated until the termination criteria is satisfied.

Chen et al. [14] proposed a multi-strategy enhanced sine cosine algorithm based on Nelder–Mead simplex (NMs) [15] concept and the opposition-based learning (OBL) strategy for the parameter estimation of photovoltaic models. The Nelder–Mead simplex method is used to deal with unconstrained minimization problems and nonlinear optimization problems. It is a derivative-free direct search method based on functional value comparison. In every iteration, the algorithm first executes the SCA algorithm for updating the population, and then, the OBL mechanism is employed to diversify the population in order to enhance the exploration capability of the algorithm. After the OBL method, the NMs mechanism is incorporated as a local search technique on every solution in order to exploit the potential neighborhood regions of the search space. In detail, the best solution found after using the OBL mechanism in the current population is selected to construct a primary simplex. Then, the simplex is updated according to the NMs simplex mechanism for some k number of iterations, and then the algorithm switched back to the SCA algorithm. The k is a vital parameter whose value is chosen to be $D + 1$, if the optimization problem is D dimensional. The concept of OBL enhances the diversity of the population and benefits the exploration capabilities of the meta-heuristic algorithms. For more applications in the field of soft computing, machine learning, and fuzzy systems, an interested reader can refer to the literature review of opposition-based learning by Mahdavi et al. [16].

5.3 Quantum-Inspired Sine Cosine Algorithm

Apart from the above-mentioned strategies and techniques, researchers have also employed many other methods to modify the sine cosine algorithm. The quantum-inspired meta-heuristics are also becoming popular in recent times. Quantum-inspired meta-heuristics take their inspiration from the various quantum mechanics principles like superposition, uncertainty, inference, entanglement, etc., to model various optimization algorithms [17–19]. The concept of quantum computing, like quantum bits (Q-bits), quantum gates (Q-gates), and their superposition have been combined with various existing meta-heuristic algorithms like particle swarm optimizer [17], gravitational search algorithm [18], gray wolf optimizer [19], to incor-

porate the merits of quantum computing to some extent. Inspired by the concept of quantum computing, Fu et al. proposed chaos quantum sine cosine algorithm (CQSCA) [20]. In the proposed algorithm, a chaotic initialization method and the quantum concept of superposition are used to improve the performance of the SCA algorithm. CQSCA is employed to produce the optimal values for the parameters involved in the support vector machine (SVM) in order to recognize the pattern of different kinds of faults. In the proposed algorithm, the population is initialized with a chaotic variable using a duffing system to enhance the quality of searching global optima [21]. The dynamical equation of the duffing system is given below:

$$x''(t) + \eta x'(t) - \xi x(t) + \mu x^3(t) = A \cdot \cos(\tau t) \tag{5.19}$$

where A is the amplitude of driving force. The coefficient η is the damping degree whose value is taken to be 0.1, and ξ is the toughness degree whose value is chosen as 1. μ is the nonlinearity of power and its value is taken to be 0.25. τ is the circular frequency of the driving force and its value is taken to be 2.

After the chaotic initialization, the inherent characteristics of the qubits and quantum gate concepts help in achieving a better balance between the exploration and exploitation phase of the search process. The proposed algorithm uses quantum bits or qubits[1] to encode the position of search agents in the search space to avoid premature convergence [20]. A qubit can be expressed by probability amplitude P_i using the following equation:

$$P_i = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} p_i^{\mathrm{c}} \\ p_i^{\mathrm{s}} \end{bmatrix} \tag{5.20}$$

where θ denotes the phase shift of a qubit.

Every search agent occupies two positions in the search space, namely the sine position (p_i^{s}) and the cosine position (p_i^{c}), represented by Eqs. (5.21) and (5.22), respectively.

$$p_i^{\mathrm{s}} = [\sin(\theta_{i1}), \sin(\theta_{i2}), \ldots, \sin(\theta_{iD})] \tag{5.21}$$

$$p_i^{\mathrm{c}} = [\cos(\theta_{i1}), \cos(\theta_{i2}), \ldots, \cos(\theta_{iD})] \tag{5.22}$$

where $\theta_{ij} = 2\pi \times \alpha$, and α is a random number in the range [0, 1].

All the encoded search agents update their positions based on an update equation utilizing the features of the SCA algorithm and quantum mechanics. The movements in the search agents are implemented using quantum rotation gate. The position update mechanism of the proposed mechanism is given in Eq. (5.23).

$$P_{\mathrm{inew}} = \begin{bmatrix} p_{\mathrm{inew}}^{\mathrm{c}} \\ p_{\mathrm{inew}}^{\mathrm{s}} \end{bmatrix} \tag{5.23}$$

[1] A qubit is the smallest unit of information in quantum theory.

where p^{c}_{inew} and p^{s}_{inew} are calculated using Eqs. (5.24) and (5.25).

$$p^{c}_{inew} = \left(\cos(\theta^{k}_{i1} + \Delta\theta^{k+1}_{i1}), \cos(\theta^{k}_{i2} + \Delta\theta^{k+1}_{i2}), \cdots \cos(\theta^{k}_{iD} + \Delta\theta^{k+1}_{iD})\right) \tag{5.24}$$

$$p^{s}_{inew} = \left(\sin(\theta^{k}_{i1} + \Delta\theta^{k+1}_{i1}), \sin(\theta^{k}_{i2} + \Delta\theta^{k+1}_{i2}), \cdots \sin(\theta^{k}_{iD} + \Delta\theta^{k+1}_{iD})\right) \tag{5.25}$$

$$\Delta\theta^{k+1}_{ij} = \begin{cases} r_1 \cdot \sin(r_2) \times \Delta\theta^{k}_{g} \\ r_1 \cdot \cos(r_2) \times \Delta\theta^{k}_{g} \end{cases} \tag{5.26}$$

and,

$$\Delta\theta_g = \begin{cases} 2\pi + \theta_{gj} - \theta_{ij}, & \theta_{gj} - \theta_{ij} < -\pi \\ \theta_{gj} - \theta_{ij}, & -\pi \leq \theta_{gj} - \theta_{ij} < \pi \\ \theta_{gj} - \theta_{ij} - 2\pi, & \theta_{gj} - \theta_{ij} > \pi \end{cases} \tag{5.27}$$

Then a mutation operator with quantum non-gate is adopted to avoid local optimum and increase the population diversity [20]. For each search agent, a random number is generated between (0, 1) and is compared with the mutation probability p_m. Then the probability amplitudes of randomly chosen qubits are updated as follows:

$$P_i = \begin{cases} \begin{bmatrix} \cos(\theta_{ij}) \\ \sin(\theta_{ij}) \end{bmatrix} & \text{if } \text{rand}_i < p_m \\ \\ \begin{bmatrix} \sin(\theta_{ij}) \\ \cos(\theta_{ij}) \end{bmatrix} & \text{otherwise} \end{cases} \tag{5.28}$$

Similarly, Lv et al. [22] proposed a quantum encoding scheme inspired modification in the encoding scheme of the search agents in the SCA algorithm. In the proposed algorithm, instead of using real-valued coding for the search agents, the idea of quaternion coding is used. In quaternion encoding, a search agent is expressed as a hyper-complex number containing one real part and three imaginary parts. The real part and imaginary parts of a solution are updated in parallel using the position update mechanism of the SCA algorithm. Quaternions are super-complex numbers represented as mentioned in Eq. (5.29) [23]

$$q = a_0 + a_1 i + a_2 j + a_3 k \tag{5.29}$$

Here, a_0, a_1, a_2, and a_3 are real numbers, and i, j, and k are imaginary numbers following given algebraic rules (Eq. 5.30),

$$\begin{gathered} i \cdot j = k \quad j \cdot i = -k \quad j \cdot k = i \\ k \cdot j = -i \quad k \cdot i = j \quad i \cdot k = -j; \\ i \cdot i = j \cdot j = k \cdot k = -1 \end{gathered} \tag{5.30}$$

The quaternion (q) described in Eq. (5.29) can be further simplified as,

$$q = a_0 + a_1 i + a_2 j + a_3 k = c + d\mathrm{j} \tag{5.31}$$

Here, both c and d are complex numbers, and j is an imaginary part defined as follows,

$$c = a_0 + a_1 i \tag{5.32}$$

$$d = a_2 + a_3 i \tag{5.33}$$

A random population of quaternion encoded search agents (Q) is initialized in the problem's definition domain [L, U], using Eq. (5.34),

$$Q = Q_{\mathrm{R}} + Q_{\mathrm{I}} \cdot i \tag{5.34}$$

where Q_{R} and Q_{I} are complex numbers given by the following equations (see Eqs. 5.35 and 5.36).

$$Q_{\mathrm{R}} = Q_{\mathrm{RR}} + Q_{\mathrm{RI}} \cdot i = \rho_{\mathrm{R}} \cos\theta_{\mathrm{R}} + \rho_{\mathrm{R}} \sin\theta_{\mathrm{R}} \cdot i \tag{5.35}$$

$$Q_{\mathrm{I}} = Q_{\mathrm{IR}} + Q_{\mathrm{II}} \cdot i = \rho_{\mathrm{I}} \cos\theta_{\mathrm{I}} + \rho_{\mathrm{I}} \sin\theta_{\mathrm{I}} \cdot i \tag{5.36}$$

where ρ_{R}, ρ_{I} are random numbers generated in the range,

$$\rho_{\mathrm{R}}, \rho_{\mathrm{I}} \in \left[0, \left(\frac{L-U}{2}\right)\right] \tag{5.37}$$

and, θ_{I}, θ_{R} are random numbers generated in the range,

$$\theta_{\mathrm{I}}, \theta_{\mathrm{R}} \in [-2\pi, 2\pi] \tag{5.38}$$

The quaternion encoded search agents (Q) could be converted to their real-value counterpart X using Eqs. (5.39)–(5.41),

$$X_{\mathrm{R}} = \rho_{\mathrm{R}} \mathrm{sgn}\left(\sin\left(\frac{Q_{\mathrm{RR}}}{\rho_{\mathrm{R}}}\right)\right) + \frac{L+U}{2} \tag{5.39}$$

$$X_{\mathrm{I}} = \rho_{\mathrm{I}} \mathrm{sgn}\left(\sin\left(\frac{Q_{\mathrm{II}}}{\rho_{\mathrm{I}}}\right)\right) + \frac{L+U}{2} \tag{5.40}$$

$$X = \sqrt{\left(X_{\mathrm{R}}^2 + X_{\mathrm{I}}^2\right)} \tag{5.41}$$

The position the search agents (Q_k) is updated using the following position update mechanism mentioned in the equation below:

$$Q_k^{t+1} = \begin{cases} Q_k^t + r_1 \times \sin(r_2) \times \left| r_3 \times \mathrm{QP}_k^t - S_k^t \right| & \text{if } r_4 < 0.5 \\ Q_k^t + r_1 \times \sin(r_2) \times \left| r_3 \times \mathrm{QP}_k^t - Q_k^t \right| & \text{if } r_4 \geq 0.5 \end{cases} \tag{5.42}$$

where $k = \mathrm{RR}, \mathrm{IR}, \mathrm{RI}, \mathrm{II}$. And, QP represents the best solution obtained so far and is represented in the quaternion form as follows in Eq. (5.43),

$$\mathrm{QP} = \mathrm{QP}_{\mathrm{R}} + \mathrm{QP}_{\mathrm{I}} \cdot i \tag{5.43}$$

Here, QP_{R} and QP_{I} are given as follows,

$$\mathrm{QP}_{\mathrm{R}} = \mathrm{QP}_{\mathrm{RR}} + \mathrm{QP}_{\mathrm{RI}} \cdot i \tag{5.44}$$

$$\mathrm{QP}_{\mathrm{I}} = \mathrm{QP}_{\mathrm{IR}} + \mathrm{QP}_{\mathrm{II}} \cdot i \tag{5.45}$$

The incorporation of quantum techniques in the sine cosine algorithm improves the exploration-exploitation capabilities of the algorithm. However, in digital computers, one can not exactly simulate the true nature of quantum computing. Despite of having the limitations, quantum-inspired techniques can be realized as effective techniques for advancements in the existing meta-heuristic algorithms.

5.4 Covariance Guided Sine Cosine Algorithm

Liu et al. [24] proposed an improved version of sine cosine algorithm called covariance guided sine cosine algorithm (COSCA). In COSCA, sine cosine algorithm (SCA) is embedded with the covariance concept to speed up its convergence, and the OBL mechanism to improve the diversity in the population. In every iteration, search agents in the population are sorted based on their fitness value in ascending order. The top $H = \lceil N/4 \rceil$ agents are selected to create a guiding population, say, (P_{Guide}). For updating the position of the agents, the position update mechanism of the SCA algorithm is utilized. After updating the position of every search agent in the population, the opposition-based learning (OBL) strategy is employed. The opposite positions of all the agents are calculated to form an opposite population. The best agents are selected from both the current population and its opposite population to proceed with the search process. The concept of covariance is utilized in the guided population P_{Guide}. The value of the covariance $C_{j,k}$ between any two dimensions j and k in the guided population is calculated using the following equation:

$$C_{j,k} = \frac{1}{H-1} \times \sum_{h=1}^{H} (G_{h,j} - \overline{G}_j) * (G_{h,k} - \overline{G}_k) \quad j,k = 1,2,\ldots D \tag{5.46}$$

where $G_{h,j}$, and $G_{h,k}$ represent the jth and kth dimensions of the hth variable in the guided population (P_{Guide}), respectively. $\overline{G}_j$ and $\overline{G}_k$ denote the mean of the jth and kth dimensions of the guided population. The value of the covariance $C_{j,k}$ forms a covariance matrix C with the size $D \times D$.

Further, eigenvalue decomposition is performed on the covariance matrix C as given in Eq. (5.47).

$$C = OM^2O^{\mathrm{T}} \tag{5.47}$$

Here, M is a diagonal matrix whose elements are equals to the eigenvalues of C, O is an orthogonal matrix, such that each column of O comprises the orthogonal basis for each eigenvector of the covariance matrix C.

For ith search agent X_i, its candidate position Y_i is calculated using the following equation:

$$Y_i = \overline{G} + \sigma \cdot O \cdot M \cdot \gamma \tag{5.48}$$

where $\overline{G}$ is the mean value of the guided population. σ is a zoom factor whose value is taken to be 1.5, γ is a D-dimensional random vector, and the range of each component of γ lies in the [0, 1]. If Y_i is better than X_i, X_i is replaced by Y_i, otherwise, X_i remains unchanged.

5.5 Hybridization of SCA with Other Meta-heuristics

In the context of meta-heuristic algorithms, hybridization refers to the process of integrating two or more existing meta-heuristic algorithms to form a new variant or hybrid algorithm. The hybrid algorithm produced by integrating two or more different algorithms are meant to report better performance when compared to the algorithms, which are used in the process of hybridization. The basic idea of merging two or more existing algorithms is to utilize the merits and strengths of used algorithms while improving their drawbacks. For instance, suppose an algorithm (say) $\mathcal{A}$ is known for better exploration capabilities but suffers from the drawback of weak exploitation, and on the other hand, a different algorithm (say) $\mathcal{B}$ owns better exploitation capabilities but is prone to get stuck in the region of local optimum. The hybrid algorithm (say) $\mathcal{C}$ produced by integrating algorithms $\mathcal{A}$ and $\mathcal{B}$ is supposed to contain the merits of both of the parent algorithms and perform relatively better when compared to both of the algorithms. However, utilizing the techniques of hybridization is a challenging task and requires careful analysis. The process of hybridization can also be achieved using different classical algorithms like simplex methods, Nelder-Mead simplex methods, and local random search techniques [25].

Sine cosine algorithm (SCA) holds the decent ability to achieve a fine balance between the exploration and exploitation phase. However, the performance of SCA can be enhanced using the technique of hybridization for any specific application-oriented problems. Sine cosine algorithm (SCA) has been successfully hybridized with the algorithms like particle swarm optimizer (PSO) [25, 26], genetic algorithm (GA) [27], differential evolution (DE) [28], simulated annealing (SA) [29], gray wolf optimizer (GWO) [30], and artificial bee colony (ABC) algorithm [31], etc. It is beyond the scope of this book to discuss about all the hybridization techniques employed in the SCA algorithm. However, for giving a fair idea to the readers, some of the hybrid algorithms concerning to SCA algorithm are discussed below.

Elaziz et al. [28] proposed a hybridization of sine cosine algorithm (SCA) with differential evolution (DE) algorithm for tackling the feature selection problem. The proposed hybrid algorithm is called SCADE, which has the strengths of DE algorithm and SCA algorithm combined. The feature selection problem is a binary optimization problem, so that solutions in the population represent the binary vectors with length equal to the number of features. Suppose X_i is a solution, and elements of x_i will take values 1 or 0, where 1 represents the selection of the particular feature, while 0 represents the non-selection of the feature. The underlying objective function for evaluating the fitness of the solutions is mentioned below:

$$f(X_i) = \psi \times \mathrm{Err}_{X_i} + (1 - \psi) \times \left(1 - \left(\frac{|S|}{D}\right)\right) \tag{5.49}$$

where Err_{X_i} represents the classification error of the logistic regression classifier with respect to the solution X_i, $|S|$ is the number of selected features, and D is the total number of features in the given data set. $\psi \in [0, 1]$ is a random number used to balance the accuracy of the classifier and the number of selected features.

The normalized fitness value for each solution X_i, s.t. ($i = 1, 2, \ldots, N$, and N is the total number of features), is computed using the following equation:

$$\mathrm{Fit}_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \tag{5.50}$$

The best solution (say) P_g is determined from the population after assigning a fitness value to every solution in the population. In updating phase, the position update mechanism of the DE algorithm or SCA algorithm is used, according to a random value $p \in [0, 1]$. Suppose X_i is a solution and Fit_i represents the normalized fitness value of the solution X_i, if $\mathrm{Fit}_i > P$, then position update mechanism of DE algorithm is utilized, else ($\mathrm{Fit}_i \leq P$) position update mechanism of the SCA algorithm is used. The performance of the hybrid SCADE was tested on UCI datasets, and significant improvement in the classification accuracy of the logistic regression classifier is reported [28]. The hybridization technique is an effective technique for improving the performance and robustness of the underlying meta-heuristic algorithm. In the similar fashion, we will discuss below the hybridization of sine cosine algorithm (SCA) with

gray wolf optimizer (GWO), and the hybridization of sine cosine algorithm with Particle Swarm Optimizer (PSO).

Singh et al. [30] proposed a hybridization of gray wolf optimizer (GWO) [32] and sine cosine algorithm (SCA) [2]. In the proposed hybrid algorithm, the exploitation phase utilizes the GWO algorithm, while the exploration phase incorporates the exploration capabilities of the SCA algorithm. A randomly generated population is initialized, and the fitness value of each search agent is calculated. Based on the fitness value of the search agents, the best search agent (alpha wolf x_α), the 2nd best search agent (beta wolf X_β), and the 3rd best search agent (delta wolf X_δ) is selected, in the same manner as in the GWO algorithm. After this, for the movement of X_α, the position update mechanism of sine cosine algorithm is utilized. The other parameters involved in the GWO algorithm are kept the same, except the position update mechanism of alpha wolf or the best solution X_α. The position update mechanism of X_α is mentioned in Eq. (5.51).

$$d_\alpha = \begin{cases} r_1 \times \sin(r_2) \times |r_3 \times X_\alpha - X_i| & \text{if } r_4 < 0.5 \\ r_1 \times \cos(r_2) \times |r_3 \times X_\alpha - X_i| & \text{if } r_4 \geq 0.5 \end{cases} \tag{5.51}$$

$$X_l = X_\alpha - a_l \times d_\alpha \tag{5.52}$$

where r_1, r_2, r_3, and r_4 are control parameters, as mentioned in the SCA algorithm [2]. And, d_α represents the movement in the X_α. X_l is the next position of the alpha gray wolf [30].

Issa et al. [25] proposed a hybridization of particle swarm optimizer (PSO) with sine cosine algorithm (SCA) in an adaptive manner, and called this hybrid algorithm ASCA-PSO. The proposed hybrid algorithm maintains two layers, namely the bottom layer and the top layer of the solutions based on their fitness value. The bottom layer divides the population into M—different groups, and each group contains N—number of search agents. From every group, a leader y_k (best search agent in a particular group K) is selected for the top layer, and the position update mechanism of PSO is utilized to update the position of the group leaders $y_k, k = 1, 2, \ldots, M$. The bottom layer is responsible for the exploration of the search space and, on the other hand, the top layer is responsible for performing the exploitation phase in the hybrid algorithm [25]. This hybridization ensures a good balance between the exploration and exploitation phase in the entire optimization process [25].

Following the similar trend, Nenavath et al. [33] proposed a hybrid sine cosine algorithm with teaching–learning-based optimization algorithm (SCA–TLBO) to solve global optimization problems and visual tracking. In hybrid SCA–TLBO, first, the standard SCA algorithm is utilized to increase the diversification in the population at the early stages of the search process for exploring the search space extensively, and helping the algorithm in avoiding local optimal regions. After applying the SCA algorithm, search agents are then passed to the teacher-learning phase of the TLBO algorithm in order to move solutions in the direction of the best solution found so far. This strategy helps the proposed algorithm to maintain a fine-tune balance

between the exploration and exploitation phase to perform the global and local search effectively.

Gupta et al. [34] proposed the sine cosine artificial bee colony (SCABC) algorithm, which hybridizes the ABC algorithm with the sine cosine algorithm (SCA). The proposed algorithm improves the exploitation and exploration capabilities of the artificial bee colony (ABC) algorithm. In the ABC algorithm, the employed bee phase plays an important role in exploring more promising regions during the search process. The employed bee phase of the ABC is improved using the SCA, it helps the employed bees to prevent irregular exploration, and increases the efficiency of the proposed hybrid algorithm. The position update mechanism of the employed bee phase in the proposed algorithm utilizes the best solution (or elite solution), and is given in Eq. (5.53).

$$X_i^{t+1} = \begin{cases} P_g^t + \left|\frac{f_{\text{best}}}{f_{\text{worst}}}\right| \times \sin r_2 \times \left|r_3 \times P_g^t - X_i^t\right| & \text{if rand} < 0.5 \\ P_g^t + \left|\frac{f_{\text{best}}}{f_{\text{worst}}}\right| \times \cos r_2 \times \left|r_3 \times P_g^t - X_i^t\right| & \text{otherwise} \end{cases} \tag{5.53}$$

where P_g^t represents the elite (best) solution at the iteration t. f_{best} and f_{worst} represent the best fitness and worst fitness respectively.

Practice Exercises

1. Discuss the rationale behind opposition-based SCA.
2. Explain the levy flight walk. How this concept is implemented in SCA?
3. Discuss the rationale behind using the covariance concept in SCA.
4. Explain the notion of chaos in Chaotic quantum SCA.
5. What do you mean by encoding? Discuss quantum encoding with suitable examples.
6. How is hybridization going to help SCA in giving better results?

References

1. W. Long et al., Solving high-dimensional global optimization problems using an improved sine cosine algorithm. Expert Syst. Appl. **123**, 108–126 (2019)
2. S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. **96**, 120–133 (2016)
3. Y. Shi, R. Eberhart, A modified particle swarm optimizer, in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence* (Cat. No. 98TH8360) (IEEE, 1998), pp. 69–73
4. M. Suid, M. Tumari, M. Ahmad, A modified sine cosine algorithm for improving wind plant energy production. Indones. J. Electr. Eng. Comput. Sci. **16**(1), 101–106 (2019)

5. N. Kumar et al., Peak power detection of PS solar PV panel by using WPSCO. IET Renew. Power Gener. **11**(4), 480–489 (2017)
6. N. Kumar et al., Single sensor-based MPPT of partially shaded PV system for battery charging by using Cauchy and Gaussian sine cosine optimization. IEEE Trans. Energy Convers. **32**(3), 983–992 (2017)
7. M. Ali, M. Pant, Improving the performance of differential evolution algorithm using Cauchy mutation. Soft Comput. **15**(5), 991–1007 (2011)
8. L.S. Coelho, Novel Gaussian quantum-behaved particle swarm optimiser applied to electromagnetic design. IET Sci. Meas. Technol. **1**(5), 290–294 (2007)
9. X.-S. Yang, *Nature-Inspired Optimization Algorithms* (Academic Press, 2020)
10. A.-F. Attia, R.A. El Sehiemy, H.M. Hasanien, Optimal power flow solution in power systems using a novel sine cosine algorithm. Int. J. Electr. Power Energy Syst. **99**, 331–343 (2018)
11. C. Qu et al., A modified sine cosine algorithm based on neighborhood search and greedy levy mutation. Comput. Intell. Neurosci. **2018** (2018)
12. H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCAIAWTIC'06)*, vol. 1 (IEEE, 2005), pp. 695–701
13. M.A. Elaziz, D. Oliva, S. Xiong, An improved opposition-based sine cosine algorithm for global optimization. Expert Syst. Appl. **90**, 484–500 (2017)
14. H. Chen et al., An opposition-based sine cosine approach with local search for parameter estimation of photovoltaic models. Energy Convers. Manag. **195**, 927–942 (2019)
15. J.A. Nelder, R. Mead, A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
16. S. Mahdavi, S. Rahnamayan, K. Deb, Opposition based learning: a literature review. Swarm Evol. Comput. **39**, 1–23 (2018)
17. Y.-W. Jeong et al., A new quantum-inspired binary PSO: application to unit commitment problems for power systems. IEEE Trans. Power Syst. **25**(3), 1486–1495 (2010)
18. M. Soleimanpour-Moghadam, H. Nezamabadi-Pour, M.M. Farsangi, A quantum inspired gravitational search algorithm for numerical function optimization. Inf. Sci. **267**, 83–100 (2014)
19. K. Srikanth et al., Meta-heuristic framework: quantum inspired binary grey wolf optimizer for unit commitment problem. Comput. Electr. Eng. **70**, 243–260 (2018)
20. W. Fu et al., A hybrid fault diagnosis approach for rotating machinery with the fusion of entropy-based feature extraction and SVM optimized by a chaos quantum sine cosine algorithm. Entropy **20**(9), 626 (2018)
21. X.Y. Deng, H.B. Liu, T. Long, A new complex Duffing oscillator used in complex signal detection. Chin. Sci. Bull. **57**(17), 2185–2191 (2012)
22. L. Lv et al., A quaternion's encoding sine cosine algorithm, in *International Conference on Intelligent Computing* (Springer, 2019), pp. 707–718
23. C. Schwartz, Calculus with a quaternionic variable. J. Math. Phys. **50**(1), 013523 (2009)
24. G. Liu et al., Predicting cervical hyperextension injury: a covariance guided sine cosine support vector machine. IEEE Access **8**, 46895–46908 (2020)
25. M. Issa, A.E. Hassanien, D. Oliva, A. Helmi, I. Ziedan, A. Alzohairy, ASCA-PSO: adaptive sine cosine optimization algorithm integrated with particle swarm for pairwise local sequence alignment. Expert Syst. Appl. **99**, 56–70 (2018)
26. H.N. Fakhouri, A. Hudaib, A. Sleit, Hybrid particle swarm optimization with sine cosine algorithm and Nelder-Mead simplex for solving engineering design problems. Arab. J. Sci. Eng. (2019)
27. M.A. El-Shorbagy, M.A. Farag, A.A. Mousa, I.M. El-Desoky, *A Hybridization of Sine Cosine Algorithm with Steady State Genetic Algorithm for Engineering Design Problems* (Springer Nature Switzerland AG, 2020)
28. M.E. Abd Elaziz, A.A. Ewees, D. Oliva, P. Duan, S. Xiong, *A Hybrid Method of Sine Cosine Algorithm and Differential Evolution for Feature Selection* (Springer International Publishing AG, 2017)

29. H. Jouhari, D. Lei, M.A.A. Al-qaness, M. Abd Elaziz, A.A. Ewees, O. Farouk, Sine cosine algorithm to enhance simulated annealing for unrelated parallel machine scheduling with setup times. Mathematics **7**, 1120 (2019)
30. S.B. Singh, N. Singh, A novel hybrid GWO-SCA approach for optimization problems. Eng. Sci. Technol. Int. J. **20**, 1586–1601 (2017)
31. K. Deep, S. Gupta, Hybrid sine cosine artificial bee colony algorithm for global optimization and image segmentation. Neural Comput. Appl. (2019)
32. S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
33. H. Nenavath, R.K. Jatoth, Hybrid SCA-TLBO: a novel optimization algorithm for global optimization and visual tracking. Neural Comput. Appl. **31**(9), 5497–5526 (2019)
34. S. Gupta, K. Deep, Hybrid sine cosine artificial bee colony algorithm for global optimization and image segmentation. Neural Comput. Appl. **32**(13), 9521–9543 (2020)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.