Engineering with Computers (2022) 38 (Suppl 4):S3191–S3215
https://doi.org/10.1007/s00366-021-01460-1

ORIGINAL ARTICLE

Hazelnut tree search algorithm: a nature-inspired method for solving numerical and engineering problems

Hojjat Emami[1]

Received: 15 February 2021 / Accepted: 16 June 2021 / Published online: 26 June 2021

Abstract
In this paper, a novel nature-inspired optimization algorithm, hazelnut tree search (HST) is proposed for solving numerical and engineering optimization problems. HST is a multi-agent algorithm that simulates the search process for finding the best hazelnut tree in a forest. The algorithm is composed of three main actuators: growth, fruit scattering, and root spreading. In the growth phase, trees compete with each other on shared resources to grow up and improve their fitness. In the fruit scattering phase, HTS performs exploration by simulating the movement of hazelnuts around the forest with the help of animals and rodents. In the root spreading, HTS performs exploitation by modeling the root spreading mechanism of trees around themselves. The performance of the proposed algorithm is evaluated on multi-variable unconstraint numerical optimization benchmarks and constraint engineering problems. Comparing the proposed algorithm with a few other optimization algorithms shows the superiority of the HTS in terms of problem-solving success and finding the global optimum on most benchmark problems.

Keywords Evolution · Numerical optimization · Nature-inspired algorithm · Hazelnut tree search algorithm

1 Introduction

Optimization is the process of maximizing desired factors and minimizing undesired ones to make something as perfect, functional, or effective as possible. In recent years, many meta-heuristic algorithms have been emerged to solve optimization problems. These algorithms can not guarantee to find the optimum solutions, but they can find near-optimal solutions in a limited time for complex non-differentiable, and non-linear problems [1]. Several surveys and comparison frameworks of meta-heuristic algorithms are given in [2–4]. The meta-heuristic algorithms can be categorized into several groups based on various classification factors. Two well-known classification factors are the source of inspiration and the number of solutions.

According to the source of inspiration, meta-heuristics may be classified into four main groups [3]: human-inspired, animal-inspired, physics and chemistry inspired, and nature-inspired. Human inspired algorithms mimic the social and collective behavior of human beings. For example, the class topper optimization (CTO) algorithm [5] is inspired by the learning behavior of students in a class of school. The students compete with each other to enhance their learning performance and become the topper of the class.

Animal-inspired algorithms simulate the social and collective behavior of various creatures such as ants, birds, flocks, wolves, and fishes. Three popular examples of animal-inspired algorithms are particle swarm intelligence (PSO) [6], ant colony optimization (ACO) [7], and artificial bee colony (ABC) [8]. Some other recently proposed algorithms are firefly algorithm (FA) [9], Krill herd (KH) [10], spider monkey optimization algorithm [11], grey wolf optimization (GWO) [12], whale optimization algorithm (WOA) [13], squirrel search algorithm (SSA) [14], and grasshopper optimization algorithm (GOA) [15].

Physics and chemistry-inspired algorithms are motivated by chemical or physical phenomena such as magnetic force, gravity, annealing, and Lichtenberg patterns. Four popular algorithms in this category are simulated annealing (SA) [16], gravitational search algorithm (GSA) [17], magnetic optimization algorithm (MOA) [18], and Lichtenberg algorithm (LA) [19, 20]. The LA algorithm is inspired by the Lichtenberg figures and the physical phenomenon of radial

✉ Hojjat Emami
emami@ubonab.ac.ir

[1] University of Bonab, Bonab, Iran

intra-cloud lightning. LA is different from many other algorithms because it performs the search based on population and trajectory.

In nature-inspired algorithms, the sources of inspiration are the special behavior of organisms, plants, and biological events. According to the characteristics of natural biological systems in solving complicated problems, nature-inspired algorithms are more powerful in solving optimization problems. Examples include genetic algorithm (GA) [21], runner-root algorithm (RRA) [22], forest optimization algorithm (FOA) [23], flower pollination algorithm (FPA) [24], farmland fertility algorithm (FFA) [25], and sunflower optimization algorithm (SFO) [26, 27]. FPA simulates the reproduction and pollination behavior of the flowers in nature. SFO models the peculiar behavior of sunflowers in searching for the best orientation towards the sun [27]. The inspiration source of FPA and SFO is similar, but their way of modeling the problem is different. The FOA simulates the seeding behavior of trees in forests that try to extend their territory and lifespan through seed dispersal. They proposed two seed dispersal approaches including local seeding and global seeding. In local seeding, some seeds fall near the trees, while in global seeding, the seeds are distributed throughout the forest with the help of animals and natural procedures such as wind and water flows. The FOA only focused on the seeding process and ignored the more complex procedures such as competition that can be lead to better results. SBA [28], and RRA are two nature-inspired algorithms that simulate the function of runners and root spreading mechanisms of plants. Some plants such as strawberry expand runners and roots for propagation and searching for water resources and minerals. SBA and RRA attempt to provide a proper balance between local and global searches through formulating the function of runners and root hairs of plants. Runners increase the exploration power of the algorithm through searching around the plant with random big steps, while the root spreading provides the exploitation ability by searching the around of plant with small steps.

According to the number of solutions in the search process, meta-heuristics are divided into two main categories [12]: single-solution based and population-based. In the former class, the algorithm begins the search process by one single candidate solution. This single solution is then updated and improved over the course of iterations [12]. The most well-known algorithm in this branch is simulated annealing (SA) [16]. Single solution-based meta-heuristics are relatively simple and require a low number of function evaluations, however, these techniques may stick at local optima [29]. Population-based techniques work on a set of solutions collectively known as population. In this case, the algorithms begin the optimization process with a randomly initialized population which is improved throughout iterations by applying population-updating operators. Population-based algorithms can avoid local optimum points of the optimization problems. Some of the most well-known population-based algorithms are differential evolution (DE) [30], GA, PSO, ACO, ABC, and GWO.

It is important to note that each of the above-mentioned meta-heuristics has been enhanced over the years and different versions of them are developed. The enhanced versions are varying in their working principle and operators. Some examples include binary farmland fertility algorithm (BFFA) [31], proposed hybrid optimization algorithm (PHOA) [32], and hybrid multi–population algorithm (HMPA) [33]. BFFA improved the original FFA by introducing three operators including binary global memory update (BGMU), binary local memory update (BLMU), and a dynamic mutation (DM). PHOA is an efficient optimization algorithm developed based on atom search optimization (ASO) and tree seed algorithm (TSA). PHOA also is equipped with levy flight random walk and chaos theory. With a test on various numerical functions and engineering problems, PHOA obtained superior results compared with its counterparts. The main advantages of PHOA are providing a proper balance between exploration and exploitation, and avoiding local optima. HMPA is a combination of artificial ecosystem-based optimization (AEO) and Harris Hawks optimization (HHO) algorithms. It is also equipped with chaos theory, local search strategy, quasi-oppositional learning, and levy flight mechanism to maximize the performance of the HMPA. The statistical results demonstrate that HMPA outperformed the counterpart algorithms in terms of solution quality and convergence performance.

However, the development of optimization algorithms are dated back quite early and many algorithms proposed so far, this topic has regained importance recently. The reason for this issue can be found in the no free lunch (NFL) theorem [34], which states that there is no algorithm for solving all optimization problems. Thus, introducing new optimization algorithms or improving the existing techniques is needed.

In the meantime, a few researches have attempted to model the behavior of plants in nature. The existing works only model very limited characteristics of plants. According to this issue and the NFL theory, this paper introduces a hazelnut tree search (HTS) algorithm, which is an attempt to model the growth and the interaction of hazelnut trees in nature to solve optimization problems. The HTS algorithm is equipped with three operators including growth, fruit scattering, and root spreading to explore the entire solution space and exploit the promising areas. These operators hopefully cause the trees to converge to the global optimum of the problem. To summarize, the main contributions of this research include:

– Introducing a nature-inspired optimization algorithm, namely hazelnut tree search (HTS), which models the

search process for finding the fittest hazelnut tree in a forest.
– Evaluating the potential of HTS in solving numerical and engineering optimization problems, and compare it with counterpart algorithms. The results confirm that the HTS achieves superior results compared with its counterparts on most benchmark problems.

The remaining parts are organized as follows. The HTS algorithm is introduced in Sect. 2. The experimental results on unconstraint optimization test problems are presented in Sect. 3. The application of HTS for solving engineering problems is discussed in Sect. 4. Finally, Sect. 5 gives some conclusions and proposes several directions for future researches.

2 Hazelnut tree search (HTS) algorithm

This section presents the source of inspiration and the mathematical model of the proposed HTS algorithm. Also, an example is presented to show the working principle of HTS.

2.1 Source of inspiration

Trees are one of the oldest entities in the world [23]. The hazel (Corylus) is a kind of large shrubs and deciduous trees native to the temperate northern hemisphere. Similar to other trees, the life cycle of hazel trees includes seed, sprout, seedling, sapling, mature, and death. Figure 1 shows a schematic view of the life cycle of a hazel tree in nature.

Every year in the spring, trees bloom and begin to grow. Anywhere the environmental conditions are favorable for germination, some seeds that fall on the ground in the past autumn start to germinate and sprout. Some old trees will replace with new ones, and this process continues. Trees need adequate water, nutrients, sunlight, proper temperature, and space to grow up and get more robust.

In the growing season, trees have to compete on shared resources to grow as much as the environmental conditions permit. In competition, trees have a negative effect on other ones through direct interference or indirect exploitation of limited resources [36]. Survival of the fittest trees is the main principle of the competition phase. The success or failure of a tree in the competition phase is relying on its physical conditions including height, root spread, sunlight absorption, and accessing water resources. The competition gradually decreases or increases the fitness of trees. The distinguished trees that are in suitable environmental conditions can get more resources and become stronger, while weak trees grow slowly and maybe weakened or collapsed [37]. In addition to the competition, different stresses such as diseases, harsh cold weather, insects, and injuries can cause the trees to weaken and die.

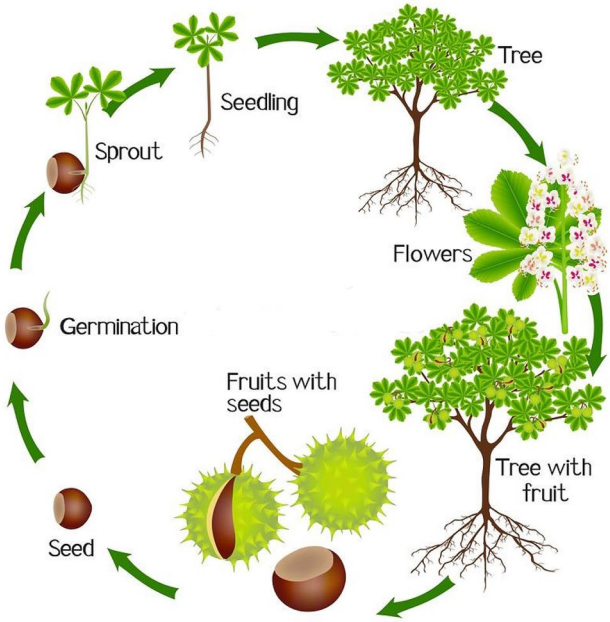


Fig. 1 Growth cycle of a hazelnut tree

In the autumn, hazelnuts are scattered with the help of gravity, water, wind, and animals in different areas [38, 39]. Seeds may disperse into different areas with the help of natural mechanisms or animals, especially squirrels, or fall near the trees. Squirrels are known for eating nuts, especially hazelnuts. Most species of squirrels are scatter-hoarders meaning they split up hazelnuts and store them in many different locations to survive in the winter. This type of food storage is an evolved process because it reduces the risk of suffering total loss. Fruit dispersal is essential for trees as it guarantees the survival of trees, increases the diversity of the population, and expands the territory of trees in nature.

In the winter, trees go into hibernation and spend the winter. The fittest trees pass the winter and continue their growth in the next spring, but some of them collapsed. This cycle continues for many years. The growth of trees is an optimization process, in which the main objective of each tree is to get more robust and could survive for a long time. This is proportionate with the rule of "survival of the fittest". The HTS algorithm attempts to model this optimization process by modeling the growth, fruit scattering, and root spreading procedures. The main goal is to find the fittest hazelnut tree in the forest that corresponds to the optimal solution for a given problem. Figure 2 shows a big picture of competition, fruit scattering, and root spreading of trees.

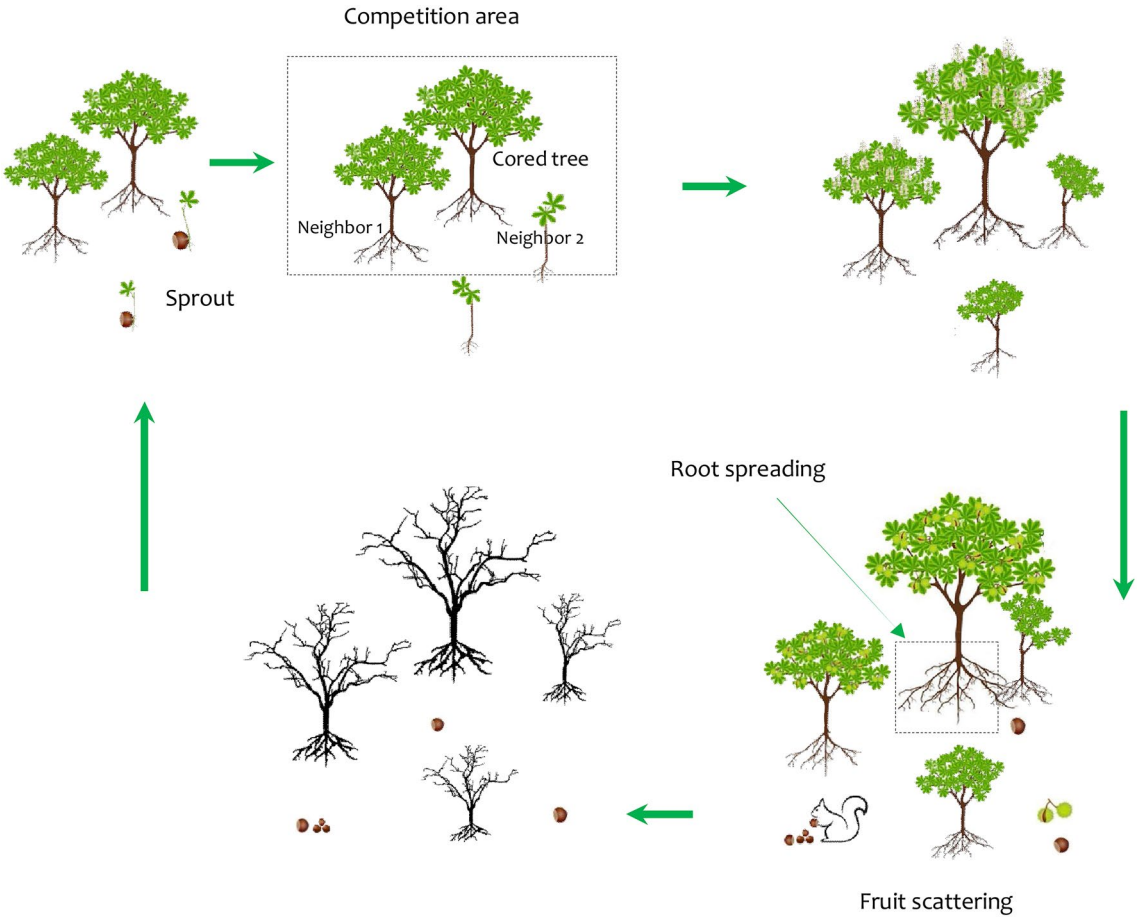


Fig. 2 A big picture of competition, fruit scattering and root spreading of hazelnut trees

2.2 Mathematical model

HTS is an iterative and population-based algorithm. As shown in Fig. 3, it has the following components:

- A **population initialization** phase that forms a set of representative solutions dispersed randomly in the solution space.
- A **fitness function** that calculates the fitness of the individuals.
- A **growth calculation** phase that computes the effect of trees on each other in competition on shared resources.
- A **fruit scattering** step that explores the search space to find promising solutions beyond the current region.
- A **root spreading** phase which performs a chaotic local search around trees to exploit optimal solutions.
- A **termination condition** that recommends when to stop the algorithm's execution

The mathematical formulation of the operators is described in the following.

2.2.1 Population initialization

For a generalized unconstraint D-dimensional problem $f(x) = f(x_1, x_2, \ldots, x_D)$, a randomly distributed population G of trees is initialized. The population is referred to as forest. Each tree $H_i \in G$ in the forest is a real-valued $1 \times D$ vector defined as

$$H_i = [h_{i1}, h_{i2}, \ldots, h_{iD}] \tag{1}$$

each element $h_{ij} \in H_i$ is initialized as follows:

$$h_{ij} = U(0, 1) \times (ub_j - lb_j) + lb_j \tag{2}$$

where $U(0, 1)$ is a uniformly distributed random number in the range [0, 1], ub_j and lb_j are upper and lower bounds of H_i in jth dimension, respectively. Each tree has a fitness found by evaluating the fitness function f. The fitness of each tree is related to the objective function of the problem. The trees that are located in good areas grow better and obtain higher fitness.

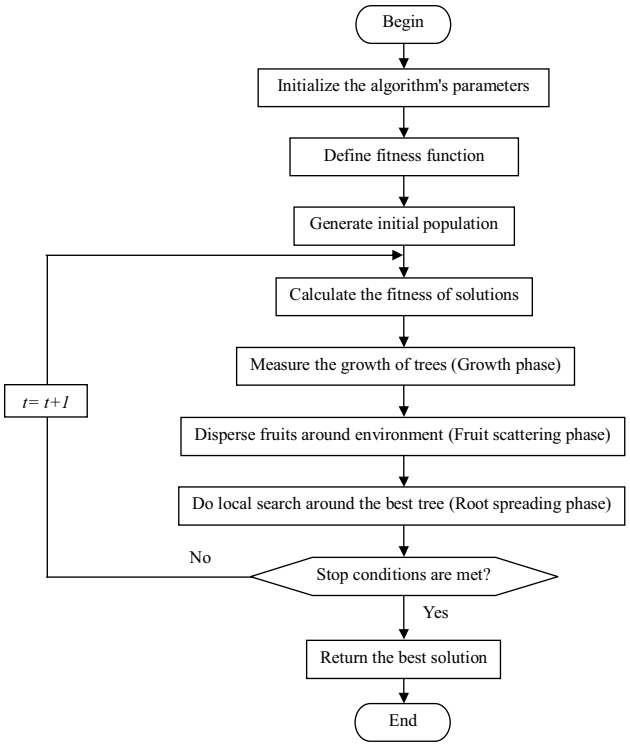


Fig. 3 Flowchart of the HTS algorithm

2.2.2 Growth

Some trees that are in low-dense areas grow without competition with others. However, some others in dense areas have to compete with neighbors on common resources [36]. The following equation is proposed to model the growth of a tree H_i.

$$H_i^{t+1} = H_i^t + \left(\alpha g + (1-\alpha)\frac{1}{1+\lambda} g\right) \tag{3}$$

where H_i^t is the position of tree H_i at current generation t, g is the growth rate of tree H_i in the absence of neighbor trees, λ is the competition index, and α is a switch probability that controls the growth of tree in sparse or dense areas. α is a binary random number, where $\alpha = 1$ indicates that the tree H_i is in sparse area and grows without taking part in competition with neighbors, and $\alpha = 0$ means that the growth of trees decreases monotonically with λ. The growth rate g is computed as follows:

$$g = \left(1 - e^{-\left(\frac{t}{T}\right)^2}\right) \times a \quad \text{where} \quad a = U(-\gamma, \gamma) \tag{4}$$

t is the current iteration and T indicates the maximum number of iterations. The symbol a is a $1 \times D$ vector of uniform random numbers in the range $[-\gamma, +\gamma]$, where γ is a uniform random number in the range [0, 1]. The coefficient a adds a random amount of deviation in variable values of H_i. It is considered to further search around the tree H_i. Equation (4) follows the Weibull equation [40], due to its ideal match for tree growth in nature. There are various formulations to model tree growth such as Bertalanffy, Logistic, Sloboda, and Levakovic, among others [41]. However, the Weibull equation is more accurate than others in computing the growth amount of trees.

In the literature, different equations are defined to calculate the competition index λ. Contreras et al. [36] present a comprehensive survey of competition indexes. In this paper, we define the competition index λ as follows:

$$\lambda = \sum_{j=1}^{Z_i} \left(\frac{f_i^t}{f_j^t}\right) \times \arctan\left(\frac{f_j^t}{d_{ij}^t}\right) \tag{5}$$

where Z_i is the number of neighbor trees around H_i at current iteration, f_j is the fitness of the jth neighbor at iteration t; f_i is the fitness of H_i, and d_{ij} denotes the distance from the jth neighbor to H_i.

$$d_{ij}^t = \left\| H_j^t - H_i^t \right\| \tag{6}$$

Z_i is defined as follows:

$$Z_i = \lceil \mu_i \times N \rceil \tag{7}$$

where N is the population size, μ_i is the normalized fitness of H_i, computed as

$$\mu_i = \frac{\max(K) - f_i}{\sum_{k=1}^{N} \left(\max(K) - f_k\right)}, \quad \text{where} \quad K = \{f_k | k = 1, 2, \ldots, N\} \tag{8}$$

Figure 4 shows the schematic of the horizontal angles from a cored tree's center to each neighbor when computing the competition index λ. According to Eqs. (7) and (8), the fittest trees will compete with few neighbors while weak trees have to compete with more neighbors. This is consistent with the laws of nature; the weak trees are often located in densely populated areas where neighbor trees weaken them over generations. Next, Z_i closest trees are selected from

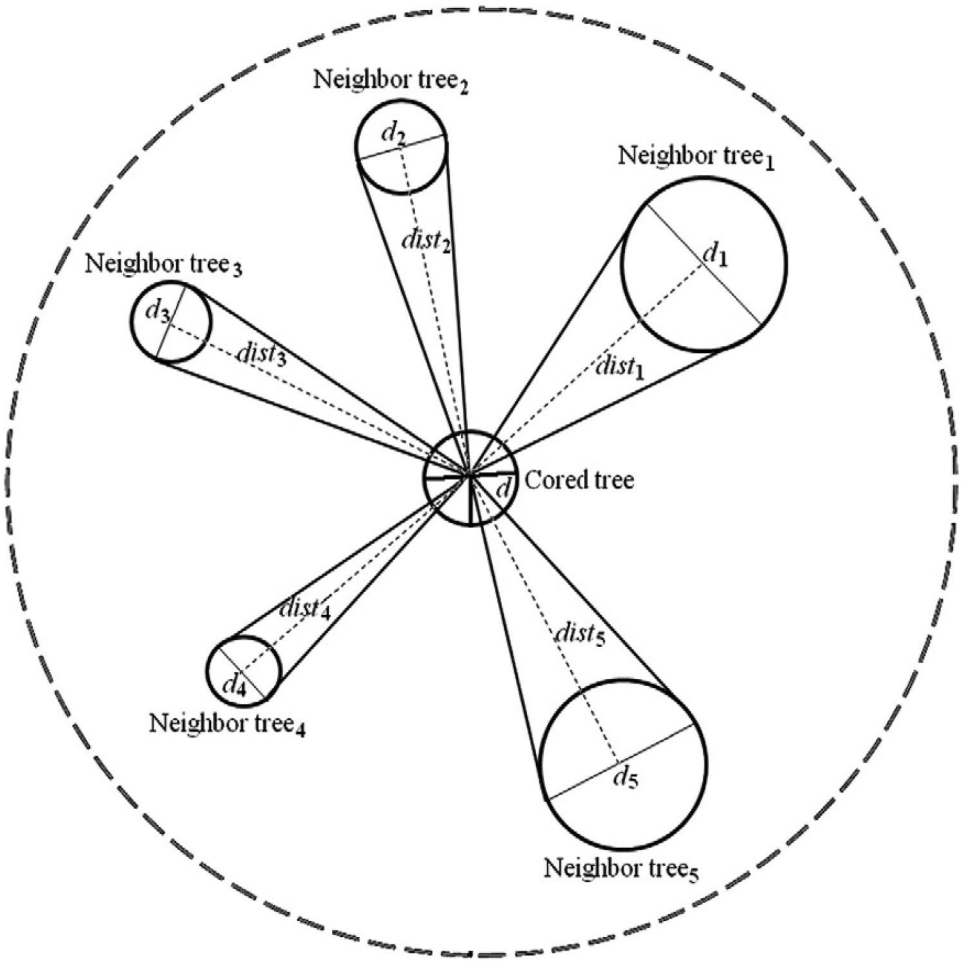


Fig. 4 Schematic of the horizontal angles from a cored tree's center to each neighbor within the neighborhood zone used to compute λ

the population to form the neighborhood zone Ng_i of H_i as follows:

$$Ng_i = \left[H_1, H_2, \ldots, H_{Z_i}\right] \quad H_i \notin Ng_i \tag{9}$$

We first calculate the distance of trees from each other and then consider the nearest trees as neighbors of H_i. Because the distance calculation between trees is time-consuming, the neighbors can be identified randomly. However, the random selection of neighbors is not compatible with the laws of nature. The trees in list Ng_i are those that take part in computing Eq. (5). The smaller neighbor tree with less fitness will have less impact on the cored tree H_i, but the fittest neighbor will weaken the H_i. The success or failure of a tree in the competition phase is relying on its physical conditions including height, root spread, sunlight absorption, and accessing water resources. The competition gradually decreases or increases the fitness of trees. In general, the fitness of strong trees is expected to increase and conversely decrease the fitness of weak trees. However, weak trees have the chance to win the competition.

Survival of the fittest trees is the main inspiration of the competition phase. Competition operator increases the exploitation ability of the algorithm, because it makes some deviations in the previous values of the trees to examine the points around their vicinity.

2.2.3 Fruit scattering

To mathematically model the fruit scattering phase, first, the trees are sorted according to their current fitness in descending order, and then $\{H_1, H_2, \ldots, H_p\}$ trees are selected for taking part in the fruit scattering process. p is the number

of trees that are selected at iteration t to take part in the fruit dispersal phase, defined as follows:

$$p^t = \lfloor c^t \times N \rfloor \tag{10}$$

c^t is the fruit scattering rate at iteration t, which controls the number of trees that will participate in the fruit scattering process. Increasing the fruit scattering rate increases the algorithm's freedom to search outside the current region of solution space; however, it increases the computational complexity. To prevent an ever-increasing population size, it is assumed only one fruit is generated from each tree at every iteration. Therefore, the number of seedlings at every generation is p^t. The position of a seedling generated from a seed that has fallen to the ground is calculated as follows:

$$P_i = H_i^t + \Lambda(H_i^t - H_{\text{best}}^t) \tag{11}$$

where H_i^t is the tree from which the seed fell on the ground. H_{best}^t indicates the best solution found among all of the solutions at iteration t. Λ is a random number generated using Levy flight distribution [24]. Λ is defined as follows:

$$\Lambda \sim \frac{m\Gamma(m)\sin(\pi m/2)}{\pi}\frac{1}{w^{1+m}}, \quad w \gg w_0 > 0, \quad w_0 = 0.1 \tag{12}$$

$\Gamma(m)$ denotes the standard gamma distribution, where m is set to be 1.5. The variable w is computed as follows:

$$w = \frac{U}{|V|^{1/m}}, \quad U \sim N(0, \delta^2), \quad V \sim N(0, 1) \tag{13}$$

where U and V are Gaussian distributions, $N(0, \delta^2)$ indicates the normal distribution with mean 0 and variance δ^2, and $N(0, 1)$ is the standard normal distribution

Equation (11) simulates the condition in which the fruits may be scattered further away by the help of wind, water flows, and animals. Since the algorithm should do more global searches at the beginning iterations, and more local searches as it proceeds toward the end, the fruiting rate c should be greater at initial iterations and decreases over generations. Thus, c at each iteration t is defined as

$$c^t = c_{\max} - \frac{t}{T}(c_{\max} - c_{\min}) \tag{14}$$

where t is the current iteration, and T is the maximum iterations. $c_{\max}$ and $c_{\min}$ are the upper and lower bound of c. The variables $c_{\max}$ and $c_{\min}$ are identified empirically via statistical tests.

Finally, a check is made between the tree H_i^t and the seedling P_i, and whichever has gained better fitness is considered for further generation.

$$H_i^{t+1} = \begin{cases} P_i & \text{if } f(P_i) > f(H_i^t) \\ H_i^t & \text{otherwise} \end{cases} \tag{15}$$

The fruit scattering phase increases the exploration of the algorithm and keeps the algorithm from converging too fast before searching the entire solution space.

2.2.4 Root spreading

The root spreading operator is proposed to increase the exploitation power and convergence speed of the algorithm by performing a local search. It is important to notice that the local search is performed only around the current fittest tree H_{best}. This method has two advantages: (i) there is a high probability that a better solution could be found in the space near to the current best solution, and (ii) comparing with performing the local search on all trees, it has less computational cost because local search is carried out on the fittest tree only once at each generation. In each iteration, root spreading operator forms a neighborhood hypercube $[H_{\text{best}} - r, H_{\text{best}} + r]^D$ around the best tree H_{best} and exploits all its dimensions in turn to generate new solutions $S_k, k = \{1, 2, \ldots, D\}$, where D is the number of dimensions. Let $S_k = \{s_1, s_2, \ldots, s_D\}$ and $H_{\text{best}} = \{h_{\text{best},1}, h_{\text{best},2}, \ldots, h_{\text{best},D}\}$, each component $s_j \in S_k$ is updated as follows:

$$s_j = \begin{cases} h_{\text{best},j}^t + r^t(ub_j - lb_j)(q^t - 0.5) & j == k \\ h_{\text{best},j}^t & j \neq k \end{cases} \tag{16}$$

where s_j indicates the position to which the tree has taken root. r^t is the search radius around the best tree H_{best}, which decreases over generations. lb_j and ub_j are the lower and upper bound of jth variable, respectively. q^t is a chaotic variable generated at iteration t by a chaotic map given in Table 1. If the value of s_j violates the search boundary, it is reset to the closest value in the search boundary. Notice that all dimensions of S_k use the same value of a chaotic map. Due to the ergodicity and randomicity of chaos [42], using the chaotic variables instead of random numbers improves the search efficiency of the algorithm. Chaotic local search works better in a small range. To narrow the search range

Table 1 Characteristics of chaotic maps used in the root spreading phase

No.	Name	Equation
M_1	Chebyshev map	$q_{n+1} = \cos(n.\cos^{-1} q_n)$
M_2	Circle map	$q_{n+1} = q_n + a - b.\sin(2\pi n) \bmod (1);\ a = 0.2, b = 0.5$
M_3	Gaussian map	$q_{n+1} = \begin{cases} 0 & q_n = 0 \\ (1/q_n) \bmod (1) & q_n \neq 0 \end{cases}$
M_4	Iterative	$q_{n+1} = \sin\left(\frac{a\pi}{q_n}\right);\ a \in (0,1)$
M_5	Logistic map	$q_{n+1} = a.q_n(1-q_n);\ q_0 \notin \{0, 0.25, 0.5, 0.75, 1\},\ a = 4$
M_6	Piecewise map	$q_{n+1} = \begin{cases} q_n/P & 0 \le q_n < P \\ \frac{q_n - P}{0.5 - P} & P \le q_n < 0.5 \\ \frac{1-P-q_n}{0.5-P} & 0.5 \le q_n < 1-P \\ \frac{1-q_n}{P} & 1-P \le q_n < 1 \end{cases};\ P \in (0, 0.5),\ P \neq 0$
M_7	Sine map	$q_{n+1} = \frac{a}{4}\sin(\pi q_n);\ 0 < a \le 4$
M_8	Singer map	$q_{n+1} = \mu(7.86q_n - 23.31q_n^2 + 28.75q_n^3 - 13.302875q_n^4);\ \mu = 1.07$
M_9	Sinusoidal map	$q_{n+1} = q_n^2 \sin(\pi q_n);\ a = 2.3,\ q_0 = 0.7$
M_{10}	Tent map	$q_{n+1} = \begin{cases} \frac{q_n}{0.7} & q_n < 0.7 \\ \frac{10}{3} q_n(1-q_n) & \text{otherwise} \end{cases}$

over generations and improve the local search efficiency, the variable r is defined as follows:

$$r^t = \frac{1}{e^{t/T}} \tag{17}$$

In each generation, the root spreading phase generates D solutions in local search. Finally, the best solution is identified among new solutions as follows:

$$S_{\text{best}} = \max\left\{S_1, S_2, \ldots, S_D\right\} \tag{18}$$

where max returns the solution whose fitness is the maximal. If the fitness of the new best solution S_{best} is better than that of the current best solution H_{best}, S_{best} replaces it to be considered as the best solution for the next generation.

$$H_{\text{best}}^{t+1} = \begin{cases} S_{\text{best}} & \text{if } f(S_{\text{best}}) > f(H_{\text{best}}^t) \\ H_{\text{best}}^t & \text{otherwise} \end{cases} \tag{19}$$

The main idea behind the above equation is to realize the principle of the survival of the fittest, which dictates that the fittest tree should continue its growing and get more robust. In Eq. (19), the problem is considered to be maximization problem.

The pseudo-code of the proposed HTS is shown in Algorithm 1. Until termination conditions are met, the algorithm iterates the growth, fruit scattering, and root spreading phases on the population. Finally, the fittest tree is returned as an optimal solution for the problem.

2.3 A numerical example to show the functioning of HTS

Algorithm 1: Pseudo code of the proposed HTS algorithm

Input: The population size N, maximum iterations T, maximum number of function evaluations M_f
Output: The fittest hazelnut tree H_{best} and its fitness

Initialize algorithm parameters;
Create initial population H_i, $i = 1, 2, \ldots, N$ by Eqs. (1) and (2);
Calculate the fitness of trees;
Find the best solution H_{best} in the population;

$t = 0$;
while *($t \leq T$ or $f \leq M_f$)* **do**

  // Growth phase;
  for *i=1 to N* **do**
    Compute growth rate g^t by Eq. (4);
    α=generate a binary random number;
    if *(α==0)* **then**
      Identify the neighbors of H_i^t by Eqs. (7)-(9);
      Compute the competition index λ by Eq. (5);
    end
    Compute the growth of tree H_i^t by Eq. (3);
    Evaluate the fitness of tree H_i^t;
  end

  // Fruit scattering phase;
  Sort the trees according to their current fitness;
  Calculate p^t by Eq. (10);
  for *i=1:p^t* **do**
    Compute Λ by Eq. (12);
    Generate seedling P_i by Eq. (11);
    Evaluate the fitness of P_i;
    Update c by Eq. (14);
    Compare the fitness of S_i and H_i^t to compute H_i^{t+1} by Eq. (15);
  end

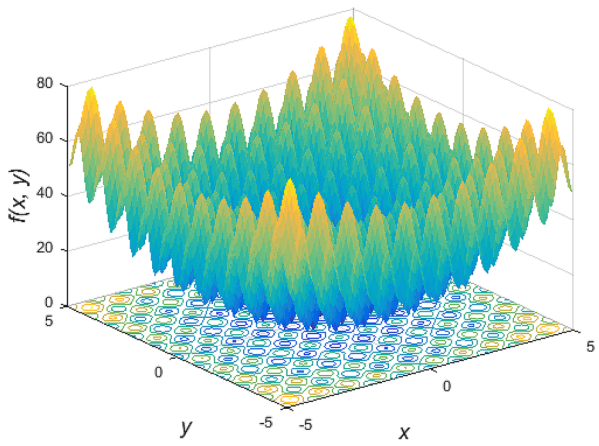
  // Root spreading phase;
  Find the current best solution H_{best}^t;
  Generate chaotic variable q^t using a chaotic map M_i provided in Table 1;
  for *k=1 to D* **do**
    Compute new solution S_k, the position to which the tree has taken root by Eq. (16);
    Calculate the fitness of S_k;
  end
  Update search radius r by Eq. (17);
  Find the best solution S_{best} by Eq. (18);
  Compare the fitness of S_{best} and H_{best}^t to compute the best solution H_{best}^{t+1} by Eq. (19);

  Calculate the new value of M_f;
  $t = t + 1$;
end
Return the fittest tree H_{best} and its fitness;

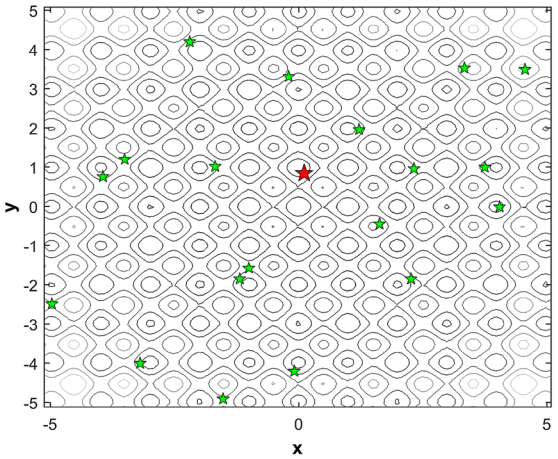
Figure 5 shows the functioning of HTS on a two-dimensional rastrigin function, which is defined as follows:

$$f(x, y) = 20 + x^2 + y^2 - 10\cos(2\pi x) - 10\cos(2\pi y) \quad (20)$$

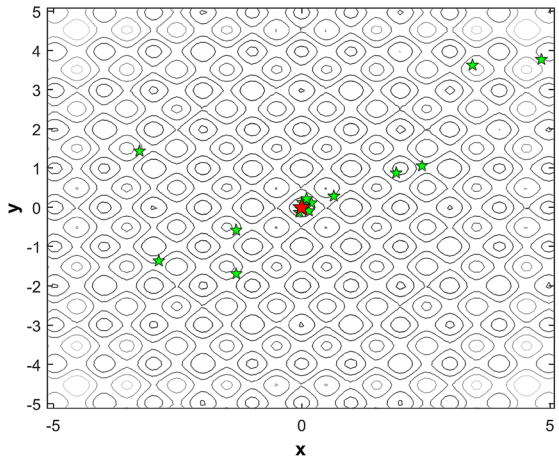
This test case is a two-dimensional multimodal function with several local minima. Its global optimum is 0 located at $(x, y) = (0, 0)$, and the search boundary is $[-5.12, 5.12]$. Figure 5a shows the graphical plot of test function. Figure 5b illustrates the initial forest with 20 trees that are scattered throughout the solution space. Figure 5c–d shows how trees gradually converge to the global minimum point at



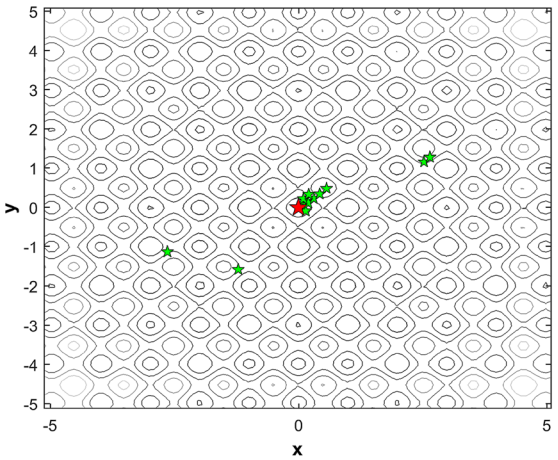
(a) benchmark problem



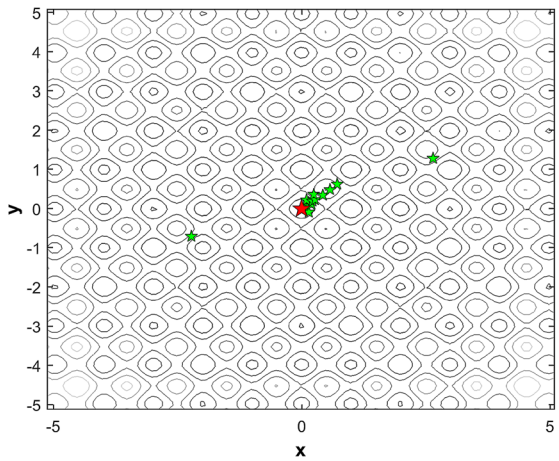
(b) initial forest with 20 trees



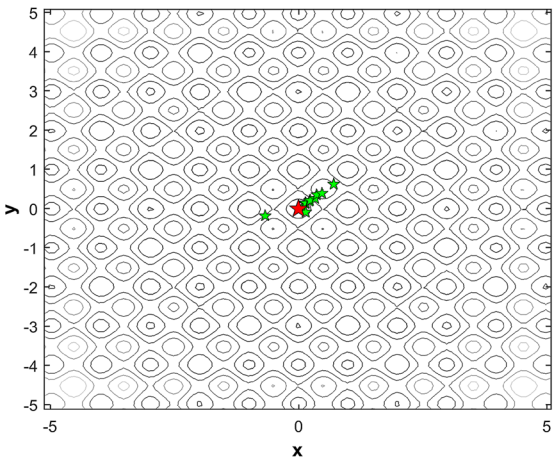
(c) iteration 5



(d) iteration 10



(e) iteration 15



(f) iteration 20

Fig. 5 The functioning of the proposed HTS on a test function

Table 2 Characteristics of group I unimodal test functions

Function	Name	Formulation	Domain	F_{opt}
f_1	Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	[−100 100]	0
f_2	Sum squares	$f(x) = \sum_{i=1}^{D} ix_i^2$	[−10, 10]	0
f_3	Quartic	$f(x) = \sum_{i=1}^{D} ix_i^4 + rand$	[−1.28, 1.28]	0
f_4	Dixon price	$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{D} i(2x_i^2 - x_i - 1)^2$	[−10, 10]	0
f_5	Zakharov	$f(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	[−5.12, 5.12]	0

Table 3 Characteristics of group II multimodal test functions

Function	Name	Formulation	Domain	F_{opt}
f_6	Rosenbrock	$f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	[−30, 30]	0
f_7	Rastrigin	$f(x) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i)\right)$	[−5.12 5.12]	0
f_8	Griewank	$f(x) = \frac{1}{4000}\left(\sum_{i=1}^{D} x_i^2\right) - \left(\prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)$	[−600, 600]	0
f_9	Shubert	$f(x) = \prod_{i=1}^{D}\left(\sum_{j=1}^{5} \cos((j+1)x_i + j)\right)$	[−10, 10]	−186.7309
f_{10}	Penalized	$f(x) = 0.1 \times \{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2\left[1 + \sin^2(2\pi x_D)\right]\} + \sum_{i=1}^{D} u(x_i, a, k, m)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$ $(a = 5, k = 100, m = 4)$	[−50, 50]	0

Table 4 Characteristics of group III fixed-dimension test functions

Function	Name	Formulation	Domain	Dimension	F_{opt}
f_{11}	Schaffer	$f(x) = 0.5 + \frac{sin^2(x_1^2 + x_2^2) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	[−100 100]	2	0
f_{12}	Six hump camel back	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	[−5, 5]	2	−1.0316
f_{13}	Bohachevsky2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	[−100, 100]	2	0
f_{14}	Powell	$f(x) = \sum_{i=1}^{D/4} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$	[−4, 5]	24	0
f_{15}	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	[−10, 10]	2	0

coordinates (0, 0). The best tree at each iteration is shown with a red star marker and the remaining trees with a green star marker. The algorithm starts with a population of 20 trees. Initially, trees are scattered in the solution space. A tree that attained the best fitness is identified and considered as the best tree. The algorithm then repeatedly applies

Table 5 Characteristics of group IV test functions drawn from CEC2005

Function	Function	Range	F_{opt}
f_{16}	Shifted sphere function	[−100, 100]	−450
f_{17}	Schwefel's problem with Global Optimum on Bounds	[−100, 100]	−310
f_{18}	Shifted rotated Rastrigin's	[−30, 30]	−330
f_{19}	Shifted rotated Weierstrass	[−0.5, 0.5]	90
f_{20}	Hybrid composition function	[−5, 5]	120

growth, fruit scattering, and root spreading operators to the population. During the growth phase, the trees improve their physical condition and compete with their neighbors if necessary. At this stage, the weak trees are replaced by the stronger ones. In the fruit scattering step, some of the fittest trees disperse their fruits and seeds in the solution space. This issue increases the exploration ability of the algorithm. In the rooting stage, the best tree in the population exploits its surroundings to find a better solution, At each iteration of the algorithm, trees close to the best tree. In this example, the fruit scattering rate that controls the exploration rate is set to 0.2, which is why we see that the rate of exploitation is higher than exploration. If the fruit scattering rate is higher, then the population diversity will increase, and the algorithm will converge later. From plots, it is clear that trees explore the solution space at the beginning iterations, and as the algorithm proceeds, the trees exploit around the best tree and converge to the global minimum point.

3 Experimental results and discussions

3.1 Benchmark functions

The performance of the proposed algorithm is tested on five types of well-studied classic and modern numerical optimization functions.

– **Group I:** F_1–F_5 are unimodal functions. These functions are used to test the exploitation ability and fast converging performance of the algorithms.

Table 7 Parameter setting for comparison algorithms

Algorithm	Control parameters
GA [16]	$P_c = 0.8$, $P_m = 0.1$, selection type= tournament, crossover method= arithmetic
PSO [6]	$c_1 = 2, c_2 = 2, \omega = 0.2$
GWO [12]	$r_1, r_2 = \text{rand}[0, 1], a = \text{rand}[0, 2], C = 2r_2, A = 2ar_1 - a$
WOA [13]	$a \in [0, 2], a_2 \in [-1, -2]$
SSA [14]	$N_{fs} = 4, G_c = 1.9, P_{dp} = 0.1$
LFD [46]	Threshold = 2, CSV = 0.5, $\beta = 1.5, \alpha_1 = 10, \alpha_2 = 0.00005, \alpha_3 = 0.005, \partial_1 = 0.9, \partial_2 = 0.1$
SO [47]	$p_r = p_c = p_s = p_w \in \text{rand}[0.4, 0.6]$
HTS	$c \in (0, 1)$, Chaotic map = Gaussian

Table 8 Average and final ranking of the HTS algorithms with different chaotic maps on 30 dimension $F_1 - F_{25}$ problems

Chaotic map	Average rank	Final rank
Chebyshev	4.52	2
Circle	5.72	6
Gaussian	3.80	1
Iterative	5.70	5
Logistic	4.82	3
Piecewise	6.30	8
Sine	5.26	4
Singer	6.56	10
Sinusoidal	6.50	9
Tent	5.82	7

– **Group II:** F_6–F_{10} are multimodal functions. These functions have many local optima in the function landscape. In the search process, an algorithm may get trapped in local optimums and diverted in a direction far away from the optimum point. Multimodal functions are used to evaluate the exploration and local optimum avoidance capabilities of algorithms.
– **Group III:** F_{11}–F_{15} are fixed-dimension functions. This group of functions is used to test the ability of algorithms to avoid premature convergence and jump out of local

Table 6 Characteristics of group V test functions drawn from CEC2018

Function	Name	Range	F_{opt}
f_{21}	Shifted and Rotated Rastrigin's Function	[−100, 100]	400
f_{22}	Shifted and Rotated Lunacek Bi Rastrigin Function	[−100, 100]	600
f_{23}	Shifted and Rotated Schwefel's Function	[−100, 100]	900
f_{24}	Hybrid Function 2 ($K = 3$)	[−100, 100]	1100
f_{25}	Composition Function 1 ($K = 3$)	[−100, 100]	2000

Table 9 Average and final ranking of the HTS algorithms with different chaotic maps on $F_1 - F_{10}$ functions in different dimension

Chaotic map	100D		500D		1000D	
	Average rank	Final rank	Average rank	Final rank	Average rank	Final rank
Chebyshev	4.85	3	4.75	2	4.5	3
Circle	5.75	5	5.6	5	6	7
Gaussian	2.75	**1**	3.4	**1**	3.76	2
Iterative	6.45	9	6.3	9	5.8	6
Logistic	4.6	2	5.35	3	3.7	**1**
Piecewise	7.1	10	5.8	6	6.6	10
Sine	6.2	8	5.95	7	5.4	4
Singer	6.1	6	6.36	10	6.1	8
Sinusoidal	6.15	7	5.45	4	5.6	5
Tent	5.05	4	6.1	8	6.2	9

Table 10 Statistical results for 30-dimension group I unimodal test problems

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_1	Mean	1.02E+04	1.08E+05	3.87E−19	3.06E−08	4.15E−31	8.14E−15	1.11E−93	**2.18E−203**
	Std	3.19E+03	2.11E+04	1.15E−18	6.12E−09	6.31E−25	2.21E−09	9.63E−87	**6.97E−197**
	AET	11.86	5.98	**2.13**	8.05	2.71	7.05	5.28	3.96
F_2	Mean	7.72E+01	1.19E+02	**0.00E+00**	2.27E−08	3.55E−164	6.13E−13	8.35E−50	2.26E−172
	Std	6.84E+01	1.05E+02	**0.00E+00**	3.91E−07	4.18E−152	5.31E−10	2.19E−22	9.04E−155
	AET	8.91	8.15	**3.49**	8.82	3.8	11.09	7.55	5.36
F_3	Mean	2.33E+00	7.19E+00	3.12E−05	2.53E−01	3.73E−04	3.60E−01	3.78E−04	**7.14E−05**
	Std	1.67E+00	8.90E−01	1.72E−04	1.99E−01	4.48E−04	5.56E−01	**4.19E−05**	4.15E−04
	AET	20.89	19.49	16.9	23.97	17.37	15.57	22.65	**13.6**
F_4	Mean	1.42E+00	1.10E+02	7.37E−01	1.71E−01	6.67E−01	6.67E−01	6.76E−01	**8.76E−04**
	Std	9.74E−01	2.21E+01	3.10E−01	9.76E−02	**0.00E+00**	1.40E−29	1.74E−01	2.56E−03
	AET	29.42	25.34	20.16	33.81	19.48	28.64	16.49	**11.61**
F_5	Mean	1.13E−03	1.99E−147	6.74E−210	4.11E−09	1.49E+02	1.82E−08	4.23E+00	**0.00E+00**
	Std	6.47E−03	5.66E−107	6.92E−208	1.55E−08	2.11E+02	6.03E−09	2.34E+00	**0.00E+00**
	AET	10.88	9.22	6.22	8.96	5.54	8.32	25.12	**4.49**

optimum points when solving low dimension unimodal and multimodal problems.

- **Group IV:** F_{16}–F_{20} are shifted and composition functions drawn from IEEE CEC 2005 [43] and CEC 2014 [44] competitions on single objective real-parameter numerical optimization problems. These functions are used to assess the performance of algorithms in balancing exploitation and exploration.
- **Group V:** F_{21}–F_{25} are modern optimization functions considered from CEC 2018 test suite. These problems are suitable test cases to evaluate the reliability and accuracy of algorithms because they are more difficult and complex. Detailed information about these test cases is given in [45].

Tables 2, 3, 4, 5, 6 summarize the characteristics of the test functions, where F_{opt} shows the global optimum of the functions, and Domain shows the upper and lower bounds of search space. In Table 6, K indicates the number of functions in the test problem F_i.

3.2 Comparison algorithms

The proposed HTS is compared with seven optimization algorithms namely GA [16], PSO [6], GWO [12], WOA [13], SSA [14], Lévy flight distribution (LFD) [46], and seasons optimization (SO) [47] algorithm. These algorithms cover the recently introduced optimizers like SSA, LFD and SO, and widely used techniques such as GA, PSO, GWO, and WOA. Note that a large number of optimization metaheuristics is introduced in recent years. Some of these algorithms are an enhanced version of the basic algorithm, while others are developed recently based on a new phenomenon.

Table 11 Statistical results for 30-dimension group II multi-modal test problems

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_6	Mean	8.15E+00	4.02E+02	2.62E+01	8.32E−01	2.49E+01	1.76E+01	**5.26E−10**	2.12E−09
	Std	4.33E+00	4.66E+02	2.79E+01	2.65E+00	2.51E+01	2.14+01	**6.12E−10**	2.68E−07
	AET	35.2	34.64	**11.75**	50.13	15.87	56.41	53.51	21.67
F_7	Mean	5.74E+01	1.55E+01	0.00E+00	5.57E−09	0.00E+00	1.21E−08	5.02E−73	0.00E+00
	Std	3.21E+01	9.56E+00	0.00E+00	2.33E−07	0.00E+00	3.21E−07	3.09E−15	0.00E+00
	AET	26.54	15.64	5.15	21.73	**4.96**	18.02	24.95	7.13
F_8	Mean	2.56E−02	2.87E−01	0.00E+00	6.54E−06	0.00E+00	9.15E−22	0.00E+00	0.00E+00
	Std	9.14E−03	4.86E−03	0.00E+00	4.03E−06	0.00E+00	1.12E−14	0.00E+00	0.00E+00
	AET	7.73	7.35	5.12	5.06	**4.62**	8.02	9.42	5.08
F_9	Mean	− 1.13E+02	− 1.80E+02	-1.87E+02	− 1.87E+02	− 1.87E+02	− 1.87E+02	− 1.87E+02	− 1.87E+02
	Std	9.17E−01	5.15E−08	2.50E−07	3.58E−16	2.00E−15	6.32E−17	5.75E−05	1.37E−34
	AET	33.09	27.65	**13.74**	18.92	14.32	20.43	38.1	16.04
F_{10}	Mean	8.64E+00	2.00E−01	7.04E−01	6.47E−04	1.11E−02	2.97E+00	1.17E−04	**1.74E−5**
	Std	1.67E+00	7.03E−01	1.86E−02	4.36E−03	3.05E−02	8.14E−08	5.19E−05	**1.61E−13**
	AET	25.07	18.99	23.71	26.48	19.44	15.16	**11.40**	11.77

Table 12 Statistical results for group III fixed-dimension test problems

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_{11}	Mean	2.01E−12	3.48E−10	0.00E+00	2.36E−05	0.00E+00	2.22E−15	0.00E+00	0.00E+00
	Std	5.68E−13	1.32E−11	0.00E+00	7.02E−17	0.00E+00	1.10E−22	0.00E+00	0.00E+00
	AET	8.28	7.79	4.94	5.19	4.84	9.09	11.56	**4.76**
F_{12}	Mean	− 2.33E−01	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	**-1.03E+00**
	Std	1.45E−07	9.81E−03	7.57E−06	8.24E−09	5.70E−09	2.13E−12	1.19E−39	**0.00E+00**
	AET	7.58	7.43	4.33	5.09	6.57	6.37	6.38	**3.95**
F_{13}	Mean	1.56E−05	8.32E−08	2.32E−07	2.35E−07	1.25E−07	1.58E−09	5.47E−43	**2.60E−166**
	Std	8.30E−06	1.38E−07	3.54E−07	3.19E−07	1.22E−07	1.17E−09	8.11E−53	**0.00E+00**
	AET	4.13	3.68	2.95	3.15	2.77	5.04	6.11	**2.67**
F_{14}	Mean	2.97E−12	1.78E−13	1.15E−98	9.34E−19	3.61E−19	1.44E−19	1.68E−21	**8.54E−132**
	Std	2.52E−12	1.02E−13	3.49E−98	2.82E−18	1.02E−18	2.16E−22	6.41E−55	**2.16E−131**
	AET	39.44	39.46	7.64	11.64	6.97	15.47	**4.15**	4.65
F_{15}	Mean	2.62E−11	2.66E−13	9.54E−96	6.98E−20	1.90E−22	7.26E−46	7.50E−31	**1.57E−109**
	Std	6.91E−12	8.39E−15	0.00E+00	2.21E−19	3.27E−22	7.87E−44	1.65E−23	**5.00E−106**
	AET	6.53	6.48	2.99	4.01	**2.29**	5.6	3.55	2.87

The employed test suite and experimental setup are not identical in all papers. Thus, in this work, the basic version of the algorithms are used in experiments.

Besides these counterparts, we also compare the HTS with other nature-inspired algorithms, namely, flower pollination algorithm (FPA) [24], forest optimization algorithm (FOA) [24], and cuckoo search (CS) algorithm [48]. As their overall performances in terms of fitness values have more differences with HTS and the other algorithms, the detailed simulation results are not presented in this paper.

3.3 Performance measures

Two statistical measures are used to evaluate the performance of algorithms: the mean and the standard deviation of the objective function found over 51 independent runs according to the instructions provided in [49]. The non-parametric Friedan test, contrast estimation, and Wilcoxon statistical test at 95% significance level is carried out to expose the significant difference between the obtained results by comparison algorithms.

Table 13 Statistical results for 30-dimension group IV CEC2005 test problems

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_{16}	Mean	− 4.29E+02	− 4.49E+02	− 2.81E+02	− 4.10E+02	− 4.50E+02	− 2.32E+02	− 4.50E+02	− 4.50E+02
	Std	3.15E+01	1.34E+01	3.61E+01	7.45E+00	3.14E+00	1.40E− 09	0.00E+00	0.00E+00
	AET	94.25	70.68	60.53	53.22	58.86	61.93	58.72	**46.14**
F_{17}	Mean	− 2.34E+02	− 1.44E+02	− 2.89E+02	− 3.01E+02	− 3.09E+02	6.12E+02	− 3.05E+02	**− 3.10E+02**
	Std	9.54E+00	1.38E+01	2.47E+00	3.60E+00	1.94E+00	8.83E+01	1.60E+00	**0.00E+00**
	AET	110.33	62.45	93.54	56.14	**55.31**	110.42	125.64	56.28
F_{18}	Mean	− 1.92E+02	− 3.12E+02	− 2.94E+02	− 3.20E+02	− 3.19E+02	− 2.08E+02	− 3.24E+02	**− 3.25E+02**
	Std	3.64E+01	1.14E+01	**1.07E+01**	1.64E+01	9.57E+00	2.31E+01	3.55E+01	1.65E+01
	AET	119.65	129.69	99.47	90.48	71.46	98.56	125.47	**68.4**
F_{19}	Mean	9.96E+01	9.64E+01	9.84E+01	9.63E+01	9.47E+01	9.78E+01	9.46E+01	**9.31E+01**
	Std	4.50E+00	1.55E+01	1.65E+01	5.40E+00	3.60E+00	**2.21E+00**	4.34E+00	6.90E+00
	AET	257.15	169.7	297.25	**122.95**	277.97	157.83	157.35	135.43
F_{20}	Mean	5.13E+02	4.04E+02	4.47E+02	3.97E+02	2.92E+02	5.11E+02	**1.61E+02**	2.15E+02
	Std	4.27E+00	6.64E+01	**9.84E+01**	3.64E+01	1.68E+01	4.97E+01	1.10E+02	9.53E+01
	AET	172.31	204.31	268.5	221.85	267.52	180.69	165.16	**160.38**

Table 14 Statistical results for 10-dimension group V CEC2018 test problems

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_{21}	Mean	4.20E+02	4.36E+02	4.11E+02	4.07E+02	4.10E+02	4.16E+02	4.00E+02	4.00E+02
	Std	2.69E+00	5.57E+00	2.60E+00	3.45E+00	6.00E−01	7.31E+00	0.00E+00	0.00E+00
	AET	126.52	269.12	210.38	145.88	197.56	117.65	119.64	**96.42**
F_{22}	Mean	6.39E+02	6.28E+02	6.09E+02	6.09E+02	6.43E+02	6.05E+02	6.17E+02	**6.02E+02**
	Std	1.35E+01	3.77E+00	2.15E+01	6.73E+00	1.90E+01	4.11E+00	2.64E+00	**2.40E+00**
	AET	203.73	192.35	176.81	187.05	193.1	192.65	144.18	**110.29**
F_{23}	Mean	9.37E+02	9.66E+02	9.07E+02	9.11E+02	9.28E+02	1.57E+03	9.01E+02	**9.00E+02**
	Std	3.80E+00	3.61E+01	2.17E+00	4.46E+00	3.14E+00	3.61E+00	2.23E+00	**0.00E+00**
	AET	189.53	178.64	188.96	169.82	182.16	187.65	167.71	**159.34**
F_{24}	Mean	3.02E+03	2.34E+03	1.16E+03	1.17E+03	1.21E+03	1.13E+03	1.15E+03	**1.11E+03**
	Std	2.03E+02	3.07E+00	4.79E+00	6.16E+00	3.99E+00	2.05E+00	**1.50E+00**	2.88E+00
	AET	108.23	84.96	**61.32**	85.33	94.02	115.1	116.81	79.32
F_{25}	Mean	2.33E+03	2.30E+03	2.17E+03	2.15E+03	2.07E+03	**2.05E+03**	2.16E+03	2.09E+03
	Std	1.54E+01	1.16E+01	6.54E+00	3.65E+01	5.70E+00	**2.61E+00**	6.14E+00	5.38E+00
	AET	296.55	177.19	205.32	218.44	216.14	211.59	176.28	**138.46**

3.4 Experimental setup

All algorithms were coded using Matlab 2016b on a computer with 8 GB RAM, 2.2 GHz CPU, and Windows 7 OS. The source codes of algorithms are collected from various sources published by their authors and customized to be compatible with our experimental configuration. Population size for all algorithms set to 25. The algorithms are compared in terms of the average performance obtained over 51 independent runs on each test function, each time with a different initial population. In each run, the maximum number of $10^3 \times D$ function evaluations (M_f) is considered as the termination condition, where D indicates the dimension of the test function. The quality of solutions reported by the algorithms is calculated by the Mean, the standard deviation (Std), and average execution time (AET) measures. The parameters of counterpart algorithms are configured according to the recommended settings in their original papers. The used parameters in simulations are summarized in Table 7.

To identify which chaotic map in the root spreading phase provides better performance for HTS, we tested the algorithm with ten chaotic maps given in Table 1. Table 8 reports the average rank obtained by HTS with different chaotic maps on total 25 benchmark functions in 30 dimensions. The rank of algorithms is computed by Friedman test [50]. The results reveal that HTS with a Gaussian chaotic map

Table 15 Rankings of the algorithms computed by Friedman test

Function	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
$F_1 - F_5$	Average rank	6.8	7	2.8	4.6	4.1	4.9	4.4	1.4
	Final rank	7	8	2	5	3	5	4	**1**
$F_6 - F_{10}$	Average rank	7	7	4.2	4.1	3.6	5.3	2.6	2.2
	Final rank	6	6	5	4	3	6	2	**1**
$F_{11} - F_{15}$	Average rank	7.6	5.8	3.3	6.2	4.3	3.8	3.1	1.9
	Final rank	8	6	3	7	5	4	2	**1**
$F_{16} - F_{20}$	Average rank	7	5.2	6.2	4.2	2.8	7.2	2	1.4
	Final rank	7	5	6	4	3	8	2	**1**
$F_{21} - F_{25}$	Average rank	7.2	7	4.3	3.9	5	3.8	3.3	1.5
	Final rank	8	7	5	4	6	3	2	**1**
$F_1 - F_{25}$	Average rank	7.12	6.4	4.16	4.6	3.96	5	3.08	1.68
	Final rank	8	7	4	5	3	6	2	**1**

Table 16 Unadjusted p values where HTS is the control algorithm

Algorithm	Unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
GA	4.096E−15	2.87E−14	2.87E−14	2.87E−14	2.87E−14
PSO	9.58E−12	6.70E−11	5.75E−11	5.75E−11	5.75E−11
GWO	3.44E−4	2.41E−3	1.03E−3	1.03E−3	1.03E−3
SSA	2.50E−5	1.75E−4	1.00E−4	1.00E−4	1.00E−4
WOA	9.99E−4	6.99E−3	2.00E−3	2.00E−3	2.00E−3
LFD	1.65E−6	1.16E−5	8.26E−6	8.26E−6	8.26E−6
SO	4.30E−2	3.3E−1	4.33E−2	4.33E−2	4.33E−2

Table 17 Adjusted p values where HTS is the control algorithm

Algorithm	Adjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
GA	1.30E−4	9.10E−4	7.80E−4	7.80E−4	7.80E−4
PSO	8.71E−6	6.10E−5	6.10E−5	6.10E−5	6.10E−5
GWO	9.93E−2	6.95E−1	3.97E−1	3.97E−1	3.97E−1
SSA	3.66E−1	2.56	1.10	7.63E−1	7.32E−1
WOA	4.19E−1	2.94	1.10	7.63E−1	7.63E−1
LFD	7.88E−3	5.52E−2	3.94E−2	3.94E−2	3.94E−2
SO	7.63E−1	5.34	1.10	7.63	7.63

Table 18 Contrast estimation between HTS and its counterparts

	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
GA	0.00E+00	− 2.75E+00	− 1.02E+01	− 1.03E+01	− 1.03E+01	− 8.56E+00	− 1.23E+01	− 1.27E+01
PSO	2.75E+00	0.00E+00	− 7.40E+00	− 7.56E+00	− 7.58E+00	− 5.81E+00	− 9.50E+00	− 9.90E+00
GWO	1.02E+01	7.40E+00	0.00E+00	− 1.55E−01	− 1.75E− 01	1.60E+00	− 2.10E+00	− 2.50E+00
SSA	1.03E+01	7.56E+00	1.55E− 01	0.00E+00	− 1.99E− 02	1.75E+00	− 1.94E+00	− 2.35E+00
WOA	1.03E+01	7.58E+00	1.75E− 01	1.99E− 02	0.00E+00	1.77E+00	− 1.92E+00	− 2.33E+00
LFD	8.56E+00	5.81E+00	− 1.60E+00	− 1.75E+00	− 1.77E+00	0.00E+00	− 3.70E+00	− 4.10E+00
SO	1.23E+01	9.50E+00	2.10E+00	1.94E+00	1.92E+00	3.70E+00	0.00E+00	− 4.04E− 01
HTS	1.27E+01	9.90E+00	2.50E+00	2.35E+00	2.33E+00	4.10E+00	4.04E− 01	0.00E+00

Table 19 Comparison of algorithms by Wilcoxon signed-rank test with 95% significant level on test functions $F_1 - F_{25}$

HTS vs.	T+	T-	p value	Winner
GA	253	0	1.00E−05	HTS
PSO	210	0	1.00E−05	HTS
GWO	118	2	1.20E−04	HTS
SSA	231	0	1.00E−05	HTS
WOA	111	9	5.40E−04	HTS
LFD	141	12	1.20E−04	HTS
SO	78	13	9.60E−04	HTS

obtains the first rank on all functions. The second and third ranks belong to Chebyshev and Logistic maps, respectively. Table 9 shows the rank of different HTS variants on the scalable unimodal and multimodal $F_1 - F_{10}$ test functions in 100, 500, and 1000 dimensions. Results show that the HTS performs more effectively on 100 and 500 dimensions when equipped with a Gaussian map. In 1000 dimensions, the HTS with Logistic map obtains the first rank and the second rank when using the Gaussian map. The difference between the average rank of Logistic and Gaussian maps in 1000 dimensions is negligible. The results recommend selecting the Gaussian map as the final optimal map to be used for exploitation in the root spreading phase.

3.5 Numerical results

Tables 10, 11, 12, 13, 14 show the statistical results obtained by comparison algorithms. The best results are presented in boldface. Overall, HTS outperforms its counterparts in terms of statistical tests on most benchmark problems. From Tables 10, 11, 12, 13, 14, we conclude the following points:

- Except function F_2, the mean results obtained by HTS in solving unimodal problems are superior. This justifies that the HTS has good convergence speed and exploitation power. The mean results obtained in F_2 is very competitive to the best mean achieved by GWO. An investigation on the standard deviation values proves that the performance of HTS is comparable to counterpart algorithms.
- HTS, SO, WOA, and GWO obtains the best results in solving group II multi-modal functions. GWO, WOA, and SO performs well on 3 test problems. Out of 5 test problems, HTS was able to get the best mean results for 4 functions. The mean results of the F_6 function where the HTS is not the ideal is still competitive to the SO. The results on group II functions reveal that HTS can avoid local optimums and efficiently explore the solution space.
- HTS is very powerful in solving group III problems with fixed dimensions. It generates the best mean and standard deviation in solving all functions compared with its counterparts. Results confirm that the HTS is powerful enough to avoid premature convergence to local optimums in solving fixed and low dimension optimization problems.
- The results obtained on CEC2005 shifted and hybrid functions show the superior performance of HTS compared with others. The results justify that the HTS is a reliable and accurate algorithm in solving complex functions. HTS is top-performing in solving $F_{16} - F_{19}$ functions. The second and third ranks belong to SO and WOA, respectively.
- The performance of HTS in solving group V CEC2018 test problems is superior and it obtains the best mean results on 4 problems. An accurate investigation of standard deviations reveals that the HTS obtains comparable performance compared with other algorithms. The results confirm the ability of HTS in balancing exploration and exploitation in the search process.

Table 15 compares the average and final ranks of algorithms on test problems in terms of the mean results. The rank of algorithms is computed by Friedman test [50]. The comparison reveals that HTS is top-performing among other algorithms. The second and third ranks belong to the SO and WOA. The worst rank belongs to the GA. This issue confirms that introducing new algorithms is helpful in solving classic and modern optimization problems.

The comparison of unadjusted and adjusted p values [50] between counterparts and HTS are summarized in Tables 16 and 17. The results are calculated based on the mean objective values. A careful investigation on p values shows that the HTS performs well on test problems, matching or exceeding the best results achieved by other algorithms.

Table 18 shows how far the HTS algorithm outperforms its counterparts. The quantitative differences is computed by contrast estimation measure [50] according to the medians obtained over test cases. Results confirm that the HTS obtains low error rates compared with other algorithms.

Table 19 summarizes the statistical results obtained by the two-sided Wilcoxon signed-rank test with a significant level of 95% between HTS and counterpart algorithms on benchmark functions [50]. The comparison is performed based on the mean values obtained through simulation runs. The results reveal that the HTS is statistically superior to other algorithms in finding the global optimum of test problems.

3.6 Convergence test

To investigate the searching performance of the algorithms, we perform a convergence test on five candidate functions F_3, F_7, F_{11}, F_{20}, and F_{22} from each group of unimodal,

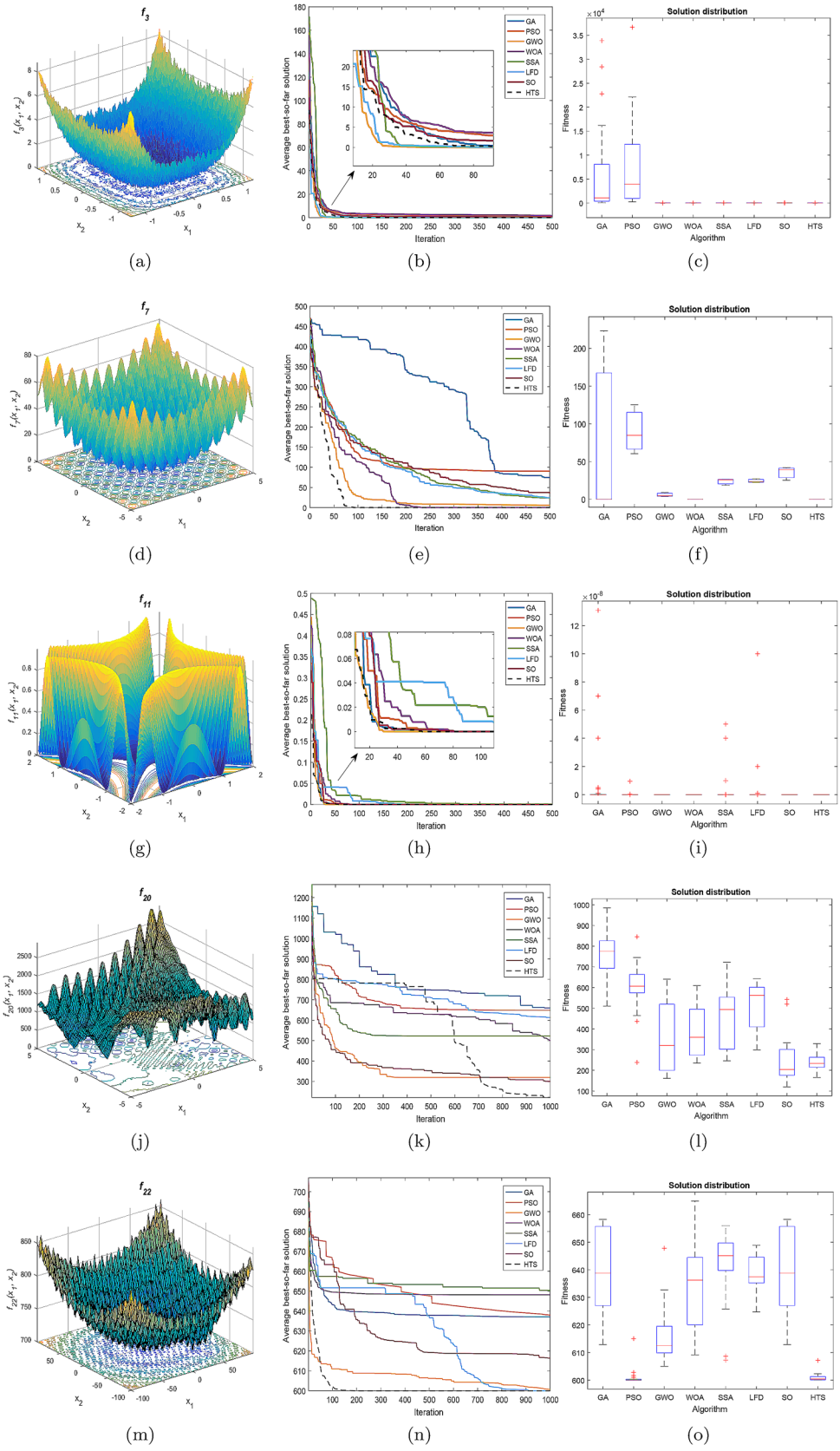


Fig. 6 Graphical representation, convergence performance and solution distribution of comparison algorithms on test functions f_3, f_7, f_{11}, f_{20}, and f_{22}

Table 20 Results of test functions $F_1 - F_{10}$ with 100 dimensions

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_1	Mean	1.20E+04	3.07E+04	1.86E−44	2.86E−53	3.44E−66	3.54E−06	2.15E−16	**1.58E−85**
	Std	3.07E+04	3.74E+03	1.26E−51	6.02E−65	2.96E−66	2.94E−07	3.14E−27	**1.19E−90**
	AET	11.42	8.85	8.36	12.53	5.31	15.02	19.25	**4.85**
F_2	Mean	1.96E+02	3.87E+02	1.96E−50	5.07E−53	1.97E−67	1.91E−06	1.23E−11	**1.69E−87**
	Std	5.55E+01	6.23E+01	2.38E−50	7.21E−34	3.41E−67	1.61E−07	2.13E−11	**2.02E−85**
	AET	10.58	9.17	10.1	8.31	7.32	15.79	13.46	**6.16**
F_3	Mean	3.65E+01	2.06E+02	2.20E−03	1.45E−02	2.60E−03	5.69E+01	**7.00E−05**	1.30E−04
	Std	2.31E+01	3.61E+01	1.30E−03	3.52E−04	1.20E−03	3.01E+01	**1.14E−05**	8.58E−04
	AET	17.57	13.31	13.35	13.64	**12.92**	31.88	19.76	14.85
F_4	Mean	1.58E+01	4.17E+02	6.67E−01	2.58E−01	6.67E−01	6.90E−01	2.54E−01	**2.34E−08**
	Std	6.54E+00	1.28E+01	1.75E−06	7.64E−02	3.17E−05	4.87E−03	7.41E−03	**1.62E−06**
	AET	12.61	9.3	8.14	13.46	**6.03**	16.61	19.55	9.06
F_5	Mean	5.17E−03	2.57E−100	3.25E−02	2.86E−02	1.63E+03	2.71E−06	4.81E+01	**1.81E−87**
	Std	1.76E−03	1.15E−59	4.22E−02	3.46E−04	2.60E+02	3.02E−06	8.33E+01	**1.94E−127**
	AET	11.86	9.57	8.38	13.41	**5.73**	17.51	22.06	11.05
F_6	Mean	2.45E+06	1.76E+07	9.76E+01	1.16E−08	9.70E+01	9.77E+01	3.27E−07	**3.98E−12**
	Std	6.00E+03	4.20E+03	8.48E−01	4.70E−03	3.90E−01	1.20E−02	5.66E−04	**2.23E−07**
	AET	10.32	8.95	8.46	10.07	**6.32**	14.86	17.33	7.63
F_7	Mean	3.26E+03	8.41E+03	3.79E−14	9.14E−10	**0.00E+00**	6.96E−06	3.22E−14	1.06E−86
	Std	2.55E−01	1.74E+00	6.56E−14	6.12E−17	**0.00E+00**	2.60E−06	5.58E−04	1.50E−54
	AET	9.21	9.85	8.54	9.13	6.2	12.08	10.86	**6.08**
F_8	Mean	3.15E+01	6.74E+03	0.00E+00	0.00E+00	0.00E+00	2.34E−06	0.00E+00	0.00E+00
	Std	2.20E+01	7.88E+01	0.00E+00	0.00E+00	0.00E+00	6.25E−08	0.00E+00	0.00E+00
	AET	11.6	11.4	9.55	10.73	6.38	10.79	9.66	**6.35**
F_9	Mean	− 1.84E+02	− 1.47E+02	− 1.87E+02	− 1.87E+02	− 1.87E+02	− 1.87E+02	− 1.87E+02	**− 1.87E+02**
	Std	3.19E−01	2.56E−01	5.20E−02	9.21E−02	3.70E−06	1.06E−10	1.25E−02	**6.10E−25**
	AET	9.8	9.02	7.86	13.2	6.38	10.2	13.15	**6.17**
F_{10}	Mean	4.18E+02	6.37E+02	3.53E−01	5.10E−05	3.50E−03	7.85E−01	**1.31E−21**	1.60E−06
	Std	2.09E+07	3.14E+08	1.31E−02	1.14E−03	6.85E−04	3.10E−02	**2.24E−21**	3.10E−08
	AET	15.57	14.42	17.03	11.6	**9.16**	14.17	11.58	9.69

multimodal, fixed-dimension, shifted and shifted, and hybrid test cases. Each algorithm is executed 51 times on each test function and the average results are computed. Figure 6 shows the graphical representation, convergence plot, and distribution of solutions for test functions. From Fig. 6, we can conclude the following results:

- Since the HTS performs exploitation and exploration simultaneously in the search process, its convergence speed in solving unimodal function f_3 is not as fast as GWO, LFD, and SSA algorithms. The fruit dispersing phase forces the algorithm to explore the search space and prevents the premature convergence to local optimums. The degree of exploration can be controlled to increase the convergence speed of HTS in solving unimodal functions.
- In solving multimodal function f_7, HTS converges faster than other algorithms. This is because HTS efficiently explores the search space and avoids local optimums. As plotted in the box-and-whisker diagram of F_7, the HTS generates the solutions with minimum distribution compared with its counterparts.
- For fixed-dimension function f_{11}, the HTS avoids premature convergence and it converges after exploring the solution space. In fixed-dimension problems, an algorithm is preferable if it explores the solution space and jump out of local optimums. The distribution of solutions obtained by HTS has minimum distribution and is comparable with GWO, WOA, and SO.
- HTS outperformed other algorithms in solving composition function f_{20}. HTS converges relatively slow at the early generations, but then very fast as it proceeds. This reveals the advantages of combining exploration

Table 21 Results of test functions $F_1 - F_{10}$ with 500 dimensions

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_1	Mean	5.15E+04	3.05E+03	1.17E−22	3.20E−08	1.44E−60	3.28E−05	1.19E−31	**1.17E−85**
	Std	3.12E+02	1.54E+02	5.93E−23	4.22E−07	2.49E−60	1.39E−06	2.07E−31	**5.22E−85**
	AET	20.09	**16.31**	21.86	34.05	19.91	24.9	27.65	18.21
F_2	Mean	3.68E+05	3.77E+04	1.69E−22	1.70E−34	2.32E−53	8.36E−05	1.28E−41	**2.12E−87**
	Std	6.04E+01	6.33E+01	1.92E−22	4.00E−32	4.02E−53	1.64E−05	7.61E−38	**1.39E−62**
	AET	21.68	17.08	24.12	34.17	**14.63**	26.21	19.8	18.99
F_3	Mean	5.30E+03	6.70E+03	4.90E−03	6.77E−02	2.40E−03	6.12E+02	2.63E−03	**2.29E−04**
	Std	5.13E+03	2.24E+03	2.10E−03	5.02E−01	1.80E−03	3.84E+02	3.75E−03	**1.24E−04**
	AET	50.2	36.76	44.51	36.3	**30.51**	68.4	47.5	50.8
F_4	Mean	2.55E+04	5.02E+04	6.67E−01	9.44E−01	6.67E−01	9.97E−01	2.50E−01	**1.43E−01**
	Std	4.16E+03	1.17E+03	**1.96E−05**	7.04E−01	8.62E−05	5.55E−04	8.19E−01	2.37E−01
	AET	22.52	20.5	22.75	36.23	**13.71**	25.39	19.3	18.78
F_5	Mean	8.00E+04	9.09E+03	3.18E+03	6.61E−04	8.49E+03	5.72E−03	4.10E+03	**6.03E−88**
	Std	6.74E+02	2.87E+02	5.32E+02	1.25E−04	9.43E+02	9.21E−03	6.57E−01	**1.04E−87**
	AET	24.7	20.19	22.17	34.91	**19.86**	29.36	22.93	24.01
F_6	Mean	3.97E+08	9.82E+08	4.98E+02	9.33E−01	9.05E+02	4.94E+02	5.33E−01	**4.51E−01**
	Std	1.51E+06	4.33E+07	3.00E−01	2.73E−01	3.88E−01	**3.76E−02**	2.78E+01	4.02E−01
	AET	21.48	**18.4**	22.82	36.12	18.67	25.46	19.33	18.61
F_7	Mean	8.06E+03	3.08E+04	1.21E−12	5.59E−90	0.00E+00	2.13E−05	5.60E−18	0.00E+00
	Std	3.41E+03	9.47E+03	5.25E−13	4.33E−88	0.00E+00	9.04E−06	6.90E−05	0.00E+00
	AET	22.08	19.29	25.61	35.15	**16.73**	26.92	21.46	20.08
F_8	Mean	5.42E+02	4.95E+02	1.11E−16	0.00E+00	0.00E+00	6.05E−06	8.85E−14	0.00E+00
	Std	6.18E+01	8.45E−01	2.84E−59	0.00E+00	0.00E+00	1.27E−06	5.98E−13	0.00E+00
	AET	31.83	28.95	29.49	36.87	26.68	36.94	29.35	**25.44**
F_9	Mean	− 1.68E+02	− 1.80E+02	− 1.87E+02	1.81E+02	− 1.87E+02	**− 1.87E+02**	1.73E+02	− 1.66E+02
	Std	2.43E+01	8.87E+00	8.46E−02	2.05E−01	2.94E−06	**2.04E−11**	2.65E+01	3.65E+01
	AET	23.04	17.94	21.8	35.19	19.79	28.7	29.43	**16.3**
F_{10}	Mean	3.66E+08	1.24E+09	9.01E−01	6.14E−01	1.32E−02	9.96E−01	**6.73E−08**	3.06E−04
	Std	3.15E+08	7.15E+08	3.70E−03	6.88E−02	4.30E−03	2.10E−03	**1.17E−07**	5.27E−04
	AET	49.81	38.02	50.00	34.81	**32.31**	64.2	57.1	53.3

and exploitation abilities in one searching process. The counterpart algorithms are trapped in local optimums.

– With regards to shifted and rotated function f_{22}, HTS performs very well and converges faster than other ones. It obtains the lowest solution distribution compared with others. The results confirm that HTS is an efficient choice for solving complex and difficult problems.

3.7 Scalability analysis

As the dimension of test problems increases, the performance of the algorithms may degrade. We test the algorithms on problems of different sizes to evaluate their scalability. In addition to 30-dimension benchmark functions, we performed a series of tests on 100, 500, and 1000 dimension functions. To fulfill this aim, the algorithms are applied on the scalable unimodal and multimodal $F_1 - F_{10}$ test functions. The statistical results obtained by algorithms are presented in Tables 20, 21, 22. The results prove that HTS obtains competitive results compared with others.

HTS generates the best mean results over 6, 5, and 4 functions in 100, 500, and 1000 dimensions, respectively. If we consider the functions that HTS obtains the similar mean results compared with other algorithms, the success numbers will increase. HTS generates the best statistical results in F_1, F_2, F_5, F_8 in all dimensions. The mean results of remaining functions where HTS is not the best performing algorithm are competitive to the best performance. This confirms the superior scalability of the HTS algorithm.

3.8 Engineering problems

In addition to numerical optimization problems, we performed a series of simulations on three engineering optimization problems to test the ability of HTS in solving practical problems.

Table 22 Results of test functions $F_1 - F_{10}$ with 1000 dimensions

Problem	Statistics	GA	PSO	GWO	SSA	WOA	LFD	SO	HTS
F_1	Mean	3.19E+02	2.11E+02	1.43E−19	1.91E−22	1.87E−51	7.18E−05	5.10E−23	**5.10E−86**
	Std	1.45E+00	3.64E+00	7.21E−20	5.14E−17	3.24E−51	1.11E−05	1.08E−24	**8.84E−86**
	AET	32.11	25.04	38.16	64.82	21.88	39.67	19.5	**17.27**
F_2	Mean	2.35E+05	1.56E+05	7.87E−19	2.09E−68	2.09E−57	4.01E−04	1.56E−34	**5.44E−88**
	Std	2.53E+03	8.68E+02	2.08E−19	7.25E−56	3.62E−57	6.77E−05	1.64E−30	**9.42E−88**
	AET	32.7	27.02	40.67	69.2	24.84	31.95	22.79	**19.36**
F_3	Mean	9.07E+04	2.34E+04	1.00E−02	9.51E−02	5.80E−03	4.86E+04	9.33E−03	**1.40E−0**4
	Std	1.29E+03	2.24E+03	4.80E−03	1.06E−03	3.90E−03	4.12E+04	2.65E−04	**8.25E−05**
	AET	91.42	64.8	105.64	71.28	57.45	138.49	59.3	**51.29**
F_4	Mean	2.14E+05	3.01E+05	**1.67E−01**	1.53E+00	6.67E−01	1.00E+00	2.50E−01	9.36E−01
	Std	1.61E+03	2.96E+03	**2.23E−06**	1.89E−01	5.52E−05	2.25E−04	5.94E−02	2.36E−02
	AET	35.68	32.65	40.05	67.97	22.84	44.8	32.55	**25.92**
F_5	Mean	1.24E+14	2.51E+11	7.84E+03	2.55E−03	1.48E+04	2.08E−01	6.15E−08	**6.06E−88**
	Std	1.75E+14	4.15E+11	5.95E+02	4.02E+00	8.70E+02	3.58E−01	4.22E−03	**1.05E−87**
	AET	38.74	33.16	38.95	51.93	**21.55**	48.75	36.23	23.67
F_6	Mean	5.16E+09	4.97E+09	9.98E+02	2.45E+00	9.91E+02	**1.55E−01**	6.75E+00	1.76E−01
	Std	3.79E+07	2.37E+06	6.35E−01	1.58E−01	1.34E−01	**7.49E−02**	2.57E−01	3.00E−01
	AET	34.88	28.95	42	68.74	24.27	42.06	29.42	**22.91**
F_7	Mean	1.06E+04	1.16E+04	3.64E−15	6.23E−59	0.00E+00	5.09E−05	2.16E−18	0.00E+00
	Std	9.96E+01	9.04E+02	0.00E+00	6.82E−55	0.00E+00	4.37E−05	6.65E−07	0.00E+00
	AET	35.72	29.01	43.26	66.75	25.54	45.17	41.2	**28.45**
F_8	Mean	2.26E+03	5.75E+03	2.96E−16	0.00E+00	0.00E+00	6.67E−06	1.16E−10	0.00E+00
	Std	3.61E+02	4.10E+02	6.41E−17	0.00E+00	0.00E+00	2.02E−06	7.64E−09	0.00E+00
	AET	54.76	48.02	52.3	69.44	**37.89**	67.59	49.58	46.17
F_9	Mean	− 1.81E+02	− 1.66E+02	− 1.87E+02	− 1.69E+02	− 1.87E+02	**− 1.87E+02**	− 1.87E+02	− 1.17E+02
	Std	5.49E+00	9.92E−03	9.60E−03	2.37E−02	3.44E−07	**2.67E−10**	1.74E+01	6.10E+01
	AET	36.8	27.32	37.96	66.43	28.74	46.05	52.31	**26.17**
F_{10}	Mean	4.66E+00	3.02E+00	1.00E+00	1.09E+00	1.19E−02	1.03E+00	**5.98E−06**	3.24E−05
	Std	7.98E−01	1.59E−01	1.14E−02	2.01E−01	1.50E−03	8.70E−03	9.41E−03	**3.26E−05**
	AET	91.37	66.23	83.08	63.93	49.79	122.57	69.46	**61.07**

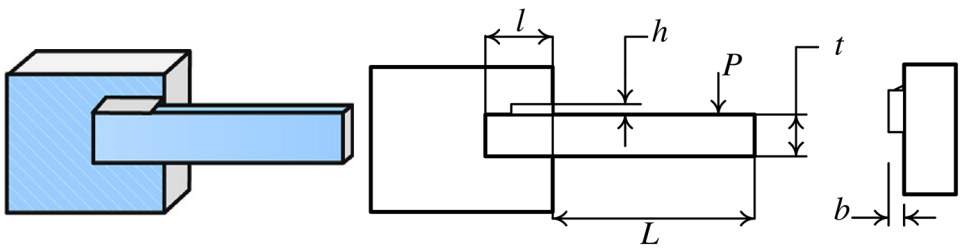


Fig. 7 A schematic plot of welded beam [51]

Table 23 The optimal values obtained by HTS and competitive algorithms for welded design problem

Algorithm	Problem parameters				Optimum value
	h	*l*	*t*	*b*	
GA	0.1252	8.4099	7.6141	0.2898	2.5244
PSO	0.2034	3.3420	9.5280	0.2034	1.7697
GWO	0.2057	3.4784	9.0368	0.2058	1.7264
SSA	0.2040	3.9101	9.0200	0.2076	1.7933
WOA	0.2054	3.4843	9.0374	0.2063	1.7305
LFD	0.1944	3.7317	9.0369	0.2057	1.7418
SO	0.2056	3.4740	9.0368	0.2058	1.7257
HTS	0.2056	3.4769	9.0366	0.2057	**1.7255**

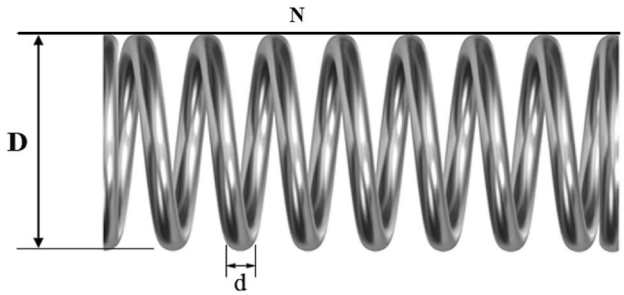


Fig. 8 Schematic of tension/compression spring

Table 24 The optimal values obtained by HTS and competitive algorithms for tension/compression spring design problem

Algorithm	Problem parameters			Optimum value
	d	*D*	*N*	
GA	0.0551	0.4451	7.5276	0.012886
PSO	0.0514	0.3577	11.6187	0.012700
GWO	0.0517	0.3567	11.2889	**0.012666**
SSA	0.0521	0.3667	10.7271	0.012669
WOA	0.0512	0.3452	12.0040	0.012676
LFD	0.0517	0.3575	11.2442	0.012700
SO	0.0508	0.3363	12.5980	0.012684
HTS	0.0517	0.3579	11.2243	0.012667

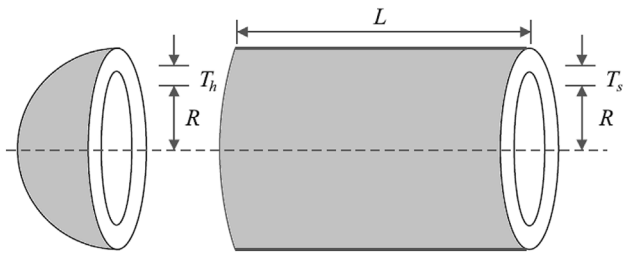


Fig. 9 Plot of pressure vessel design problem

Table 25 Comparison of results found for pressure vessel design problem

Algorithm	Problem parameters				Optimum value
	T_s	T_h	*R*	*L*	
GA	1.1568	10.4356	52.8657	131.6100	58800.0000
PSO	1.0574	0.5985	53.3912	74.0986	6810.0000
GWO	0.8125	0.4345	42.0892	176.7587	6051.5639
SSA	0.8535	0.4296	44.2076	152.3881	6062.1780
WOA	0.8125	0.4375	42.0983	176.6390	6059.7410
LFD	0.8777	0.4339	45.4755	139.0654	6080.0000
SO	0.9222	0.4559	47.7559	117.5730	6194.1751
HTS	0.8332	0.4279	43.1424	164.2261	**6045.2901**

3.8.1 Welded beam design problem

The purpose of this problem is to find the optimal manufacturing cost concerning some design constraints. Figure 7 shows the schematic of this problem. This problem is composed of 4 design parameters $\{h, l, t, b\}$ and 7 constraints $c_1 - c_7$ that are formulated as follows:

$$
\begin{aligned}
&\text{Let} \quad \vec{x} = [x_1, x_2, x_3, x_4] = [h, l, t, b] \quad \text{where} \quad 0.1 \le x_1, x_4 \le 2, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 0.1 \le x_2, x_3 \le 10, \\
&\text{Minimize} \quad f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \\
&\text{Subject to} \quad c_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0, \\
&\qquad\qquad\quad c_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \le 0, \\
&\qquad\qquad\quad c_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0, \\
&\qquad\qquad\quad c_4(\vec{x}) = x_1 - x_4 \le 0, \\
&\qquad\qquad\quad c_5(\vec{x}) = P - P_c(\vec{x}) \le 0, \\
&\qquad\qquad\quad c_6(\vec{x}) = 0.125 - x_1 \le 0, \\
&\qquad\qquad\quad c_7(\vec{x}) = 1.10471 x_1^2 + 0.0811 x_3 x_4 (14.0 + x_2) - 5.0 \le 0
\end{aligned}
\tag{21}
$$

where b is the thickness of the beam, t is the height of the beam, l is the length of the weld, and h is the thickness of the weld. The other parameters are calculated as follows:

$$
\begin{aligned}
&\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \\
&\tau' = \frac{P}{\sqrt{2}x_1 x_2}, \tau'' = \frac{MR}{J}, \ \mathrm{M} = \mathrm{P}(\mathrm{L} + \tfrac{x_2}{2}), \\
&R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2}, \\
&J = 2\{\sqrt{2}x_1 x_2[\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2]\}, \\
&\sigma(\vec{x}) = \frac{6PL}{x_4 x_3^2}, \ \delta(\vec{x}) = \frac{6PL^3}{E x_3^2 x_4}, \\
&P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}) \\
&P = 6000lb, \ \mathrm{L} = 14\mathrm{in}, \ \delta_{\max} = 0.25in, \\
&\mathrm{E} = 30 \times 10^6 psi, \ \mathrm{G} = 12 \times 10^6 psi, \\
&\tau_{\max} = 13600psi, \ \sigma_{\max} = 30000psi
\end{aligned}
\tag{22}
$$

The results obtained by HTS and counterpart algorithms on the welded beam design problem are shown in Table 23. The

results show that the HTS obtains minimum fitness value and the best setting variables compared to other algorithms.

3.8.2 Tension/compression spring design

The main purpose of this problem is to minimize the weight of a spring. A schematic plot of tension/compression spring is shown in Fig. 8. This problem is composed of 4 design variables including the number of active coils (N), mean coil diameter (D), and wire diameter (d). Mathematically this problem is formulated as follows:

$$\begin{aligned}
&\text{Let} \quad \mathbf{x} = [x_1, x_2, x_3] = [d, D, N] \quad \text{where} \quad 0.05 \le x_1 \le 2,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad 0.25 \le x_2 \le 1.3,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\quad 2 \le x_3 \le 15\\
&\text{Minimize} \quad f(\mathbf{x}) = (x_3 + 2)x_2x_1^2\\
&\text{Subject to} \quad c_1(\mathbf{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0,\\
&\qquad\qquad\quad c_2(\mathbf{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \le 0,\\
&\qquad\qquad\quad c_3(\mathbf{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0,\\
&\qquad\qquad\quad c_4(\mathbf{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0
\end{aligned} \tag{23}$$

Table 24 compares the optimal values find by HTS and comparison algorithms for the spring design problem. It is obvious that the difference between HTS and the best performing GWO algorithm is negligible. This confirms that HTS is able to find optimal values for spring design variables with minimum cost.

3.8.3 Pressure vessel design problem

The goal of this problem is to minimize the total cost (material, forming, and welding) of the cylindrical pressure vessel [13]. Figure 9 shows the plot of the pressure vessel design problem. It is composed of four parameters and constraints. The design parameters are the thickness of the head (T_h), length of the cylindrical section without the head (L), thickness of the shell (T_s), and inner radius (r). The constraints and the objective function are mathematically defined as follows:

$$\begin{aligned}
&\text{Let} \quad \mathbf{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \quad \text{where} \quad 0 \le x_1, x_2 \le 99,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 10 \le x_3, x_4 \le 200,\\
&\text{Minimize} \quad f(\mathbf{x}) = 0.6224x_1x_2x_4 + 1.778x_2x_2^3 + 3.1661x_1^2x_4 + 19.84x_1^2x_3\\
&\text{Subject to} \quad c_1(\mathbf{x}) = -x_1 + 0.0193x_3 \le 0,\\
&\qquad\qquad\quad c_2(\mathbf{x}) = -x_3 + 0.00954x_3 \le 0,\\
&\qquad\qquad\quad c_3(\mathbf{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0,\\
&\qquad\qquad\quad c_4(\mathbf{x}) = x_4 - 240 \le 0
\end{aligned} \tag{24}$$

The results found by HTS and competitive algorithms for the pressure vessel design problem are reported in Table 25.

The experimental results demonstrate the superiority of HTS compared with counterparts.

Overall, the results obtained by HTS and counterpart algorithms on real-world engineering problems show that HTS is an efficient strategy in solving complex practical problems. HTS generates results that are comparable with its counterpart algorithms. Here, we only applied the HTS on three engineering problems. One of the interesting works is to apply HTS to more practical and real-world problems to test its potentials.

3.9 Computational complexity

3.9.1 Space complexity

The space complexity of HTS is computed as follows:

- It needs $O(N \times D)$ space to store population at each iteration, where D is the dimensions of the problem under consideration, and N where N is the number of trees in the population.
- It needs $O(N)$ space to store the fitness of trees at each generation.

Overall, the space complexity of HTS in the worst case is $O(ND)$.

3.9.2 Time complexity

Each generation of the HTS is composed of five main steps: population initialization, fitness calculation, growth, fruit scattering, and root spreading. The time complexity of each phase is as follows:

- The population initialization phase costs $O(N)$.
- Fitness calculation needs the time complexity $O(N)$.
- The cost of the growth phase is $O(N^2)$ in the worst case.
- The computational cost of the fruit scattering phase is $O(N)$ in the worst case.
- The root spreading phase costs $O(D)$, where D is the problem dimension in the worst case.

Therefore, the time complexity of each generation of the algorithm in the worst case is calculated as

$$3O(N) + O(N^2) + O(D) \simeq O(N^2) \quad (25)$$

The computational complexity of the GA, PSO, GWO, WOA, SSA, LFD and SO equals $O(N^2)$. This justifies that the HTS obtains better results in the same time complexity compared with other algorithms.

4 Conclusions

This paper introduced a population-based meta-heuristic, namely hazelnut tree search (HTS) algorithm to solve various optimization problems. The proposed HTS is equipped with three operators including growth, fruit scattering, and root spreading. These operators provide a proper balance between exploration and exploitation and guide the search process. Fruit scattering helps the algorithm to jump from local optimums and explore promising areas in solution space. The growth operator and incorporating chaotic maps in the root spreading phase improve the local search performance in exploiting inside promising areas. The proposed HTS and popular counterpart algorithms are evaluated on 25 unconstrained optimization functions and three constraint engineering problems. The results demonstrate that the proposed HTS algorithm achieves competitive results compared with other algorithms in terms of solution quality and finding global optimum solutions.

In this paper, a simple basic version of HTS is designed with three operators. One of the future directions is to design complex and efficient operators to model competition and cooperation among trees as well as modeling various environmental phenomena. Additionally, the multi-objective version of the HTS can be designed. Another interesting work is to investigate the algorithm capability in solving more practical tasks in different fields. Also, we believe that the HTS can be refined in competition and fruit scattering phases for better computational performance.

References

1. Emami H, Derakhshan F (2015) Election algorithm: a new socio-politically inspired strategy. AI Commun 28(3):591–603
2. Beheshti Z, Mariyam S, Shamsuddin H (2013) A review of population-based meta-heuristic algorithms. Int J Adv Soft Comput Appl 5(1):1–35
3. Sotoudeh-anvari A, Hafezalkotob A (2018) A bibliography of metaheuristics-review from 2009 to 2015. Int J Knowl Intell Eng Syst 22:83–95
4. Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
5. Das P, Das DK, Dey S (2018) A new class topper optimization algorithm with an application to data clustering. IEEE Trans Emerg Top Comput 6750:1–11
6. Thangaraj R, Pant M, Abraham A, Bouvry P (2011) Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl Math Comput 217(12):5208–5226
7. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1:28–39
8. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
9. Yang X (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio Inspired Comput 2(2):78–84
10. Gandomia AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
11. Yu JJQ, Li VOK (2015) A social spider algorithm for global optimization. Appl Soft Comput J 30:614–627
12. Mirjalili S, Mohammad S, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
13. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
14. Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175
15. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
16. Kirkpatrick S, Vecchi GCD, Science MP (1983) Optimization by simulated annealing. Science 220:671–680
17. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inform Sci 179(13):2232–48
18. Tayarani M, Akbarzadeh M (2014) Magnetic-inspired optimization algorithms: operators and structures. Swarm Evol Comput 19:82–101
19. Pereira J, Francisco MB, Diniz CA, Oliver GA, Cunha SS, Gomes GF (2021) Lichtenberg algorithm: a novel hybrid physics-based meta-heuristic for global optimization. Expert Syst Appl 170(2021):114522
20. Pereira J, Francisco MB, Cunha SS, Gomes GF (2021) A powerful lichtenberg optimization algorithm: a damage identification case study. Eng Appl Artif Intell 97:104055
21. Haupt RL, Haupt SE (2004) Practical genetic algorithms. Wiley
22. Merrikh-Bayat F (2015) The runner-root algorithm: a metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. Appl Soft Comput J 33:292–303
23. Ghaemia M, Feizi-Derakhshi MR (2014) Forest optimization algorithm. Expert Syst Appl 41(15):6676–6687
24. Al-Betar MA, Awadallah MA, Abu Doush I, Hammouri AI, Mafarja M, Alyasseri ZAA (2012) Flower pollination algorithm for global optimization. Int Conf Unconvent Comput Natl Comput 2012:240–249
25. Shayanfar H, Soleimanian F (2018) Farmland fertility: a new metaheuristic algorithm for solving continuous optimization problems. Appl Soft Comput J 71:728–746
26. Gomes GF, Da Cunha SS, Ancelotti AC (2019) A sunflower optimization (SFO) algorithm applied to damage identification on laminated composite plates. Eng Comput 35(2):619–626
27. Gomes GF, Almeida FA (2020) Tuning metaheuristic algorithms using mixture design: application of sunflower optimization for structural damage identification. Adv Eng Softw 149:102877
28. Khan, M. S., ul Hassan, C. A., Sadiq, H. A., Ali, I., Rauf, A., & Javaid, N. (2017, August). A new meta-heuristic optimization algorithm inspired from strawberry plant for demand side management in smart grid. In International Conference on Intelligent Networking and Collaborative Systems (pp. 143-154). Springer, Cham.
29. Mirjalili S, Gandomi AH, Zahra S, Saremi S (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:1–29

30. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
31. Hosseinalipour A, Soleimanian F, Masdari M, Khademi A (2021) A novel binary farmland fertility algorithm for feature selection in analysis of the text psychology. Appl Intell: 1–36
32. Barshandeh S, Haghzadeh M (2020) A new hybrid chaotic atom search optimization based on tree-seed algorithm and Levy flight for solving optimization problems. Eng. Comput: 1-44
33. Barshandeh S, Piri F, Sangani, SR (2020) HMPA: an innovative hybrid multi-population algorithm based on artificial ecosystem-based and Harris Hawks optimization algorithms for engineering problems. Eng Comput: 1–45
34. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
35. Sharma A, Sharma A, Panigrahi BK, Kiran D, Kumar R (2016) Ageist spider monkey optimization algorithm. Swarm Evol Comput 28:58–77
36. Contreras MA, Affleck D, Chung W (2011) Evaluating tree competition indices as predictors of basal area increment in western Montana forests. For Ecol Manage 262(11):1939–1949
37. Das A, Battles J, Stephenson NL, Van Mantgem PJ (2011) The contribution of competition to tree mortality in old-growth coniferous forests. For Ecol Manage 261(7):1203–1213
38. Cain ML, Milligan BG, Strand AE (2000) Long-distance seed dispersal in plant populations. Am J Bot 87(9):1217–1227
39. Howe HF, Smallwood J (1982) Ecology of seed dispersal. Annu Rev Ecol Syst 13:85–110
40. Zeide B (1993) Analysis of growth equations. For Sci 39(3):594–616
41. Yi L, Li H, Guo J, Deussen O, Zhang X (2018) Tree growth modeling constrained by growth equations. Comput Graph Forum 37(1):239–253
42. Talatahari S, Farahmand Azar B, Sheikholeslami R, Gandomi AH (2012) Imperialist competitive algorithm combined with chaos for global optimization. Commun Nonlinear Sci Numer Simul 17(1):1312–1319
43. Suganthan P, Hansen N, Liang J, Deb K, Chen Y, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC2005 special session on real parameter optimization. Nanyang Technological University, Tech. Rep
44. Liu B, Chen Q, Zhang Q, Liang JJ, Suganthan PN, Qu BY (2013) Problem definitions and evaluation criteria for computationally expensive single objective numerical optimization. Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, December 2013
45. Suganthan P, Ali M, Wu G, Mallipeddi R (2018) Special session and competitions on real-parameter single objective optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC), Rio de Janeiro, Brazil, Rep., Jul 2018
46. Houssein EH, Saad MR, Hashim FA, Shaban H, Hassaballah M (2020) Lévy flight distribution: a new metaheuristic algorithm for solving engineering optimization problems. Eng Appl Artif Intell 94::103731
47. Emami H (2020) Seasons optimization algorithm. Eng Comput. https://doi.org/10.1007/s00366-020-01133-5
48. Yang X, Deb S (2009) Cuckoo search via Levy flights. 2009 World Congress on Nature and Biologically Inspired Computing (NaBIC 2009). Coimbatore, India, pp 210–214
49. Chen Q, Liu B, Zhang Q, Liang J (2015) Evaluation criteria for CEC 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization. Sendai, Japan, 25–28 May
50. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
51. Heidari AA, Mirjalili SA, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Fut Gen Comput Syst 97:849–872