Perpetual Coupled Simulated Annealing for Continuous Optimization

Kayo Goncalves-e-Silva\textsuperscript{1,3,*} and Samuel Xavier-de-Souza\textsuperscript{2,3†}

\textsuperscript{1}Digital Metropolis Institute.
\textsuperscript{2}Department of Computation and Automation.
\textsuperscript{3}Federal University of Rio Grande do Norte, Campus Universitário Lagoa Nova, Natal, Rio Grande do Norte, 59078-900, Brazil.

*Corresponding author(s). E-mail(s): kayo@imd.ufrn.br; contributing authors: samuel@dca.ufrn.br;
†These authors contributed equally to this work.

Abstract

Global optimization heuristics are popular to optimize hard non-convex problems. Despite their irrefutably large cost-to-solution, in the lack of other working greedy or convex approaches, global optimization algorithms remain the no-brainer choice. Nevertheless, successful use often requires tedious adjustments to initial parameters to avoid premature convergence. The Coupled Simulated Annealing approach proposed a method based on the coupling of multiple optimizers to escape premature convergence, having achieved success in optimizing hyperparameters of many applications of machine learning; however, a careful choice of the generation temperature is still required. In this paper we propose the Perpetual Orbit technique as a solution to control the generation temperature and avoid search stagnation. In principle, this technique can also be applied to other ensemble- and population-based algorithms that have a dispersion variable. The results of our experiments show superior performance when using the proposed technique because it makes the Coupled Simulated Annealing totally parameter-free and capable of reaching equal or better solutions in more than 85\% cases across all functions and competitor methods analyzed.

Keywords: Coupled Simulated Annealing, Perpetual Orbit, Global Optimization, Parameter-free Optimization
1 Introduction

The research interest in Metaheuristics for global optimization have been increasing along the past decades, becoming a very fertile field for new strategies [1–10]. Among other positive characteristics, metaheuristics are usually simple to implement, do not require gradient information, can escape from local optima and can be applied to a wide range of real-world optimization [10–15].

Several established algorithms for global optimization require a stochastic distribution to generate new solutions. It is usually necessary to control a variable responsible for the dispersion of the distribution. In Simulated Annealing (SA) [16], for example, the generation and acceptance temperatures are dispersion variables; in Genetic Algorithms, it is a factor in the Blend Crossover [17] and in the Linear Crossover [18]; and, in the Swarm Particle [19], it is the cognitive and social components. Such variables are hard to tune for several reasons, including: the lack of statistical information; the poorly defined operating range; the need to initialize and schedule them; the dependency on the optimization problem; and the demand for a very expensive tuning process due to its empirical nature.

The Coupled Simulated Annealing (CSA) algorithm [20] is a method for global optimization of continuous variables based on SA. It is composed by several distributed SA processes with coupled acceptance probabilities. It is mainly characterized by its acceptance probability function and a coupling term, which is a function of the current energies of all SA processes. CSA is more robust than the uncoupled case with respect to initialization parameter because the acceptance temperature does not require tuning. However, the choice of the generation temperature still affects the optimization process, and defining an initial value and a monotonic annealing scheduling is necessary.

In this paper, we propose a novel technique called Perpetual Orbit (PO) that can control variables responsible for the dispersion of the distribution without prior informations. The main purpose is to maintain or improve the quality of the final solution with an automatic parameter control that, in addition to avoid the convergence to local optima, facilitates the application of metaheuristics by reducing the dependence on metaheuristic’s success rate on the tuning of its initial parameters. In order to prove the effectiveness of the PO technique, we applied it to the generation temperature of the CSA algorithm. The resulting algorithm, called PO-CSA, is a parameter-free algorithm that avoids stagnation of the CSA’s generation temperature. We performed experiments to compare the proposed PO-CSA with classic metaheuristics, such as Cuckoo Search [21], Genetic Algorithm [22], Differential Evolution [23] and Particle Swarm Optimization [19]. We also compared the PO-CSA to the original CSA tuned with an exhaustive search of the initial generation temperature. The results show that the PO-CSA is robust with respect to the initial parameters. It obtains better solutions in most of the cases and outperforms the CSA even with the initial parameters adjusted exhaustively.
2 Coupled Simulated Annealing

Coupled Simulated Annealing (CSA) [20] is a stochastic global optimization method capable of reducing the sensitivity of the initialization parameters while guiding the optimization of continuous variables to quasi-optimal runs. It is based on the Simulated Annealing (SA) [16] and the Coupled Local Minimizers (CLM) [24] algorithms.

The CSA algorithm consists of an ensemble of SA optimizers whose behavior, individually, is similar to the execution of a SA process, i.e. the algorithmic steps that involve generation and acceptance of a single current solution are separately performed for each SA optimizer. In the generation step, the generation temperature $T_{gen}^k$ is responsible for the degree of similarity between the current solution and the probing solution at iteration $k$, i.e. the larger the value of the generation temperature, the larger the average difference between current and probing solutions. The acceptance temperature $T_{ac}^k$, used in the acceptance step at iteration $k$, composes the calculation of the chance of accepting worse probing solutions, i.e., the larger the value of the acceptance temperature, the better the chances of worse probing solutions be accepted.

Unlike SA, where the acceptance probability is a scalar function of the current and probing solutions alone, the CSA acceptance probability function is a scalar function according to

$$0 \leq A_{\Theta} (\gamma, x_i \rightarrow y_i) \leq 1,$$

for every $x_i \in \Theta, y_i \in \Omega,$ and $\Omega \in \mathbb{R}^m$, with $\Theta$ and $\Omega$ being the set of current and the set of all solutions, respectively, and $i = 1, \ldots, m$, with $m$ as the number of elements of $\Theta$. Each solutions $x_i$ has an associated energy $E(x_i)$, equivalent to the cost of the solution $x_i$. Finally, the term $\gamma$ is responsible for the coupling between the SA optimizers.

The acceptance probability function used in this work avoids numerical instabilities due to the energy of the states being positive or negative. It is defined by:

$$A_{\Theta} (\gamma, x_i \rightarrow y_i) = \frac{\exp\left(\frac{E(x_i) - \max E(x_i)}{T_{ac}^k} \right)}{\gamma},$$

where $T_{ac}^k$ is the acceptance temperature at iteration $k$ and $\gamma$ is the coupling term, defined by:

$$\gamma = \sum_{\forall x \in \Theta} \exp\left(\frac{E(x) - \max E(x)}{T_{ac}^k} \right).$$

As the variance of $A_{\Theta}$ is limited by

$$0 \leq \sigma^2 \leq \frac{m - 1}{m^2},$$

a simple rule can be applied to control this variance:

$$\begin{align*}
    & \text{if } \sigma^2 < \sigma_{D}^2, \quad T_{ac}^{k+1} = T_{ac}^k (1 - \alpha), \\
    & \text{if } \sigma^2 \geq \sigma_{D}^2, \quad T_{ac}^{k+1} = T_{ac}^k (1 + \alpha),
\end{align*}$$

(5)
where $\sigma_D^2$ is the desired variance and $\alpha$ the deprecation rate of the temperature, usually values between $[0.0, 0.1]$ [20]. The acceptance temperature is decreased by a factor of $(1 - \alpha)$ if the acceptance variance is below the desired variance, and increased by a factor of $(1 + \alpha)$ otherwise. It has been shown that the optimization performance of CSA improves to a quasi-optimal run when the variance of $A_\Theta$ is around 99% of its maximum value [20]. This way, the variance control substitutes the monotonic schedule of the acceptance temperature. This is very important because it avoids the initial setup of the acceptance temperature in classical SA that often demands an exhaustive search. In contrast, two other initial parameters are introduced, $\alpha$ and $\sigma_D^2$, but these parameters have a well-defined operating range and are much less dependent on the optimization problem at hand [20].

The probing solutions are generated through the Cauchy Distribution according to:

$$y_i = x_i + \epsilon * T_{\text{gen}}^k,$$

where each element of $\epsilon \in \mathbb{R}^m$ is a random independent variable sampled from the Cauchy distribution.

The pseudo-code of the CSA is described in Algorithm 1.

### 3 Perpetual Orbit Technique

The Perpetual Orbit (PO) technique, proposed in this work, is characterized by an ensemble of solutions $s_i \in \Theta$ that cooperate in the effort to find the best value for the dispersion variable during each moment of the execution of a chosen algorithm.

The main focus of this technique is to maintain or improve the quality of the final solution with an automatic parameter control that drives the optimization resiliently to avoid convergence to local optima and reduces the need for initial algorithm settings based on the problem to be applied. It can be potentially applied to other ensemble or population-based approaches. For that, the technique requires that each solution $s_i$ has its own dispersion variable $V_{s_i}^k$ at iteration $k$. Initially, each solution $s_i$ and dispersion variable $V_{s_i}^k$ are randomly initialized and the best initial solution is determined by assessing the costs of all solutions in the ensemble. The value of each $V_{s_i}^k$ changes between its upper and lower bounds, $U_{s_i}^k$ and $L_{s_i}^k$, respectively, derived from the best solution's dispersion variable as follows.

$$U_{s_i}^k = \beta * V_{s_{\text{best}}}^k,$$

$$L_{s_i}^k = \frac{1}{\beta} * V_{s_{\text{best}}}^k,$$

where $s_{\text{best}}$ is the best solution in $\Theta$, and $\beta$ is the boundary multiplier, a valued in the range $[1, 100]$. The parameter $\beta$ is not mandatory and can be removed from the code by simply doing $\beta$ equals 1.

During the optimization, the value of all dispersion variables moves around the value of the best solution’s dispersion variable, except for itself, which remains unchanged. At each iteration, each dispersion value moves towards either the upper or lower bound. When reaching any of these bounds, it changes its direction and moves...
Algorithm 1 Coupled Simulated Annealing Algorithm

1: **Step 1) Initialization:**
2: - Assign $m$ random initial solutions to $\Theta$;
3: - Assign $m$ random initial solutions to $\Theta$;
4: - Assess the energies $E(x_i), \forall x_i \in \Theta$;
5: - Set the iteration number $k = 0$;
6: - Set the initial temperatures $T_{\text{gen}}^k$ and $T_{\text{ac}}^k$;
7: - Assign random value to $\alpha$;
8: - Evaluate the coupling term $\gamma$;
9: - Set $\sigma_D^2 = 0.99 \frac{(m-1)}{m^2}$.

10: **Step 2) Generation:**
11: - Generate a probing solution $y_i = x_i + \epsilon \cdot T_{\text{gen}}^k$, $\forall i = 1, \ldots, m$, according to (6);
12: - Assess the energies $E(y_i), \forall i = 1, \ldots, m$.

13: **Step 3) Acceptance:**
14: - $\forall i = 1, \ldots, m$, do $x_i \leftarrow y_i$ if:
15:   A) $E(y_i) \leq E(x_i)$; or
16:   B) $A_{\Theta} > r$, where $r$ is a random variable sampled from a uniform distribution $[0, 1]$.

17: **Step 4) Update:**
18: - Calculate $\sigma^2$;
19: - Adjust $T_{\text{ac}}^k$ according to the following rules:
20:   - If $\sigma^2 < \sigma_D^2$, then $T_{\text{ac}}^{k+1} = T_{\text{ac}}^k (1 - \alpha)$;
21:   - If $\sigma^2 > \sigma_D^2$, then $T_{\text{ac}}^{k+1} = T_{\text{ac}}^k (1 + \alpha)$.
22: - Decrease the generation temperature $T_{\text{gen}}^k$ according to a chosen monotonic schedule;
23: - Evaluate the coupling term $\gamma$;
24: - Increment $k$ by 1.

25: **Step 5) Stopping Criterion:**
26: - Stop if the stopping criterion is met;
27: - Otherwise, go to Step 2.

towards the other bound. If the upper bound is reached, the bound value is multiplied by a factor of $(1 + \mu)$, for $0 < \mu < 1$. If it is the lower bound that is reached, it is multiplied by $(1 - \mu)$.

The movements of increment and decrement on the value of the dispersion variable, as well as the changes on the bounds, occur continuously until the algorithm finds a new best solution. Then, the new best solution’s dispersion variable stops changing its values, and the old best solution’s dispersion value begins to change around the new value. The upper and lower bound are then reinitialized according to (7).

Consider $D^k_s \in \{-1, 1\}$ as the direction of the solution $s_i$ at iteration $k$. If $D^k_s = 1$, $V^k_s$ has to increase its value by a factor of $(1 + \phi)$ unless it reaches its upper bound.
In that case, $V^s_i$ stops increasing its value and $D^s_i$ becomes $-1$. In a different case, if $D^s_i = -1$, then $V^s_i$ decreases by a factor of $(1 - \phi)$ until it reaches its lower bound. At this time, $V^s_i$ stops decreasing its value and $D^s_i$ becomes $1$. In any case, $\phi$ is a small value between $(0.0, 0.1]$. The standard behavior of the PO technique is illustrated in Fig. 1a. Fig. 1b depicts the behavior of the technique when the upper bound is reached.

The pseudo-code of the Perpetual Orbit technique is presented in Algorithm 2. The PO method can replace the scheduling of the dispersion variable value in ensemble-based algorithms that requires scheduling, which relieves the necessity to define the initial value and type of scheduling. In turn, three other parameters are introduced: $\beta$, $\mu$ and $\phi$. However, these parameters do not require scheduling; their values have a well-defined operating range; they do not demand a very tentative tuning process; and they are not dependent on the optimization problem at hand, as shown by the results in this work. Besides, $\beta$ can be eliminated from the algorithm by simple doing it equals $1$.

### 4 Perpetual Orbit Coupled Simulated Annealing

In CSA, although the control of the variance of the acceptance probabilities makes the optimization more robust with respect to the initial acceptance temperature value, the generation temperature still requires the tuning of a initial value and a monotonic schedule, which is very dependent of the problem at hand. The CSA will stagnate the search for better solutions when its generation temperature is sufficiently diminished, which can lead to convergence to local optima. In order to avoid this, as the CSA is a ensemble-based algorithm and its generation temperature is analogous to a the dispersion variable, the PO technique becomes an interesting strategy. In this case, we use $V^s_i$ as $T^{gen}$, $D^s_i$ as $D_i$, $U^s_i$ as $U_i$ and $L^s_i$ as $L_i$, i.e., each optimizer has its own generation temperature, direction, upper and lower bounds. When combined to the variance control, it turns the CSA into a parameter-free algorithm, and avoids the stagnation of the optimization in local optima.

The CSA acceptance mechanism was slightly adapted to better work with the PO technique. Although the classical acceptance would work for many cases, it can face problems when the objective function is very smooth or when the initial value of the generation temperature is too small. In both cases, there is a tendency for the generation temperature to be constantly reduced. Low values of the generation temperature generate better solutions whose energy is very close to the current solution, i.e. the technique continues to improve the solution with negligible gains. The less the gain, the less the value of the generation temperature and, consequently, the technique stagnates. This behavior happens because the generation temperature does not have a well-defined scope and any change in its value can have enormous impacts on the optimization process. The solution for these problems is to accept a probing solution only if there is a minimum percentage gain of $\delta$, a small value between $(0.0, 0.05]$. This strategy prevents the acceptance of probing solutions with negligible gain, forcing the technique to increase the value of $T^{gen}$ to an acceptable level where the gains are more expressive. This strategy may not be necessary for other algorithms where the
New $U_{s_2}^{k+1}$

Old $U_{s_2}^k$

$D_{s_2}^{k+1} = -1$

$D_{s_2}^k = 1$

Best $s_i$

Best $s_i$

$D_{s_2}^{k+1} = 0$

$D_{s_2}^k = 0$

$D_{s_2}^{k+1} = 1$

Dispersion variable does not need to follow an increasing or decreasing schedule. The pseudo-code for the PO-CSA algorithm is presented in Algorithms 3 and 4.

Fig. 1: The Perpetual Orbit technique. (a) The standard behavior; and (b) The increase of $U_{s_2}^{k+1}$ after $V_{s_2}^{k+1}$ reaches the upper bound.
Algorithm 2 Perpetual Orbit technique

1: Step 1) Initialization:
2: - Initialize the chosen algorithm;
3: - Initialize $D_{k_i}^{s_i} \in \{-1, 1\}$ and $V_{k_i}^{s_i}$ randomly, $\forall s_i \in \Theta$;
4: - Initialize $U_{k_i}^{s_i}$ and $L_{k_i}^{s_i}$, $\forall s_i \in \Theta$, according to (7);

5: Step 2) Generation:
6: - Generate and assess probing solutions according to the chosen algorithm;

7: Step 3) Evaluation:
8: - Accept the probing solutions if it follows the chosen algorithm’s acceptance criterion;
9: - If there is a new best solution, then
10: - Update the Bounds of all solutions according to (7);

11: Step 4) Update:
12: 4.1) Update the chosen algorithm:
13: - Update variables of the chosen algorithm;
14: 4.2) Update the Dispersion Variable:
15: - If ($s_i \neq \text{best}(V_{k_i}^{s_i})_{s_i \in \Theta}$) then
16: - If ($D_{k_i}^{s_i} > 0$) then $V_{k_i+1}^{s_i} = V_{k_i}^{s_i}(1 + \phi)$;
17: - If ($D_{k_i}^{s_i} < 0$) then $V_{k_i+1}^{s_i} = V_{k_i}^{s_i}(1 - \phi)$;
18: Else
19: - $V_{k_i+1}^{s_i} = V_{k_i}^{s_i}$
20: 4.3) Update the Upper and Lower Bound:
21: - If ($D_{k_i}^{s_i} > 0$) and ($V_{k_i}^{s_i} \geq U_{k_i}^{s_i}$) then
22: - $V_{k_i+1}^{s_i} = V_{k_i}^{s_i}$; $D_{k_i+1}^{s_i} = -1$; $U_{k+1}^{s_i} = U_{k_i}^{s_i}(1 + \mu)$;
23: - If ($D_{k_i}^{s_i} < 0$) and ($V_{k_i}^{s_i} \leq L_{k_i}^{s_i}$) then
24: - $V_{k_i+1}^{s_i} = V_{k_i}^{s_i}$; $D_{k_i+1}^{s_i} = 1$; $L_{k+1}^{s_i} = L_{k_i}^{s_i}(1 - \mu)$.

25: Step 5) Stopping Criterion:
26: - Stop if the stopping criterion is met;
27: - Otherwise, go to Step 2.

The PO-CSA algorithm starts with Initialization step by setting initial variables, such as the initial solutions, the generation and acceptance temperatures, the upper and lower bounds of $T_{\text{gen}}^k$, and the direction of each solution. The energy of each initial solution is assessed and the coupling term is calculated.

In Generation, the PO-CSA behaves like the CSA, by generating probing solutions and assessing their energies. This generation needs a statistical distribution, which in turn needs a dispersion variable. In the CSA and PO-CSA, the generation temperature is used as the dispersion variable.

In Acceptance, a better solution is always accepted and a worse probing solution is accepted if it is better than the current solution by a percentage gain of $\delta$, at least,
Algorithm 3 Part 1 - Perpetual Orbit Coupled Simulated Annealing

1: **Step 1) Initialization:**

2: \hspace{1em} 1.1) CSA

3: - Assign \(m\) random initial solutions to \(\Theta\);

4: - Assess the Energy \(E(x_i), \forall x_i \in \Theta\);

5: - Set the number of iterations \(k = 0\);

6: - Set the initial acceptance temperature;

7: - Assign random value to \(\alpha\);

8: - Evaluate the coupling term \(\gamma\);

9: - Set \(\sigma_D^2 = 0.99\frac{(m-1)}{m^2}\).

10: \hspace{1em} 1.2) PO

11: - Assign random values to \(\beta\), \(\delta\), \(\mu\) and \(\phi\);

12: - Initialize \(D_k^i \in \{-1, 1\}\) and \(T_k^{gen}\) randomly, \(\forall i \in \Theta\);

13: - Initialize the \(U_k^i\) and \(L_k^i\) according to (7);

14: **Step 2) Generation:**

15: \hspace{1em} 2.1) CSA

16: - Generate a probing solution \(y_i = x_i + \epsilon \ast T_k^{gen}, \forall i = 1, \cdots, m\), according to (6);

17: - Assess the energies \(E(y_i), \forall i = 1, \cdots, m\).

18: **Step 3) Acceptance:**

19: \hspace{1em} 3.1) CSA

20: - \(\forall i = 1, \cdots, m\), do \(x_i \leftarrow y_i\) if:

21: \hspace{1em} \hspace{1em} A) \(y_i\) is better than \(x_i\) in, at the very least, \(\delta\); OR

22: \hspace{1em} \hspace{1em} B) \(A_\Theta > r\), where \(r\) is a random variable that was sampled from a uniform distribution \([0,1]\).

23: \hspace{1em} 3.2) PO

24: - If there is a new best solution, then

25: - Update the Bounds of all solutions according to (7).

or, with probability \(A_\Theta(\gamma, x_i \rightarrow y_i)\). If any accepted solution is the new best solution, each optimizer has to update its upper and lower bounds \(U_k^i\) and \(L_k^i\), respectively, according to (7).

Then, the algorithm need to Update the acceptance temperature and the generation temperatures. The first one is updated by following the rule described in (5). The coupling term is reevaluated. The generation temperatures of each solution, \(T_k^{gen}\), follow the perpetual orbit, where they move around the best solution’s generation temperature. The upper or lower bound is updated if a solution’s generation temperature reaches it. Any optimizer that finds the best solutions keeps its generation temperature unchanged and becomes the new reference for all other generation temperatures. Each optimizer, then, has to update its upper and lower bounds \(U_k^i\) and \(L_k^i\), respectively.

The algorithm finishes its execution when the Stopping Criterion is met. Otherwise, the loop restarts through a new execution of the Generation step.
Algorithm 4 Part 2 - Perpetual Orbit Coupled Simulated Annealing

1: Step 4) Update:
2: 4.1) CSA
3: - Calculate $\sigma^2$;
4: - Adjust $T_{ac}^{k+1}$ according to the following rules:
5: - If $\sigma^2 < \sigma_0^2$ then $T_{ac}^{k+1} = T_k^{ac}(1 - \alpha)$;
6: - If $\sigma^2 > \sigma_0^2$ then $T_{ac}^{k+1} = T_k^{ac}(1 + \alpha)$.
7: - Evaluate the coupling term $\gamma$;
8: 4.2) PO
9: 4.2.1) Update the Generation Temperature:
10: - If ($i \neq \text{best}(T_{gen}^i)_{i \in \Theta}$) then
11: - If ($D_i^i > 0$) then $T_{gen}^{k+1} = T_k^{gen}(1 + \phi)$;
12: - If ($D_i^i < 0$) then $T_{gen}^{k+1} = T_k^{gen}(1 - \phi)$;
13: Else
14: - $T_{gen}^{k+1} = T_{gen}^k$
15: 4.2.2) Update the Upper and Lower Bound:
16: - If ($D_i^i > 0$) and ($T_{gen}^k \geq U_i^k$) then
17: - $T_{gen}^{k+1} = T_k^{gen}$; $D_i^{k+1} = -1$;
18: - $U_i^{k+1} = U_i^k(1 + \mu)$;
19: - If ($D_i^i < 0$) and ($T_{gen}^k \leq L_i^k$) then
20: - $T_{gen}^{k+1} = T_k^{gen}$; $D_i^{k+1} = +1$;
21: - $L_i^{k+1} = L_i^k(1 - \mu)$;
22: - Increment $k$ by 1.
23: Step 5) Stopping Criterion:
24: - Stop if the stopping criterion is met;
25: - Otherwise, go to Step 2.

5 Experiments & Results

We performed several experiments to assess the optimization potential of the proposed PO-CSA algorithm. We performed the experiments on a cluster with 64 nodes, each one with 2 Intel Xeon Sixteen-Core E5-2698v3 CPUs at 2.3 GHz, and 128 GB of RAM DDR4, from the High Performance Computing Center at UFRN (NPAD/UFRN). We used the same 14 objective functions used in the experiments with the original CSA, better described in Appendix A. The initialization of the generation temperatures and its schedule as well as the method for generating probing solutions are explained in the following subsection. The results of the experiments are presented in section 5.2.

5.1 General Settings

In the steps 2 and 4 of the CSA, it is necessary to generate and assess a new probing solutions, and decrease a generation temperature $T_{gen}^k$, previously initialized with a value $T_{gen}^0$, according to a chosen monotonic schedule, respectively. SA-based algorithms tend to be very sensitive to different schedules and initial parameters. The
impact of these choices goes beyond the convergence of the algorithm, directly impacting the quality of the final solution. In the last two of the four sets of experiments performed in this work, two initialization procedures were used for the CSA to compare the resulting behavior with the PO-CSA: R-CSA and B-CSA. They work exactly as the original CSA, however the first one had its initial generation temperature initialized as a random number between \([0, 100]\), just like it was done for the PO-CSA, while the B-CSA shows the best results from the best initial generation temperature chosen from a predefined set of initial generation temperatures:

\[
T_0^{\text{gen}} \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\},
\]

i.e. for each of the experiments that is presented here, the B-CSA shows the best solution among the solutions found by the CSA with these seven different initial generation temperatures.

The R-CSA and the B-CSA were subjected to the same generation temperature schedule [25]:

\[
T_k^{\text{gen}} = \frac{T_0^{\text{gen}}}{k+1},
\]

By choosing the same generation temperature schedule to both algorithms, we hope to better demonstrate the behavior of the PO-CSA algorithm. There was no study to conclude which schedule would be the best suit to run each function on the B-CSA and the R-CSA.

In the R-CSA, B-CSA and PO-CSA, we set the increase/decrease acceptance temperature factor \(\alpha = 0.05\) and the desired variance of the acceptance probabilities to 99% of its maximum value. Exclusively for the PO-CSA, we used the boundary multiplier \(\beta = 10\), the increase/decrease boundary factor \(\mu = \alpha\) and percentage gain \(\delta = 0.001 (0.1\%)\) for all functions and experiments.

5.2 Results

The experiments were divided in four sets. The first set compares the quality of solution of the PO-CSA and four reference algorithms. The second set shows the behavior of the generation temperature when the PO works combined with the CSA. The third set demonstrates the robustness of the quality of the final solutions of the PO-CSA comparing it to the CSA with two different initialization procedures: R-CSA and B-CSA. The fourth set shows the resilience of the proposed method for a larger number of function evaluations by comparing it to the R-CSA and the B-CSA cases using the function where PO-CSA performed worst in the third set of experiments. These experiments and their results are detailed as follows.

5.2.1 Comparing PO-CSA to Classical Algorithms

In this set of experiment, we compare the performance of the PO-CSA to the average solutions found by four of the most used heuristics in scientific applications [26]: the Cuckoo Search via Lévy Flight (CS) [21, 27], Differential Evolution (DE) [23, 28, 29], Particle Swarm Optimization (PSO) [19, 29], and Genetic Algorithm (GA) [17, 18,
The stopping criterion was $D \times 10^6$ function evaluations, where $D$ is the dimension of the problem. The population size for CS, DE and PSO was 50 [26]. As in [31] the recommended population size of the GA is no greater than 100, and in [26] the recommended the population size is 50 for CS, DE, and PSO, we decided to use the same population size for GA. The number of optimizers used in the the PO-CSA was $D$. The results are averaged across 25 runs.

In order to avoid an incessant search for better parameters for the classic algorithms, we have used the following values of algorithmic control parameters according to the references:

- CS: $\beta = 1.5$ and $p_0 = 0.25$ [26, 32];
- DE: DE/rand/1 mutation strategy with $F = 0.50$ and $Cr = 0.90$ [26, 33–35];
- PSO: $C_1 = C_2 = 1.80$ and $\omega = 0.60$ [26, 35];
- GA: Blend Crossover (BLX-$\alpha$) with $\alpha = 0.5$ [36, 37];
- PO-CSA: $\beta = 10$, $\delta = 0.001$ (0.1%) and $\mu = \alpha = 0.05$.

The average results are shown in Table 1. The average solutions found by the PO-CSA are equal or better than the other algorithms in 85.71% of the cases, against 2.38% for the CS and GA, 9.52% for the DE and 17.85% for the PSO. There is a tie of, at least, 2 metaheuristics in 15.47% of the cases.

### 5.2.2 Analysis of the Generation Temperature Control

In this section, we analyze the effects of the generation temperature control through PO on the CSA, as detailed in Section 4. Fig. 2 presents the PO-CSA generation temperature’s behavior on $f_3$ for three different initial generation temperature: 0.001, 1 and 1000. Fig. 2a depicts their reference generation temperature, i.e. the value where the generation temperature of the other solutions has to move around it. This reference is the generation temperature of the best solution, as described in section 3. Fig. 2b shows all the generations temperatures around the reference generation temperatures. Note that the convergence of PO-CSA for $T_{gen}^0 = 0.001$ is slower than the others because its initial value is too small. The control needs a few thousands of iterations before the energy starts descending in a rate similar to the other initial temperatures. The same happens to $T_{gen}^0 = 1000$, but, in this case, it is because its initial value is too large. Although this behavior reflects in the quality of solution, all three runs converge to the same level of energy and generation temperature after a transient.

Fig. 3 presents the best energy generated by results of the generation temperature control in Fig. 2. After a transient, all three runs converge to the same level of energy. Observe that the generation temperature convergence impacts on the quality of solutions. As long as the algorithms does not find a good reference generation temperature, the quality of the solution does not improve significantly. However, after reaching this reference, there is a sufficiently great improvement in the quality of the solution.

### 5.2.3 Robustness of Initialization

In this set of experiments, we compared the quality of the solutions found by the PO-CSA to those found by the R-CSA and B-CSA. The R-CSA has its initial generation temperature as a random number between [0,100], while the B-CSA shows the best
Table 1: Statistical average solution of the CS, DE, GA, PSO and PO-CSA for functions $f_1$ – $f_{14}$ with $D \in \{5, 10, 15, 20, 25, 30\}$ and the number of optimizer equals $D$. The stopping criterion is $1D \times 10^6$ function evaluations. The population size for Cuckoo, DE, GA, PSO is 50. The number of optimizers of the PO-CSA is $D$. The results are averaged across 25 runs. The best results for each dimension in a given number of function evaluations are highlighted in bold.
Fig. 2: The PO-CSA behavior on $f_3$ along 10 thousand iterations with $D = 10$ for three different initial generation temperatures. The number of optimizers equals $D$. After a transient, all three runs converge to the same level of energy. (a) The reference for the generation temperature, (b) The generation temperature for all optimizers.

results from the best initial generation temperature chosen from a predefined set of initial generation temperatures, as defined in (8).
Fig. 3: The best PO-CSA energies on $f_3$ along 10 thousand iterations with $D = 10$ for three different initial generation temperatures. The number of optimizers equals $D$. After a short transient, all three runs converge to the same level of energy.

In order to perform these experiments, we used six different values for the dimension of the problems $D \in \{5, 10, 15, 20, 25, 30\}$ as the same number of SA optimizers. The stopping criterion for the three algorithms was 1, 2, 4 and 8 million of function evaluations per each optimizer and results were averaged across 25 runs. The mean results are shown in Table 2 and Table 3.

The results of the B-CSA are always better than or equal to the ones from the R-CSA, as expected, given that the initial values of B-CSA were chosen after exhaustive executions with several different initial generation temperatures. Since the PO-CSA is parameter-free, the results of the R-CSA and B-CSA, then, represents the minimum goal and the golden goal for the PO-CSA, respectively. No so surprisingly, the PO-CSA reached equal or better solutions than the R-CSA in 98.21% of the cases, only losing in $f_2$ in 5 cases and 1 case in $f_3$. However, the effectiveness of the PO-CSA exceeds expectations when it is equal or better than the B-CSA in 86.90% of the tests. The remaining 13.10% cases, where PO-CSA looses for B-CSA, are distributed in functions $f_2$, $f_3$, $f_4$, $f_5$, $f_9$, $f_{10}$, $f_{11}$, $f_{12}$, $f_{13}$, and $f_{14}$. For the PO-CSA, the worst case is for the function $f_2$.

5.2.4 The Resilience of the PO-CSA

In this set of experiments, we compared the quality of the solutions found by the R-CSA, B-CSA and PO-CSA for $f_2$, whose PO-CSA’s failure rate was the highest in the previous set of experiment, to understand the behavior of the three algorithms when
Table 2: Statistical mean solution of the R-CSA, the B-CSA and the PO-CSA for functions $f_1$ – $f_7$ with $D \in \{5, 10, 15, 20, 25, 30\}$ and the number of optimizer equals $D$. The stopping criterion goes from $1 \times 10^6$ to $8 \times 10^6$ function evaluations per optimizer. The results are averaged across 25 runs. The best results for each dimension in a given number of function evaluations are highlighted in bold.

| Algorithm | Function | $D = 5$ | $D = 10$ | $D = 15$ | $D = 20$ | $D = 25$ | $D = 30$ |
|-----------|----------|---------|----------|----------|----------|----------|----------|
| BC-SGA    | $f_1$    | $1.75 \times 10^{-2}$ | $1.05 \times 10^{-2}$ | $4.93 \times 10^{-3}$ | $8.04 \times 10^{-4}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
| BC-SGA    | $f_2$    | $3.64 \times 10^{-2}$ | $9.47 \times 10^{-3}$ | $1.24 \times 10^{-3}$ | $1.38 \times 10^{-4}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
| BC-SGA    | $f_3$    | $5.11 \times 10^{-3}$ | $7.20 \times 10^{-4}$ | $1.07 \times 10^{-4}$ | $1.81 \times 10^{-5}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
| BC-SGA    | $f_4$    | $3.64 \times 10^{-2}$ | $9.47 \times 10^{-3}$ | $1.24 \times 10^{-3}$ | $1.38 \times 10^{-4}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
| BC-SGA    | $f_5$    | $5.11 \times 10^{-3}$ | $7.20 \times 10^{-4}$ | $1.07 \times 10^{-4}$ | $1.81 \times 10^{-5}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
| BC-SGA    | $f_6$    | $3.64 \times 10^{-2}$ | $9.47 \times 10^{-3}$ | $1.24 \times 10^{-3}$ | $1.38 \times 10^{-4}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
| BC-SGA    | $f_7$    | $5.11 \times 10^{-3}$ | $7.20 \times 10^{-4}$ | $1.07 \times 10^{-4}$ | $1.81 \times 10^{-5}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ | $0.00 \times 10^{0}$ |
Table 3: Statistical mean solution of the R-CSA, B-CSA and PO-CSA for functions \( f_s \) \( f_{14} \) with \( D \in \{5,10,15,20,25,30\} \) and the number of optimizer equals \( D \). The stopping criterion goes from \( 1 \times 10^6 \) to \( 8 \times 10^6 \) function evaluations per optimizer. The results are averaged across 25 runs. The best results for each dimension in a given number of function evaluations are highlighted in bold.
we have an increased budget of function evaluations. In order to perform these experiments, we used dimension $D \in \{5, 10, 15, 20, 25, 30\}$ and the number of optimizers equals $D$. The stopping criterion for the three algorithms was varied from $1.0 \times 10^6$ to $16.0 \times 10^6$ and results were averaged over 25 runs.

Observe in Fig. 4 that, although the PO-CSA does not beat the B-CSA in all cases, even the R-CSA for $D \in \{25, 30\}$ with 1 and 2 million of function evaluations, the larger the number of functions evaluations, the better the results of the PO-CSA. For 4, 8 and 12 million of function evaluations, the PO-CSA wins the R-CSA in all cases, but it still does not win the B-CSA for $D \in \{25, 30\}$. For 16 million of function evaluations, the PO-CSA wins the B-CSA in 100% of the cases. Besides, the PO-CSA found the global optimum for $D = 5$ when running 8 and 12 million of functions evaluations as well as for $D = 5, 10$ for 16 million of functions evaluations, which represents the lack of marking dots for PO-CSA in Fig. 4d, Fig. 4e and Fig. 4f because zero is not defined in logarithmic scale. This behavior happens because the PO-CSA does not stagnates its searching processes, since the generation temperature is not monotonically decreasing. On the other hand, increasing the number of function evaluations of the CSA can stagnate its search because the generation temperature can reach a vary small value.

6 Conclusion

We have presented an algorithm based on Coupled Simulated Annealing (CSA) that controls all its setting parameters. The Perpetual Orbit Coupled Simulated Annealing (PO-CSA) was designed by applying a novel technique called Perpetual Orbit (PO) to the CSA algorithm. Due to the coupling, the CSA is robust against the initialization of the acceptance temperature. However, the generation temperature requires tuning to avoid search stagnation. The PO technique arises as a solution that control the generation temperature, which no longer decreases monotonically. In principle, PO can also be applied to other ensemble- and population-based algorithms that have a dispersion variable.

We performed four sets of experiments to prove the effectiveness of the PO-CSA. The experiments were performed using 14 objective functions. In the first set of experiments, we compared the quality of solutions found by the PO-CSA to those found by the Cuckoo Search via Lévi Flight, Differential Evolution, Particle Swarm Optimization and Genetic Algorithm. Results show that the PO-CSA found equal or better solutions in 85.71% of the cases.

In other two sets of experiments, we compared the PO-CSA to the CSA with two different initialization procedures. The first procedure, called R-CSA, has its initial generation temperature initialized as a random number. The second one, B-CSA, shows the best solutions through an exhaustive search of the initial generation temperature. These two procedures had the same monotonically decreasing temperature schedule. Results from the R-CSA are the first frontiers that the PO-CSA must overcome. Those from the B-CSA express excellence in quality of solution. The comparison shows show that the PO-CSA overcomes the R-CSA in 98.21% of the cases and the B-CSA in 86.90%.
Fig. 4: Average energy of $f_2$ over 25 runs for different numbers of function evaluations per optimizer. The number of optimizers is the same as the dimension. In (d) and (e) for $D = 5$, and (f) for $D = 5, 10$, there are no points for PO-CSA because its costs are zero (global optimum) and not defined in logarithmic scale. The larger the number of function evaluations, the better the results of the PO-CSA because it does not stagnates its search.

For the worst case, where the PO-CSA performed less than the B-CSA, the experiments show that the R-CSA and B-CSA do not improve the quality of the solution...
when the number of function evaluations increases. However, the same does not happen with the PO-CSA due to the non-stagnation of the generation temperature introduced by the PO technique. In this case, sufficiently increasing the number of function evaluations makes the solutions of the PO-CSA equal or better than those of the B-CSA in 100% of the cases.

Lastly, the set of experiments with the analysis of the generation temperature control introduced by the PO technique shows why the PO-CSA presents superior performance. By allowing the independent control of its initial setting parameter, the PO technique makes the PO-CSA parameter-free and capable of reaching better solutions than the original CSA in the majority of functions for all competitor methods tested.

Appendix A  The Objective Functions

In order to guarantee the satisfactory of the PO-CSA, experiments were carried out repeatedly in 14 reference functions $f_1 - f_{14}$, with $D$ dimensions, used in [20]. The global optimum for all functions is $f^*(x) = 0$. They are detailed below.

1. **(Group 1: $f_1 - f_2$)** Unimodal and simple multimodal functions: the function 1 is an easy unimodal sphere function. The second function is the ubiquitous Rosenbrock’s function, which is very often used for testing optimization algorithms. They are described in Table A1.

   **Table A1: Unimodal and simple multimodal functions: test problems, Group 1.** The minimum of both functions is zero. The dimensionality of these problems can be adjusted with the term $D$.

   | No. | Function $f(x)$ | Input range |
   |-----|-----------------|-------------|
   | 1   | $f_1 = \sum_{i=1}^{D} x_i^2$ | [-100,100] |
   | 2   | $f_2 = \sum_{i=1}^{D-1} (1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2$ | [-2.048,2.048] |

2. **(Group 2: $f_3 - f_8$)** Multimodal Functions: A collection of multidimensional and multimodal continuous functions was chosen from the literature to be used as test cases. These functions feature many local minima and therefore are regarded as being difficult to optimize [38, 39]. They are detailed in Table A2.

3. **(Group 3: $f_9 - f_{14}$)** Nonseparable Functions: The functions of the Group 2 are considered hard to optimize, but they can possibly be separable. This condition means that the minimization problem can be solved using $D$ unidimensional searches, where $D$ is the dimension of the problem. Various real-life optimization problems are nonseparable. Therefore, to approximate these problems, in this group of test problems, we use a set of rotated versions of the Group 2. The rotated functions preserve the same shape characteristics as those of the original functions but cannot be solved by $D$ unidimensional searches. As described in [20], in order to rotate a
Table A2: Multimodal functions: test problems, group 2. These functions present many local minima and are considered hard problems to optimize, particularly in large dimensions. The minimum of all these functions is zero. The dimensionality of these functions can be adjusted with the term $D$.

| No. | Function $f(x)$                                                                 | Input range     |
|-----|--------------------------------------------------------------------------------|-----------------|
| 3   | $f_3 = -20 \exp \left[ -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} - \exp \left[ \frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right] + 20 + e \right]$ | $[-32.768,32.768]$ |
| 4   | $f_4 = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos \left( \frac{\pi \cdot 20}{\sqrt{D}} \right) + 1$ | $[-600,600]$ |
| 5   | $f_5 = \sum_{i=1}^{D} \left\{ \sum_{k=0}^{20} (0.5)^k \cos \left( \frac{2 \pi \cdot 3^k (x_i + 0.5)}{2} \right) \right\} - D \sum_{k=0}^{20} (0.5)^k \cos(\pi 3^k) + 20 + e$ | $[-0.5,0.5]$ |
| 6   | $f_6 = \sum_{i=1}^{D} x_i^2 - 10 \cos(2\pi x_i) + 10$ | $[-5.12,5.12]$ |
| 7   | $f_7 = \sum_{i=1}^{D} y_i^2 - 10 \cos(2\pi y_i) + 10$ , $y_i = \begin{cases} \frac{x_i}{4} & |x_i| < \frac{1}{2} \\ \text{round}(2x_i) \end{cases}$ for $|x_i| \geq \frac{1}{2}$, $\forall i = 1, \ldots, D$ | $[-5.12,5.12]$ |
| 8   | $f_8 = 419 \times D + \sum_{i=1}^{D} \sin(y_i) \sin(|x_i| \frac{1}{2})$ | $[-500,500]$ |

function, we multiply the argument $x$ by an orthogonal rotation matrix $M$ to obtain a new argument $z$ for the rotated function. This rotation matrix was obtained by using Salomon’s method [40]. So, we can define the first 5 functions in this group

$$f_n(x) = f_{n-6}(z), \quad \forall n = 9, \ldots, 13,$$  
(A1)

with $z = Mx$. The last function $f_{14}$ is defined as follows:

$$f_{14}(x) = f_8(z)$$  
(A2)

with

$$z_i = \begin{cases} y_i \sin \left( \frac{|y_i|}{2} \right), & |y_i| \leq 500 \\ 0.001(|y_i| - 500)^2, & |y_i| > 500 \end{cases}, \quad \forall i = 1, \ldots, D$$  
(A3)

$$y = y' + 420.96$$  
(A4)

$$y' = M(x - 420.96).$$  
(A5)

This condition is necessary to keep the global optimum of the original Schwefel’s function, which is located at $[420.96, 420.96, \ldots, 420.96]$, within the search range after rotation.

Acknowledgment

This research was supported by the High-Performance Computing Center at UFRN (NPAD/UFRN).
References

[1] Pan, W.-T.: A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowledge-Based Systems 26, 69–74 (2012)

[2] Das, S., Biswas, S., Panigrahi, B.K., Kundu, S., Basu, D.: A spatially informative optic flow model of bee colony with saccadic flight strategy for global optimization. IEEE Transactions on Cybernetics 44(10), 1884–1897 (2014) https://doi.org/10.1109/TCYB.2014.2298916

[3] Mahdavi, S., Shiri, M.E., Rahnamayan, S.: Metaheuristics in large-scale global continues optimization: A survey. Information Sciences 295, 407–428 (2015)

[4] Mirjalili, S.: The ant lion optimizer. Advances in Engineering Software 83, 80–98 (2015)

[5] Zhang, G., Li, Y.: A memetic algorithm for global optimization of multimodal nonseparable problems. IEEE Transactions on Cybernetics 46(6), 1375–1387 (2016) https://doi.org/10.1109/TCYB.2015.2447574

[6] Zhang, Y.F., Chiang, H.D.: A novel consensus-based particle swarm optimization-assisted trust-tech methodology for large-scale global optimization. IEEE Transactions on Cybernetics 47(9), 2717–2729 (2017) https://doi.org/10.1109/TCYB.2016.2577587

[7] Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y.: Metaheuristic research: a comprehensive survey. Artificial Intelligence Review 52(4), 2191–2233 (2019)

[8] Halim, A.H., Ismail, I., Das, S.: Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. Artificial Intelligence Review 54(3), 2323–2409 (2021)

[9] Ribeiro, L.T.M., Santos, M., Moreira, M.Á.L., Araújo Costa, I.P., Araújo Costa, A.P., Gomes, C.F.S.: Bibliometric study on the heuristics simulated annealing. In: Ullah, A., Anwar, S., Calandra, D., Di Fuccio, R. (eds.) Proceedings of International Conference on Information Technology and Applications, pp. 411–422. Springer, Singapore (2024)

[10] Juan, A.A., Keenan, P., Martí, R., McGarraghy, S., Panadero, J., Carroll, P., Oliva, D.: A review of the role of heuristics in stochastic optimisation: From metaheuristics to learnheuristics. Annals of Operations Research 320(2), 831–861 (2023)

[11] Ding, Z., Liu, J., Sun, Y., Jiang, C., Zhou, M.: A transaction and qos-aware service selection approach based on genetic algorithm. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45(7), 1035–1046 (2015) https://doi.org/10.1109/TSMC.2015.2396001
[12] Mirjalili, S., Lewis, A.: The whale optimization algorithm. Advances in Engineering Software 95, 51–67 (2016)

[13] Mistry, K., Zhang, L., Neoh, S.C., Lim, C.P., Fielding, B.: A microga embedded pso feature selection approach to intelligent facial emotion recognition. IEEE Transactions on Cybernetics 47(6), 1496–1509 (2017) https://doi.org/10.1109/TCYB.2016.2549639

[14] Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A.J., Molina, D., LaTorre, A., Suganthan, P.N., Coello, C.A.C., Herrera, F.: A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. Swarm and Evolutionary Computation 64, 100888 (2021)

[15] Martí, R., Sevaux, M., Sörensen, K.: Fifty years of metaheuristics. European journal of operational research.-Amsterdam 321(2), 345–362 (2025)

[16] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. SCIENCE 220(4598), 671–680 (1983)

[17] Eshelman, L.J., Schaffer, J.D.: Real coded genetic algorithms and intervalschemata. Foundations of genetic algorithms 2, 187–202 (1993)

[18] Wright, A.H., et al.: Genetic algorithms for real parameter optimization. Foundations of genetic algorithms 1, 205–218 (1991)

[19] Kenny, J.: Particle swarm optimization. In: Proc. 1995 IEEE Int. Conf. Neural Networks, pp. 1942–1948 (1995)

[20] Xavier-De-Souza, S., Suykens, J.A.K., Vandewalle, J., Bollé, D.: Coupled simulated annealing. Trans. Sys. Man Cyber. Part B 40(2), 320–335 (2010) https://doi.org/10.1109/TSMCB.2009.2020435

[21] Yang, X.-S., Deb, S.: Cuckoo search via levy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress On, pp. 210–214 (2009). IEEE

[22] Holland, J.H.: Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT press, ??? (1992)

[23] Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4), 341–359 (1997)

[24] Suykens, J.A., Vandewalle, J., De Moor, B.: Intelligence and cooperative search by coupled local minimizers. International Journal of Bifurcation and Chaos 11(08), 2133–2144 (2001)
[25] Szu, H., Hartley, R.: Fast simulated annealing. Physics letters A 122(3-4), 157–162 (1987)

[26] Civicioglu, P., Besdok, E.: A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artificial intelligence review, 1–32 (2013)

[27] Gálvez, A., Iglesias, A., Cabellos, L.: Cuckoo search with lévy flights for weighted bayesian energy functional optimization in global-support curve data fitting. The Scientific World Journal 2014 (2014)

[28] Song, Y., Cai, X., Zhou, X., Zhang, B., Chen, H., Li, Y., Deng, W., Deng, W.: Dynamic hybrid mechanism-based differential evolution algorithm and its application. Expert Systems with Applications 213, 118834 (2023)

[29] Piotrowski, A.P., Napiorowski, J.J., Piotrowska, A.E.: Particle swarm optimization or differential evolution—a comparison. Engineering Applications of Artificial Intelligence 121, 106008 (2023)

[30] Kuhlemann, S., Tierney, K.: A genetic algorithm for finding realistic sea routes considering the weather. Journal of Heuristics 26(6), 801–825 (2020)

[31] Sarmady, S.: An investigation on genetic algorithm parameters. School of Computer Science, Universiti Sains Malaysia (2007)

[32] Yang, X.-S., Deb, S.: Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation 1(4), 330–343 (2010)

[33] Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33(1-2), 61–106 (2010)

[34] Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE transactions on evolutionary computation 15(1), 4–31 (2011)

[35] Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Applied mathematics and computation 214(1), 108–132 (2009)

[36] Herrera, F., Lozano, M., Sánchez, A.M.: A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. International Journal of Intelligent Systems 18(3), 309–338 (2003)

[37] Takahashi, M., Kita, H.: A crossover operator using independent component analysis for real-coded genetic algorithms. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress On, vol. 1, pp. 643–649 (2001). IEEE

[38] Tu, Z., Lu, Y.: A robust stochastic genetic algorithm (stga) for global numerical optimization. IEEE Transactions on Evolutionary Computation 8(5), 456–470
(2004)

[39] Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transactions on Evolutionary computation 3(2), 82–102 (1999)

[40] Salomon, R.: Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems 39(3), 263–278 (1996)
