



The Walking Palm Tree algorithm: A new Metaheuristic Algorithm for Solving Optimization Problems

Farouq Zitouni^{1,*}, Saad Harous², Seyedali Mirjalili³, Abdelhai Mohamed Bouaicha¹, Hocine Abdellatif Houari¹, Ali Wagdy Mohamed⁴, Abdelhadi Limane¹, Rihab Lakkbichi¹, and Aridj Ferhat¹

¹ Department of Computer Science and Information Technologies, Kasdi Merbah University, Ouargla, Algeria

{zitouni.farouq,limane.abdelhadi,lakkbichi.rihab,ferhat.aridj}@univ-ouargla.dz, mohamedabdelhai.bouaicha@unisalento.it, h.hocine.abdellatif@gmail.com

² College of Computing and Informatics, Department of Computer Science, University of Sharjah, Sharjah, United Arab Emirates. harous@sharjah.ac.ae

³ Centre for Artificial Intelligence Research and Optimization, Torrens University, Australia. ali.mirjalili@gmail.com

⁴ Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza, Egypt. aliwagdy@gmail.com

* Corresponding author

Abstract. We introduce the Walking Palm Tree (WPT) optimizer, a novel metaheuristic optimization algorithm inspired by the movement pattern of the *Socratea Exorrhiza* palm tree species. This species adapts to sunlight by growing roots towards the desired direction while allowing older roots to fade away. We provide a detailed mathematical framework for the algorithm and demonstrate its effectiveness across a diverse range of search problems. By conducting experiments on 31 benchmark test functions, which encompass various characteristics such as unimodal, multimodal, hybrid, and composite functions, we validate the efficacy of WPT. Our statistical analyses, including Friedman ranking and Wilcoxon signed-rank tests, further confirm the competitiveness of WPT in tackling complex optimization challenges compared to established algorithms.

Keywords: Global Optimization · Metaheuristics · Nature-Inspired Optimizers · Walking Palm Trees · Unconstrained Optimization Problems.

1 Introduction

Optimization methods strive to enhance a designated function f with n ($n \geq 1$) decision variables, depending on multiple constraints. Practical optimization problems generally involve a vast space of feasible solutions, often with non-linear constraints, making the selection of the best solution expensive and time-consuming [25]. Creating algorithms to tackle such challenges poses significant

© The Author(s) 2024

C. A. Kerrache et al. (eds.), *Proceedings of the International Conference on Emerging Intelligent Systems for Sustainable Development (ICEIS 2024)*, Advances in Intelligent Systems Research 184,

https://doi.org/10.2991/978-94-6463-496-9_3

difficulty, especially when dealing with non-convex functions featuring multiple local optima. As a solution, various metaheuristics have been proposed to address diverse engineering problems [13,57]. Metaheuristics demonstrate effectiveness in addressing intricate optimization problems due to their computational affordability, adaptability, and straightforward implementation [63]. However, their stochastic nature means there is no guarantee of solution optimality [60,61]. Nevertheless, metaheuristics have become increasingly popular over the past two decades because of their straightforward design, lack of reliance on gradients, and minimal need for parameter adjustments.

Two categories of metaheuristics are identified: those based on individual behaviors and those based on population dynamics [13]. In the former category, a solitary random solution (referred to as an individual) is generated and refined over a set number of iterations, resulting in a nearly optimal solution by the conclusion of the process. Prominent instances encompass the simulated annealing optimization technique [33]. In the latter family, multiple random solutions (i.e., a population) are generated and evolved over a specified number of generations to enhance their positions. The best solution at the conclusion of the process is selected. Examples in this category include ant colony optimization [15], genetic algorithms [23], and particle swarm optimization [16]. In practice, population-based approaches tend to outperform individual-based algorithms.

Population-based algorithms have attracted more interest compared to individual-based approaches [14]. They can be categorized into four types [65]:

1. Evolutionary-based algorithms, governed by Darwinian principles such as reproduction and mutation, e.g., genetic algorithms [23];
2. Universe-based algorithms, inspired by the laws of physics and chemistry like electromagnetism and gravitation, e.g., the solar system algorithm [66];
3. Human-based algorithms, which emulate relationships within human societies, e.g., the gaining sharing knowledge based algorithm [41];
4. Swarm-intelligence-based algorithms, mimicking social behaviors of insects or animals, e.g., the particle swarm optimization [32].

Metaheuristics employ two key operators: exploration and exploitation [62]. In the exploration phase, the algorithm searches for promising areas within the search space [36], while during exploitation, the optimizer seeks the best solution within a specific promising area [36]. Striking a balance between these phases is crucial for any algorithm to discover satisfactory solutions [62]. The extensive variety of metaheuristics in the literature prompts the question: why are there so many algorithms? The answer lies in the No-Free-Lunch theorem [22], which asserts that a single algorithm cannot efficiently solve all optimization problems due to their inherent complexity and diverse features. Hence, new metaheuristics are continually proposed to address specific problems.

We introduce the WPT algorithm, a novel, nature-inspired, population-based, and gradient-free approach for global optimization. Drawing inspiration from the walking behavior of the *Socratea Exorrhiza* palm tree, the WPT algorithm makes the following key contributions:

- It features two control parameters, namely neighborhood size and reproduction probability, ensuring ease of implementation.
- The WPT algorithm exhibits swarming behavior akin to the firefly algorithm [64]. Notably, while the firefly algorithm excels in exploration, its exploitation is weak [67]. In the WPT algorithm, individuals are organized into clusters, enhancing exploitation.
- It employs two global optima instead of one to guide the swarming behavior of individuals, aiming to strengthen the avoidance of local optima. Besides, the algorithm demonstrates robustness and efficiency.
- Performance evaluation involves testing 31 benchmark functions and comparing results with influential and well-known optimization algorithms.

The remainder of the paper is organized as follows: Section 2 reviews well-known nature-inspired metaheuristics. Section 3 details the behavior of walking palm trees in nature and introduces the theoretical foundations, mathematical formulation, and pseudocode of the proposed WPT algorithm. Section 4 assesses the performance and efficiency of the WPT algorithm. The concluding section provides a summary and outlines directions for future work.

2 Related Work

We discuss some related state-of-the-art optimizers using the classification proposed in [65].

2.1 Evolutionary-based algorithms

Evolutionary algorithms, which are inspired by biological evolution, entail a population of potential solutions that evolve iteratively to improve their fitness. Genetic algorithms [53], rooted in Darwinian principles, are commonly utilized for optimization tasks. The red deer algorithm [17] emulates the competitive behavior of male red deer vying to attract mates. Differential evolution [48] involves perturbing candidate solutions using calculated differences among randomly selected ones. Biogeography-based optimization [55] draws insights from concepts in biogeography. The coral reefs algorithm [52] mimics the reproductive and formative processes of coral reefs. The red fox algorithm [47] takes inspiration from the behaviors of red foxes.

2.2 Universe-based algorithms

Algorithms inspired by the universe encompass mathematical equations, physical/chemical laws, and cosmic phenomena. The weighted mean of vectors [3] relies on stable structures to update vector positions. The plasma generation algorithm [34] replicates the process of generating plasma. The vibrating particle system [28] is based on the free vibration of single degree of freedom systems. Thermal exchange optimization [27] utilizes Newton's law of cooling. The vertex

separation problem algorithm [10] linearly arranges graphs' vertices while minimizing the number of separators. The momentum search algorithm [12] follows the law of momentum conservation. The Archimedes optimization algorithm [20] simulates the behavior of objects immersing based on Archimedes' principles. The supernova optimizer algorithm [24] mirrors the natural phenomena of luminous stellar explosions. The arithmetic optimization algorithm [1] draws inspiration from the behavior of arithmetic operators in mathematics. The atomic orbital search [7] adopts principles from quantum mechanics. The material generation algorithm [56] is based on the process of generating new materials. The solar system algorithm [66] is inspired by the orbital movements of objects in the solar system.

2.3 Human-based algorithms

The study of human societies, with their diverse cultures, community dynamics, and enduring traditions, has long been a subject of interest. Many metaheuristic algorithms draw inspiration from these facets. The dynastic optimization algorithm [58] is influenced by the evolutionary processes observed in human societies. The artificial coronary circulation system [30] replicates the growth of coronary arteries in the human heart. The Ali Baba and the forty thieves' algorithm [8] derives its principles from the tale of Ali Baba and the forty thieves. The past present future algorithm [26] mimics how individuals learn from their experiences or from successful figures in society. The farmland fertility algorithm [54] divides farmland into sections based on soil quality. The Ebola optimization search [46] takes inspiration from the Ebola virus disease and its spread within human societies. The political optimizer [6] imitates politicians' strategies for adjusting their political positions. The search-in-forest optimizer [4] is based on the organized behavior of search teams looking for missing persons in a forest. The shuffled shepherd optimization [31] method emulates the intelligence and strategies of shepherds when leading herds of animals for pasturing.

2.4 Swarm-intelligence-based algorithms

We present several innovative swarm-intelligence-based algorithms. The artificial butterfly optimization [49] draws inspiration from the strategies of butterfly species in competing for sunspots. The salp swarm algorithm [38] imitates the swarming behavior of salps as they navigate the ocean and search for food. The snake optimizer [21] simulates the mating behavior of snakes. The chameleon swarm algorithm [9] replicates the dynamic behavior of chameleons while they navigate and hunt for food. The earthworm optimization algorithm [59] mimics the reproduction mechanism of earthworms. The emperor penguins' colony [19] simulates the movement patterns of penguins to retain heat and safeguard their eggs from cold. The Namib beetle optimization [11] algorithm imitates the behavior of Namibian beetles during water collection. The archerfish hunting optimizer [65] draws inspiration from the shooting and jumping behaviors of archerfish when catching insects. The lion pride optimization algorithm [29]

replicates the natural collective behavior of lions within their social groups. The golden eagle optimizer [42] is derived from the intelligence of golden eagles in adjusting their speed to effectively hunt prey.

3 Proposed Algorithm

3.1 Biological inspiration

There exists a captivating species of palm tree with a remarkable capability – it can move and relocate! Scientifically termed *Socratea Exorrhiza*, it is commonly known as the walking palm tree. Figure 1 showcases its distinctive appearance. The *Socratea Exorrhiza* thrives in the tropical rainforests of Central and South America. It boasts long and sturdy roots that extend outward from its base, resembling multiple legs. This palm tree maneuvers from shaded to sunny areas by guiding the growth of its roots in the desired direction. As it progresses, old roots are lifted into the air and eventually decay. The walking palm tree elegantly advances in the direction of the newly formed roots.



Fig. 1: The walking palm tree.

3.2 Mathematical definition

We introduce a novel metaheuristic algorithm for global optimization, inspired by the behavior of walking palm trees, termed the Walking Palm Tree (WPT) algorithm. Considering a search space of dimension d and an optimization function $f(\cdot)$ to be minimized. Our algorithm defines a set T comprising n palm

trees ($T = \{t_1, \dots, t_n\}$, with each $t_i \in T$ represented by its location in the search space: $X_{\langle t_i \rangle} = (x_1, \dots, x_d)$). Moreover, each palm tree $t_i \in T$ possesses a set R_{t_i} of m roots ($R_{t_i} = \{r_1, \dots, r_m\}$). The roots of a given palm tree must be situated within its vicinity. The position of a root r_j attached to a palm tree t_i is denoted by $X_{\langle r_j, t_i \rangle} = (x_1, \dots, x_d)$, and it must adhere to Equation 1.

$$\forall t_i \in T, \forall r_j \in R_{t_i}, \exists \epsilon \in \mathbb{R}^+ : d(X_{\langle t_i \rangle}, X_{\langle r_j, t_i \rangle}) \leq \epsilon \quad (1)$$

The expression $d(X_{\langle t_i \rangle}, X_{\langle r_j, t_i \rangle})$ denotes the distance between positions $X_{\langle t_i \rangle}$ and $X_{\langle r_j, t_i \rangle}$, which could be measured using metrics like Euclidean, Manhattan, or Minkowski distance. Following this, the subsequent sections comprehensively outline the WPT algorithm, providing a step-by-step delineation along with detailed explanations of all essential concepts and equations at each stage.

Initialization of the first population A starting group of potential solutions is scattered randomly throughout the exploration area. The size of this group is determined to be $n(m+1)$. At the start of iteration $t = 0$, n positions are randomly chosen within the exploration space using Equation 2 for the trees' positions. Following this, for each tree position generated, m positions for the roots are randomly initialized using Equation 3.

$$X_{\langle t_i \rangle}^{(0)} = \mathbb{U} \circ ((U_1 - L_1) + L_1, \dots, (U_d - L_d) + L_d) \quad (2)$$

$$X_{\langle r_j, t_i \rangle}^{(0)} = X_{\langle t_i \rangle}^{(0)} + \epsilon(2\mathbb{U} - \mathbf{1}_{1 \times d}) \quad (3)$$

In this context, \mathbb{U} and \mathbb{U} denote two $1 \times d$ random matrices drawn from a uniform distribution within the range $[0, 1]$. The variables L and U indicate the lower and upper boundaries of decision variables, respectively. ϵ signifies the neighborhood size where the roots of each palm tree are situated. The initialization phase, outlined in lines 1 to 6 of Algorithm 1, organizes individuals into n clusters, each represented by centroids at points $X_{\langle t_1 \rangle}, \dots, X_{\langle t_n \rangle}$. Each cluster comprises $m+1$ locations. To depict these clusters, we utilize n sequences⁵: $S_{\langle 1 \rangle}, \dots, S_{\langle n \rangle}$. The initial element of a particular sequence $S_{\langle i \rangle}$ corresponds to the location $X_{\langle t_i \rangle}$, followed by subsequent elements representing the locations $X_{\langle r_j, t_i \rangle}$. The position of any element within a sequence $S_{\langle i \rangle}$ is determined by its index, denoted as $S_{\langle i, j \rangle}$.

Optimization process of the WPT algorithm The swarm members kick-start the swarming process, and the WPT algorithm proceeds through iterations until a designated stopping criterion is fulfilled, often achieving a particular accuracy or reaching a predetermined maximum iteration count. The optimization pattern of the WPT algorithm encompasses cyclic repetition of the following three stages. It is crucial throughout the optimization journey to guarantee that

⁵ In mathematics, a sequence is an ordered collection of elements where repetitions are allowed and the order matters.

the elements within sequences $S_{\langle 1 \rangle}, \dots, S_{\langle n \rangle}$ consistently comply with both Equation 1 and the defined boundary constraints.

Exploration of the search space In a manner akin to a palm tree shifting from shade to sunlight by extending roots in the preferred direction while the old roots wither away, we adopt a similar concept to illustrate the movement of individuals within the WPT algorithm. The intricacies of the exploration phase are expounded upon in the following paragraphs and are transcribed into lines 8 to 26 of Algorithm 1.

1. Step 1: For every sequence $S_{\langle i \rangle}$ ($i \in 1, \dots, n$), we determine the best sequence (referred to as $S_{\langle \mathbf{fb} \rangle}$) and the second-best sequence (referred to as $S_{\langle \mathbf{sb} \rangle}$) within the current population. The indices \mathbf{fb} and \mathbf{sb} are calculated using Equations 4 and 5, respectively. Let $S_{\langle i,1 \rangle}$, $S_{\langle \mathbf{fb},1 \rangle}$, and $S_{\langle \mathbf{sb},1 \rangle}$ denote the first elements of sequences $S_{\langle i \rangle}$, $S_{\langle \mathbf{fb} \rangle}$, and $S_{\langle \mathbf{sb} \rangle}$, respectively. Then, a triangle is constructed with vertices at $S_{\langle i,1 \rangle}$, $S_{\langle \mathbf{fb},1 \rangle}$, and $S_{\langle \mathbf{sb},1 \rangle}$, and its area is computed using Heron’s formula [44]. This resulting area is labeled as A .

$$\mathbf{fb} = \underset{i \in \{1, \dots, n\}}{\operatorname{argmin}} \{f(S_{\langle i,1 \rangle})\} \tag{4}$$

$$\mathbf{sb} = \underset{i \in \{1, \dots, n\} \setminus \{\mathbf{fb}\}}{\operatorname{argmin}} \{f(S_{\langle i,1 \rangle})\} \tag{5}$$

2. Step 2: We assess each element $S_{\langle i,j \rangle} \in S_{\langle i \rangle}$ ($j \in 2, \dots, m+1$) to decide whether it should be kept or discarded (similar to the shedding of roots). This decision involves computing the areas of three triangles: $\Delta S_{\langle i,1 \rangle} S_{\langle \mathbf{fb},1 \rangle} S_{\langle i,j \rangle}$, $\Delta S_{\langle i,1 \rangle} S_{\langle \mathbf{sb},1 \rangle} S_{\langle i,j \rangle}$, and $\Delta S_{\langle \mathbf{fb},1 \rangle} S_{\langle \mathbf{sb},1 \rangle} S_{\langle i,j \rangle}$. These areas are denoted as A_1 , A_2 , and A_3 , respectively. If the quantity $|A - (A_1 + A_2 + A_3)|$ exceeds a predefined threshold (denoted as τ), the element $S_{\langle i,j \rangle}$ is removed from the sequence $S_{\langle i \rangle}$. The threshold τ is estimated using Equation 6.

$$\tau = e^{\mu_0} \times d^{\mu_1} \times \epsilon^{\mu_2} \tag{6}$$

The values $\mu_0 = -1.6$, $\mu_1 = 1.08$, and $\mu_2 = 1.00$ were empirically determined through the multiplicative model of multiple linear regression [5]. Here, e represents Euler’s number, d signifies the dimension of the search space, and ϵ denotes the neighborhood size.

3. Step 3: We elucidate the swarming behavior of the locations $X = (x_1, \dots, x_d)$ that persist within each sequence $S_{\langle i \rangle}$. To achieve this objective, we examine the first-best location $S_{\langle \mathbf{fb},1 \rangle} = (y_1, \dots, y_d)$ and the second-best location $S_{\langle \mathbf{sb},1 \rangle} = (z_1, \dots, z_d)$, respectively. Then, we calculate the location T using Equation 7. Finally, the elements remaining in each sequence $S_{\langle i \rangle}$ are updated through Equation 8. The first-best and second-best sequences collaboratively contribute to updating the locations of candidate solutions in each sequence and thereby to preventing premature convergence.

$$T = \Phi \circ (y_1, \dots, y_d) + \hat{\Phi} \circ (z_1, \dots, z_d) \tag{7}$$

$$\begin{aligned}
\Theta &= \left(\tan^{-1} \left(\frac{\Delta_1}{|x_1 - y_1|} \right), \dots, \tan^{-1} \left(\frac{\Delta_1}{|x_d - y_d|} \right) \right) \\
\acute{\Theta} &= \left(\tan^{-1} \left(\frac{\Delta_2}{|x_1 - z_1|} \right), \dots, \tan^{-1} \left(\frac{\Delta_2}{|x_d - z_d|} \right) \right) \\
\Delta_1 &= |f(X) - f(S_{\langle \mathbf{fb}, 1 \rangle})|, \quad \Delta_2 = |f(X) - f(S_{\langle \mathbf{sb}, 1 \rangle})| \\
\Phi &= \left(\frac{\min(\theta_1, \acute{\theta}_1)}{\theta_1 + \acute{\theta}_1}, \dots, \frac{\min(\theta_d, \acute{\theta}_d)}{\theta_d + \acute{\theta}_d} \right) \\
\acute{\Phi} &= \left(\frac{\max(\theta_1, \acute{\theta}_1)}{\theta_1 + \acute{\theta}_1}, \dots, \frac{\max(\theta_d, \acute{\theta}_d)}{\theta_d + \acute{\theta}_d} \right) \\
X^{(t)} &= X^{(t-1)} + t_1 t_2 (T - X^{(t-1)}) + \alpha G \tag{8} \\
t_1 &= 1 - e^{\beta t^2}, \quad \beta = \frac{\ln 0.5}{\left(\frac{\Delta}{2}\right)^2} \\
t_2 &= \left(1 - \frac{t}{\Delta}\right)^\gamma (\sigma_1 - \sigma_2) + \sigma_2, \quad \gamma = \frac{\ln \left(\frac{0.5 - \sigma_2}{\sigma_1 - \sigma_2}\right)}{\ln 0.5}
\end{aligned}$$

In this context, the symbol \circ signifies the Hadamard product of two vectors; α represents a randomization parameter drawn from a uniform distribution; G denotes a vector of random numbers drawn from a Gaussian distribution; Δ indicates the total number of iterations; σ_1 denotes the initial value desired at iteration $t = 1$; and σ_2 represents the final desired value at iteration $t = \Delta$. The term t_1 serves as a scale-dependent parameter influencing the design of light intensity, while t_2 acts as a scale-dependent parameter governing the speed of displacement of palm trees.

4. Step 4: To refill each sequence $S_{\langle i \rangle}$ with fresh candidate solutions $S_{\langle i, j \rangle}$ (i.e., new roots) to substitute the ones removed in step 2, Equation 3 is utilized for this task.

Exploitation of the search space Mirroring the natural behavior of walking palm trees, some candidate solutions perish and are replaced with new ones throughout the optimization process. To emulate this occurrence, we adopt a straightforward strategy. For each sequence $S_{\langle i \rangle}$ ($i \in 1, \dots, n$), we pinpoint the index (referred to as ω) of the element with the minimum fitness value using Equation 9. Subsequently, we swap the elements $S_{\langle i, 1 \rangle}$ and $S_{\langle i, \omega \rangle}$. Then, for every element $S_{\langle i, j \rangle} \in S_{\langle i \rangle}$ ($j \in 2, \dots, m+1$), we evaluate whether it satisfies Equation 1. If this condition is not fulfilled, the element $S_{\langle i, j \rangle}$ is substituted with a new one generated using Equation 3. The exploitation phase is executed in lines 27 to 35 of Algorithm 1.

$$\omega = \underset{j \in \{1, \dots, m+1\}}{\operatorname{argmin}} \{f(S_{\langle i, j \rangle})\} \tag{9}$$

Local optimum avoidance Much like walking palm trees dispersing seeds for new offspring, the generated seeds are randomly spread throughout the environment in the WPT algorithm. Each walking palm tree in the algorithm is represented by a sequence. At each iteration t , we assess whether the reproduction process should occur, determined by a specific probability ρ . If this condition is satisfied, a sequence is chosen arbitrarily using the roulette wheel selection algorithm [35]. The chosen sequence is then eliminated from the current population, replaced with a new one, and included in the population under consideration. This step introduces a vital aspect to the WPT algorithm, preventing it from becoming stuck in local optima. Unlike the conventional approach in evolutionary algorithms, where feasible individuals with superior fitness values take precedence over infeasible ones, the WPT algorithm acknowledges the probabilistic and iterative nature of evolutionary methods. It acknowledges that, during the walking process, some infeasible sequences may offer more valuable insights than feasible ones. The phase of avoiding local optima is executed in lines 36 to 43 of Algorithm 1.

The roulette wheel selection algorithm (RWSA) [35] is employed to determine the sequence to be removed from the current population. Firstly, a roulette wheel is constructed based on the relative fitness values of each sequence $S_{\langle i \rangle}$ ($i \in 1, \dots, n$) using Equation 10. Secondly, a random number between 0 and 1 (denoted as S) is generated. Thirdly, starting from the top of the population, we incrementally add the relative fitness values P_i of sequences $S_{\langle i \rangle}$ to a partial sum (denoted as P) until $P < S$. Finally, the sequence $S_{\langle p \rangle}$ for which P surpasses S is the one selected for replacement.

$$P_i = \frac{f(S_{\langle i,1 \rangle})}{\sum_{j=1}^n f(S_{\langle j,1 \rangle})} \quad (10)$$

Upon selecting the sequence $S_{\langle p \rangle}$ for replacement, its initial element (i.e., $S_{\langle p,1 \rangle}$) is modified using Equation 11. Subsequently, the remaining elements (i.e., $S_{\langle p,j \rangle}$, $j \in 2, \dots, m+1$) are generated employing Equation 3. The second component in Equation 11 is a row matrix of size d drawn from the Lévy distribution [45].

$$X^{(t)} = X^{(t-1)} + \left(\frac{u_1}{|v_1|^{\frac{1}{\beta}}}, \dots, \frac{u_d}{|v_d|^{\frac{1}{\beta}}} \right) \quad (11)$$

$$u_i \sim N(0, \sigma_u^2), \sigma_u = \left(\frac{\Gamma(1 + \beta) \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}$$

$$v_i \sim N(0, \sigma_v^2), \sigma_v = 1$$

In this context, u_i and v_i denote two random numbers sampled from the normal distribution [18]; β stands for the power law index ($1 < \beta < 2$); and Γ represents the Gamma function.

3.3 Pseudocode and temporal complexity

Algorithm 1 outlines the steps of the WPT algorithm, with computational complexity primarily dependent on two phases: initialization and updating of candidate solutions' locations. The first phase has a computational complexity of $T_1 = O(n \times m)$. The complexity of the second phase is $T_2 = O(\Delta \times n \times m + \Delta \times n \times m + \Delta \times m)$. Consequently, the overall computational complexity of the WPT algorithm is $O(T_1 + T_2)$. Since $O(\alpha + \beta)$ can be simplified to $O(\max(\alpha, \beta))$, the computational complexity of the WPT algorithm can be streamlined to $O(\Delta \times n \times m) \sim O(n^3)$.

4 Implementation and Experiments

To evaluate the performance of the WPT algorithm, we selected 31 benchmark functions from the literature [2]⁶⁷⁸. These functions are categorized as unimodal (UF), multimodal (MF), hybrid (HF), and composition (CF) functions. UFs, characterized by a single global minimum, assess exploitation capability, with seven UFs utilized in our study. MFs, featuring multiple global or local minima, evaluate both exploration capability and the ability to avoid local minima. We employ 16 MFs in our study, of which ten are fixed-dimension functions (FDFs). HFs and CFs assess the balance between exploitation and exploration phases and the ability to escape local minima during the optimization process. In our study, we use eight functions, including two HFs and six CFs. Due to space constraints, detailed information about these functions is not provided in this paper.

The performance of the WPT algorithm is compared with twelve metaheuristics, namely BBO [55], PSO [55], DE [55], GSA [51], TLBO [50], SSA [38], GWO [40], WOA [39], MFO [37], FFA [54], IPO [43], and AHO [65]. TLBO, SSA, GWO, WOA, MFO, FFA, IPO, and AHO are selected as influential and recent optimization algorithms, while BBO, PSO, DE, and GSA serve as reference optimization optimizers due to their frequent use in the context of metaheuristics. The WPT algorithm is implemented in Java on a laptop with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor, 16.0 GB RAM, and running Windows 11 Familial 64-bit. The population size is 30 palm trees, each with 30 roots, and 500 generations are used for each run. Average (AVG) and standard deviation (STD) values are calculated over 30 independent runs. Metaheuristic settings for the comparative study are summarized in Table 1, with values extracted from the respective papers.

We conducted a total of 30 independent runs for each optimizer, applying them to assess performance across the 31 test functions. For each configuration, we recorded the mean and standard deviation values. These statistical measures are intended for use in comprehensive statistical analyses to compare the algorithms' performance. Unfortunately, due to the inherent limitation of page

⁶ <https://www.sfu.ca/~ssurjano/optimization.html>

⁷ <https://al-roomi.org/benchmarks/unconstrained/n-dimensions>

⁸ <https://github.com/P-N-Suganthan/CEC2014>

Algorithm 1: Pseudocode of the WPT algorithm.

Input: d : The search space dimension; $[L_1, U_1], \dots, [L_d, U_d]$: The domains of decision variables; f : The objective function aiming for minimization; n : The count of trees; m : The count of roots; $S_{(1)}, \dots, S_{(n)}$: A set of n sequences; ϵ : The size of the neighborhood; Δ : The maximum number of iterations; and ρ : The probability of reproduction.

```

1 for  $i \leftarrow 1$  to  $n$  do
2   Compute  $S_{(i,1)}$  using Equation 2;
3   for  $j \leftarrow 1$  to  $m$  do
4     Compute  $S_{(i,j+1)}$  using Equation 3;
5   end
6 end
7 for  $t \leftarrow 1$  to  $\Delta$  do
8   for  $i \leftarrow 1$  to  $n$  do
9     Compute  $\mathbf{fb}$  using Equation 4;
10    Compute  $\mathbf{sb}$  using Equation 5;
11    if  $i \notin \{\mathbf{fb}, \mathbf{sb}\}$  then
12      Compute  $A(\Delta S_{(i,1)} S_{(\mathbf{fb},1)} S_{(\mathbf{sb},1)})$ ;
13      Update  $S_{(i,1)}$  using Equation 8;
14      for  $j \leftarrow 2$  to  $m+1$  do
15        Compute  $A_1(\Delta S_{(i,1)} S_{(\mathbf{fb},1)} S_{(i,j)})$ ;
16        Compute  $A_2(\Delta S_{(i,1)} S_{(\mathbf{sb},1)} S_{(i,j)})$ ;
17        Compute  $A_3(\Delta S_{(\mathbf{fb},1)} S_{(\mathbf{sb},1)} S_{(i,j)})$ ;
18        if  $|A - (A_1 + A_2 + A_3)| > \tau$  then
19          Replace  $S_{(i,j)}$  using Equation 3;
20        end
21      else
22        Update  $S_{(i,j)}$  using Equation 8;
23      end
24    end
25  end
26 end
27 for  $i \leftarrow 1$  to  $n$  do
28   Compute  $\omega$  using Equation 9;
29   Swap  $S_{(i,1)}$  and  $S_{(i,\omega)}$ ;
30   for  $j \leftarrow 2$  to  $m+1$  do
31     if  $d(S_{(i,1)}, S_{(i,j)}) > \epsilon$  then
32       Replace  $S_{(i,j)}$  using Equation 3;
33     end
34   end
35 end
36  $\rho_0 \leftarrow$  Generate a random number between 0 and 1;
37 if  $\rho_0 \leq \rho$  then
38   Compute  $p$  using the RWSA;
39   Update  $S_{(p,1)}$  using Equation 11;
40   for  $j \leftarrow 2$  to  $m+1$  do
41     Replace  $S_{(p,j)}$  using Equation 3;
42   end
43 end

```

Table 1: Parameters' values of the algorithms used for the comparative study.

Algorithm	Parameter	Value
WPT	ϵ	10^{-4}
	σ_1	1
	σ_2	10^{-6}
	ρ	10^{-6}
BBO	Habitat modification probability	1
	Immigration probability limits	[0, 1]
	Step size	1
	$E I$	1
PSO	Mutation probability	0.005
	Inertia factor	0.3
DE	c_1	1
	c_2	1
GSA	Scaling factor	0.5
	Crossover probability	0.5
TLBO	α	20
	G_0	100
	Power of R	$N \rightarrow 1$
	k	1
SSA	T	[1, 2]
	c_1	$2 \times e^{-\left(\frac{t}{L}\right)^2}$
	c_2	random
GWO	c_3	random
	Convergence constant a	$2 \rightarrow 0$
WOA	Convergence constant a	$2 \rightarrow 0$
	Spiral factor b	1
MFO	Convergence constant a	$-1 \rightarrow 0$
	Spiral factor b	1
FFA	K	2
	α	0.6
	β	0.4
	W	1
	Q	0.7
IPO	c_1	0.685
	c_2	0.653
	$shift_1$	527.392
	$shift_2$	380.811
	$scale_1$	0.423
AHO	$scale_2$	0.571
	θ	0.27
	ω	0.01

space in this paper, we regretfully acknowledge that it is not feasible to include either the mathematical definition of the different test functions or the extensive compilation of mean and standard deviation results for each algorithm and test function combination.

To conduct a thorough analysis and gain a deeper understanding of the algorithms' performance, we subjected the acquired standard deviation values to the Friedman post hoc Dunn's test. This statistical analysis facilitated a comprehensive examination of the variability displayed by the algorithms, providing nuanced insights into their performance characteristics. Through the application of the Friedman post hoc Dunn's test, our objective was to uncover any significant differences in performance and highlight the algorithms that demonstrated higher levels of consistency and reliability. The significance level for the Friedman post hoc Dunn's test was set at 0.05.

A total of seven statistical tests were executed, covering UFs and MFs across dimensions 30, 100, 500, and 1000, as well as FDFs, HF, and CFs. The outcomes of these tests are showcased in Table 3, encapsulating the respective findings and comparisons for each statistical analysis. The p -values derived from the Friedman test are 2.68×10^{-8} , 6.07×10^{-10} , 1.35×10^{-8} , 2.51×10^{-8} , 8.54×10^{-6} , 1.17×10^{-6} , 3.37×10^{-6} , 7.52×10^{-4} , 4.53×10^{-6} , 2.18×10^{-8} for each experiment, respectively. Initially, it is noteworthy that all the p -values obtained from the statistical tests are less than or equal to 0.05, signifying a statistically significant difference among the algorithms concerning their performance. In Table 3, bold p -values indicate that the WPT algorithm outperforms the algorithm mentioned in the corresponding column, while non-bold p -values suggest similar performance between both algorithms. Upon reviewing the rows of Table 2, which display the mean ranks of the optimizers, it becomes evident that the WPT algorithm consistently secured the top mean rank in all the cases. These results highlight the superior performance of the WPT algorithm across multiple scenarios, surpassing the other algorithms in all the cases.

Table 2: The mean ranks obtained from the related-samples Friedman’s two-way analysis of variance by ranks.

	UFs				MFs				FDFs	HF and CFs
	30	100	500	1000	30	100	500	1000		
WPT	1.57	1.86	2.29	2.14	1.92	2.33	2.17	2.75	3.20	1.63
AVOA	4.86	3.21	3.21	3.36	3.83	3.58	3.67	4.00	7.60	4.00
PSO	12.57	11.86	10.71	10.57	12.17	10.83	10.33	9.50	12.10	13.63
GWO	5.29	5.00	6.00	6.57	6.00	6.33	5.83	6.33	9.20	9.13
FFA	8.14	10.00	10.71	10.86	8.83	12.17	10.92	10.67	5.60	4.75
WOA	7.29	6.86	7.00	6.71	7.92	5.17	5.33	5.08	10.10	11.75
TLBO	2.71	3.43	3.29	4.00	6.00	4.50	4.17	6.17	5.20	8.50
MFO	13.71	13.29	12.14	11.43	13.17	12.00	11.67	11.00	7.80	10.38
BBO	9.29	7.86	7.71	9.86	7.67	7.33	10.17	9.50	10.70	7.63
DE	8.71	10.86	11.71	12.00	6.67	9.67	11.00	10.17	4.80	5.38
SSA	9.43	10.14	9.57	8.71	10.75	10.33	9.00	9.50	9.50	8.13
GSA	9.00	9.71	9.86	9.00	9.67	9.83	8.75	8.33	6.10	8.38
IPO	7.86	8.00	8.00	7.43	7.00	8.17	9.00	8.50	5.90	7.13
AHO	4.57	2.93	2.79	2.36	3.42	2.75	3.00	3.50	7.20	4.63
The obtained p-values	2.68E-08	6.07E-10	1.35E-08	2.51E-08	8.54E-06	1.17E-06	3.37E-06	7.52E-04	4.53E-06	2.18E-08

Table 3: The p-values obtained from the Friedman post hoc Dunn’s test for the different configurations.

	AVOA	PSO	GWO	FFA	WOA	TLBO	MFO	BBO	DE	SSA	GSA	IPO	AHO
UFs $d = 30$	1.00E+00	7.90E-05	1.00E+00	2.59E-01	6.21E-01	1.00E+00	5.12E-06	4.98E-02	1.20E-01	3.94E-02	7.81E-02	3.63E-01	1.00E+00
UFs $d = 100$	1.00E+00	7.02E-04	1.00E+00	2.43E-02	9.03E-01	1.00E+00	2.90E-05	4.86E-01	5.16E-03	1.90E-02	3.93E-02	4.22E-01	1.00E+00
UFs $d = 500$	1.00E+00	1.47E-02	1.00E+00	1.47E-02	9.61E-01	1.00E+00	9.45E-04	7.51E-01	2.25E-03	9.69E-02	6.24E-02	6.21E-01	1.00E+00
UFs $d = 1000$	1.00E+00	1.29E-02	9.86E-01	7.62E-03	9.74E-01	1.00E+00	2.53E-03	4.44E-02	7.88E-04	2.41E-01	1.64E-01	7.91E-01	1.00E+00
MFs $d = 30$	1.00E+00	1.98E-03	1.00E+00	3.16E-01	6.94E-01	1.00E+00	2.87E-04	7.94E-01	9.90E-01	2.27E-02	1.14E-01	9.62E-01	1.00E+00
MFs $d = 100$	1.00E+00	3.78E-02	1.00E+00	4.12E-03	1.00E+00	1.00E+00	5.53E-03	9.71E-01	1.93E-01	7.93E-02	1.56E-01	7.60E-01	1.00E+00
MFs $d = 500$	1.00E+00	6.23E-02	1.00E+00	2.56E-02	1.00E+00	1.00E+00	7.39E-03	7.93E-02	2.24E-02	3.43E-01	4.39E-01	3.43E-01	1.00E+00
MFs $d = 1000$	1.00E+00	3.75E-01	1.00E+00	8.99E-02	1.00E+00	1.00E+00	5.56E-02	3.75E-01	1.75E-01	3.75E-01	8.51E-01	7.93E-01	1.00E+00
FDFs	-	8.06E-01	1.51E-04	1.07E-01	1.00E+00	1.84E-02	1.00E+00	7.04E-01	4.91E-03	1.00E+00	6.13E-02	1.00E+00	1.00E+00
HF and CFs	-	1.00E+00	8.76E-07	3.01E-02	1.00E+00	1.18E-04	8.81E-02	2.61E-03	3.13E-01	9.99E-01	1.58E-01	1.08E-01	5.42E-01

5 Conclusion

The paper introduces a novel global optimization approach inspired by the movement behaviors of a specific palm tree species, which navigates from shaded to sunlit areas by strategically growing roots. In contrast to established metaheuristics, the WPT optimizer offers a simple implementation to tackle new optimization challenges, thanks to its clear mathematical formulation. Furthermore, the WPT algorithm involves adjusting two crucial parameters: neighborhood size and reproduction probability. Extensive experiments validate the efficacy of the proposed algorithm. Evaluations are conducted on a set of 31 benchmark functions, encompassing various types including unimodal, multimodal, hybrid, and composite functions. These assessments gauge the algorithm's performance in exploration, exploitation, and avoidance of local optima, as well as its capability to strike a balance between exploration and exploitation phases, and its convergence behavior. Statistical analyses, including Friedman post hoc Dunn's tests, highlight significant improvements achieved by the WPT algorithm across benchmark functions compared to the optimizers utilized in the comparative study.

References

1. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A.H.: The arithmetic optimization algorithm. *Computer methods in applied mechanics and engineering* **376**, 113609 (2021)
2. Adorio, E.P., Diliman, U.: Mvf-multivariate test functions library in c for unconstrained global optimization. Quezon City, Metro Manila, Philippines pp. 100–104 (2005)
3. Ahmadianfar, I., Heidari, A.A., Noshadian, S., Chen, H., Gandomi, A.H.: Info: An efficient optimization algorithm based on weighted mean of vectors. *Expert Systems with Applications* p. 116516 (2022)
4. Ahwazian, A., Amindoust, A., Tavakkoli-Moghaddam, R., Nikbakht, M.: Search in forest optimizer: a bioinspired metaheuristic algorithm for global optimization problems. *Soft Computing* **26**(5), 2325–2356 (2022)
5. Andrews, D.F.: A robust method for multiple linear regression. *Technometrics* **16**(4), 523–531 (1974)
6. Askari, Q., Younas, I., Saeed, M.: Political optimizer: A novel socio-inspired metaheuristic for global optimization. *Knowledge-Based Systems* **195**, 105709 (2020)
7. Azizi, M.: Atomic orbital search: A novel metaheuristic algorithm. *Applied Mathematical Modelling* **93**, 657–683 (2021)
8. Braik, M., Ryalat, M.H., Al-Zoubi, H.: A novel meta-heuristic algorithm for solving numerical optimization problems: Ali baba and the forty thieves. *Neural Computing and Applications* **34**(1), 409–455 (2022)
9. Braik, M.S.: Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems. *Expert Systems with Applications* **174**, 114685 (2021)
10. Castillo-García, N., Hernández Hernández, P.: A new heuristic algorithm for the vertex separation problem. In: *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*, pp. 487–500. Springer (2018)

11. Chahardoli, M., Eraghi, N.O., Nazari, S.: Namib beetle optimization algorithm: A new meta-heuristic method for feature selection and dimension reduction. *Concurrency and Computation: Practice and Experience* **34**(1), e6524 (2022)
12. Dehghani, M., Samet, H.: Momentum search algorithm: A new meta-heuristic optimization algorithm inspired by momentum conservation law. *SN Applied Sciences* **2**(10), 1–15 (2020)
13. Del Ser, J., Osaba, E., Molina, D., Yang, X.S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P.N., Coello, C.A.C., Herrera, F.: Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation* **48**, 220–250 (2019)
14. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering* **137**, 106040 (2019)
15. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. *IEEE computational intelligence magazine* **1**(4), 28–39 (2006)
16. Eberhart, R., Kennedy, J.: Particle swarm optimization. In: *Proceedings of the IEEE international conference on neural networks*. vol. 4, pp. 1942–1948. Citeseer (1995)
17. Fathollahi-Fard, A.M., Hajiaghahi-Keshteli, M., Tavakkoli-Moghaddam, R.: Red deer algorithm (rda): a new nature-inspired meta-heuristic. *Soft Computing* **24**(19), 14637–14665 (2020)
18. Folks, J.L., Chhikara, R.S.: The inverse gaussian distribution and its statistical application—a review. *Journal of the Royal Statistical Society: Series B (Methodological)* **40**(3), 263–275 (1978)
19. Harifi, S., Khalilian, M., Mohammadzadeh, J., Ebrahimnejad, S.: Emperor penguins colony: a new metaheuristic algorithm for optimization. *Evolutionary Intelligence* **12**(2), 211–226 (2019)
20. Hashim, F.A., Hussain, K., Houssein, E.H., Mabrouk, M.S., Al-Atabany, W.: Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. *Applied Intelligence* **51**(3), 1531–1551 (2021)
21. Hashim, F.A., Hussien, A.G.: Snake optimizer: A novel meta-heuristic optimization algorithm. *Knowledge-Based Systems* p. 108320 (2022)
22. Ho, Y.C., Pepyne, D.L.: Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications* **115**(3), 549–570 (2002)
23. Holland, J.H.: Genetic algorithms. *Scientific american* **267**(1), 66–73 (1992)
24. Hudaib, A.A., Fakhouri, H.N.: Supernova optimizer: a novel natural inspired meta-heuristic. *Mod Appl Sci* **12**(1), 32–50 (2018)
25. Jamil, M., Yang, X.S., Zepernick, H.J.: Test functions for global optimization: a comprehensive survey. In: *Swarm intelligence and Bio-inspired Computation*, pp. 193–222. Elsevier (2013)
26. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications* **80**(5), 8091–8126 (2021)
27. Kaveh, A., Dadras, A.: A novel meta-heuristic optimization algorithm: thermal exchange optimization. *Advances in Engineering Software* **110**, 69–84 (2017)
28. Kaveh, A., Ghazaan, M.I.: A new meta-heuristic algorithm: vibrating particles system. *Scientia Iranica. Transaction A, Civil Engineering* **24**(2), 551 (2017)
29. Kaveh, A., Mahjoubi, S.: Lion pride optimization algorithm: a meta-heuristic method for global optimization problems. *Scientia Iranica* **25**(6: Special Issue Dedicated to Professor Goodarz Ahmadi), 3113–3132 (2018)
30. Kaveh, A., Kooshkebaghi, M.: Artificial coronary circulation system: A new bio-inspired metaheuristic algorithm. *Scientia Iranica* **26**(5), 2731–2747 (2019)

31. Kaveh, A., Zaerreza, A.: Shuffled shepherd optimization method: a new meta-heuristic algorithm. *Engineering Computations* (2020)
32. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of ICNN'95-international conference on neural networks*. vol. 4, pp. 1942–1948. IEEE (1995)
33. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *science* **220**(4598), 671–680 (1983)
34. Kumar, S., Jangir, P., Tejani, G.G., Premkumar, M., Alhelou, H.H.: Mopgo: A new physics-based multi-objective plasma generation optimizer for solving structural optimization problems. *IEEE Access* **9**, 84982–85016 (2021)
35. Lipowski, A., Lipowska, D.: Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* **391**(6), 2193–2196 (2012)
36. Luke, S.: *Essentials of metaheuristics*, vol. 2. Lulu Raleigh (2013)
37. Mirjalili, S.: Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-based systems* **89**, 228–249 (2015)
38. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software* **114**, 163–191 (2017)
39. Mirjalili, S., Lewis, A.: The whale optimization algorithm. *Advances in engineering software* **95**, 51–67 (2016)
40. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. *Advances in engineering software* **69**, 46–61 (2014)
41. Mohamed, A.W., Hadi, A.A., Mohamed, A.K.: Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. *International Journal of Machine Learning and Cybernetics* **11**(7), 1501–1529 (2020)
42. Mohammadi-Balani, A., Nayeri, M.D., Azar, A., Taghizadeh-Yazdi, M.: Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Computers & Industrial Engineering* **152**, 107050 (2021)
43. Mozaffari, M.H., Abdy, H., Zahiri, S.H.: Ipo: an inclined planes system optimization algorithm. *Computing and Informatics* **35**(1), 222–240 (2016)
44. Nelsen, R.B.: Heron's formula via proofs without words. *The College Mathematics Journal* **32**(4), 290 (2001)
45. O'Reilly, F., Rueda, R.: A note on the fit for the levy distribution. *Communications in Statistics-Theory and Methods* **27**(7), 1811–1821 (1998)
46. Oyelade, O.N., Ezugwu, A.E.S., Mohamed, T.I., Abualigah, L.: Ebola optimization search algorithm: A new nature-inspired metaheuristic optimization algorithm. *IEEE Access* **10**, 16150–16177 (2022)
47. Połap, D., Woźniak, M.: Red fox optimization algorithm. *Expert Systems with Applications* **166**, 114107 (2021)
48. Price, K., Storn, R.M., Lampinen, J.A.: *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media (2006)
49. Qi, X., Zhu, Y., Zhang, H.: A new meta-heuristic butterfly-inspired algorithm. *Journal of computational science* **23**, 226–239 (2017)
50. Rao, R.V., Savsani, V.J., Vakharia, D.: Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. *Information sciences* **183**(1), 1–15 (2012)
51. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: Gsa: a gravitational search algorithm. *Information sciences* **179**(13), 2232–2248 (2009)
52. Salcedo-Sanz, S., Del Ser, J., Landa-Torres, I., Gil-López, S., Portilla-Figueras, J.: The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems. *The Scientific World Journal* **2014** (2014)

53. Sampson, J.R.: Adaptation in natural and artificial systems (john h. holland) (1976)
54. Shayanfar, H., Gharehchopogh, F.S.: Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. *Applied Soft Computing* **71**, 728–746 (2018)
55. Simon, D.: Biogeography-based optimization. *IEEE transactions on evolutionary computation* **12**(6), 702–713 (2008)
56. Talatahari, S., Azizi, M., Gandomi, A.H.: Material generation algorithm: a novel metaheuristic algorithm for optimization of engineering problems. *Processes* **9**(5), 859 (2021)
57. Tovey, C.A.: Nature-inspired heuristics: Overview and critique. In: *Recent Advances in Optimization and Modeling of Contemporary Problems*, pp. 158–192. INFORMS (2018)
58. Wagan, A.I., Shaikh, M.M., et al.: A new metaheuristic optimization algorithm inspired by human dynasties with an application to the wind turbine micrositeing problem. *Applied Soft Computing* **90**, 106176 (2020)
59. Wang, G.G., Deb, S., Coelho, L.D.S.: Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems. *International journal of bio-inspired computation* **12**(1), 1–22 (2018)
60. Yang, X.S.: *Nature-inspired optimization algorithms*. Elsevier (2014)
61. Yang, X.S.: Swarm intelligence based algorithms: a critical analysis. *Evolutionary intelligence* **7**(1), 17–28 (2014)
62. Yang, X.S., Deb, S., Fong, S.: Metaheuristic algorithms: optimal balance of intensification and diversification. *Applied Mathematics & Information Sciences* **8**(3), 977 (2014)
63. Yang, X.S., Deb, S., Fong, S., He, X., Zhao, Y.X.: From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. *Computer* **49**(9), 52–59 (2016)
64. Yang, X.S., He, X.: Firefly algorithm: recent advances and applications. *International journal of swarm intelligence* **1**(1), 36–50 (2013)
65. Zitouni, F., Harous, S., Belkeram, A., Hammou, L.E.B.: The archerfish hunting optimizer: a novel metaheuristic algorithm for global optimization. *Arabian Journal for Science and Engineering* pp. 1–41 (2021)
66. Zitouni, F., Harous, S., Maamri, R.: The solar system algorithm: a novel metaheuristic method for global optimization. *IEEE Access* **9**, 4542–4565 (2020)
67. Zitouni, F., Harous, S., Maamri, R.: A novel quantum firefly algorithm for global optimization. *Arabian Journal for Science and Engineering* **46**(9), 8741–8759 (2021)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial 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.

