Expert Systems with Applications 161 (2020) 113702

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Heap-based optimizer inspired by corporate rank hierarchy for global optimization

Qamar Askari [a,b,*], Mehreen Saeed [b], Irfan Younas [b]

[a] Department of Computer Science, GIFT University, Gujranwala, Pakistan
[b] Department of Computer Science, National University of Computer and Emerging Sciences, Lahore, Pakistan

ARTICLE INFO

Article history:
Received 22 September 2019
Revised 5 June 2020
Accepted 26 June 2020
Available online 18 July 2020

Keywords:
Social optimization algorithm
Corporate hierarchy based optimization
Nature-inspired meta-heuristic
Global optimization algorithm

ABSTRACT

In an organization, a group of people working for a common goal may not achieve their goal unless they organize themselves in a hierarchy called Corporate Rank Hierarchy (CRH). This principle motivates us to map the concept of CRH to propose a new algorithm for optimization that logically arranges the search agents in a hierarchy based on their fitness. The proposed algorithm is named as heap-based optimizer (HBO) because it utilizes the heap data structure to map the concept of CRH. The mathematical model of HBO is built on three pillars: the interaction between the subordinates and their immediate boss, the interaction between the colleagues, and self-contribution of the employees. The proposed algorithm is benchmarked with 97 diverse test functions including 29 CEC-BC-2017 functions with very challenging landscapes against 7 highly-cited optimization algorithms including the winner of CEC-BC-2017 (EBO-CMAR). In the first two experiments, the exploitative and explorative behavior of HBO is evaluated by using 24 unimodal and 44 multimodal functions, respectively. It is shown through experiments and Friedman mean rank test that HBO outperforms and secures 1st rank. In the third experiment, we use 29 CEC-BC-2017 benchmark functions. According to Friedman mean rank test HBO attains 2nd position after EBO-CMAR; however, the difference in ranks of HBO and EBO-CMAR is shown to be statistically insignificant by using Bonferroni method based multiple comparison test. Moreover, it is shown through the Friedman test that the overall rank of HBO is 1st for all 97 benchmarks. In the fourth and the last experiment, the applicability on real-world problems is demonstrated by solving 3 constrained mechanical engineering optimization problems. The performance is shown to be superior or equivalent to the other algorithms, which have been used in the literature. The source code of HBO is publicly available at https://github.com/qamar-askari/HBO.

1. Introduction

Optimization is a field of research, which deals with either minimizing or maximizing certain objective function(s). Real-world optimization problems are mostly classified as NP-Hard problems which classical mathematical techniques can not solve adequately or accurately. A well-known branch of approximation algorithms is nature-inspired optimization algorithms that have tremendous properties to properly solve such complex problems of optimization. A huge amount of development in the area of nature-inspired optimization algorithms has been done over the past few decades. The researchers have mapped natural processes and phenomena on optimization algorithms from every sector of life. The optimization algorithms inspired by nature are categorized into four main groups: evolution-based algorithms, swarm-based algorithms, physics-based algorithms, and human-based algorithms. A few well-known and state-of-the-art algorithms from each branch are discussed below:

1. **Evolution-based algorithms**: These algorithms are inspired by the process of biological evolution. A group of individuals called population passes through biological operators, such as selection, reproduction and mutation, and evolves through repeated applications of these operators. One of the founding algorithms in this category is Genetic Algorithm (GA) (Holland, 1992). GA is inspired by the concept of survival of fittest from Darwin's theory of evolution. One another well-known algorithm from this category is Differential Evolution (DE) (Lampinen & Storn, 2004). DE involves three fundamental operators: mutation,

* Corresponding author at: Department of Computer Science, GIFT University, Gujranwala, Pakistan.
E-mail addresses: syedqamar@gift.edu.pk, l165502@lhr.nu.edu.pk (Q. Askari), mehreen.saeed@nu.edu.pk (M. Saeed), irfan.younas@nu.edu.pk (I. Younas).

https://doi.org/10.1016/j.eswa.2020.113702
0957-4174/

Table 1
A few human behavior-based optimization algorithms.

Algorithm	Year	Inspiration
Sports-based algorithms		
Ludo Game-based Swarm Intelligence (LGSI) (Singh et al., 2019)	2019	Ludo playing strategies
Deer Hunting Optimization Algorithm (DHOA) (Brammya et al., 2019)	2019	Deer hunting strategy of humans
Team Game Algorithm (TGA) (Mahmoodabadi et al., 2018)	2018	Games involving teams
Football game algorithm (FGA) (Fadakar & Ebrahimi, 2016)	2016	Best position finding to score a goal
World cup optimization (WCO) (Razmjooy et al., 2016)	2016	FIFA world cup competitions
Soccer league competition (SLC) algorithm (Moosavian & Roodsari, 2014)	2014	Soccer league competitions
League championship algorithm (LCA) (Kashan, 2014)	2014	Championship environment
Colonization-based algorithms		
Anarchic society optimization (ASO) (Ahmadi-Javid, 2011)	2011	Anarchic behavior in social grouping
Imperialist competitive algorithm (ICA) (Atashpaz-Gargari & Lucas, 2007)	2007	Empires formation by countries
Society and civilization optimization (SCO) (Ray & Liew, 2003)	2003	Intra and inter-society interactions
Social-interaction based algorithms		
Poor and Rich Optimization (PRO) (Moosavi & Bardsiri, 2019)	2019	Achieving wealth and improve economically
Expectation Algorithm (ExA) (Shastri et al., 2019)	2019	Society individuals ass problem variables
Social Media Inspired Algorithm (SMIA) (Crawford et al., 2019)	2019	Interaction on social media
Supply–Demand-Based Optimization (SDO) (Zhao et al., 2019)	2019	Relation of supplier and consumer
Nomadic People Optimizer (NPO) (Salih & Alsewari, 2019)	2019	Behavior of nomadic people
Social Mimic Optimization (SMO) (Balochian & Baloochian, 2019)	2019	Assimilation to famous people
Socio-evolution and learning optimization (SELO) (Kumar et al., 2018)	2017	Families interaction
Social group optimization (SGO) (Satapathy & Naik, 2016)	2016	Complex problem solving social behavior
Social based algorithm (SBA) (Ramezani & Lotfi, 2013)	2013	Social interaction
Teaching learning based optimization (TLBO) (Rao et al., 2011)	2012	Classroom interaction
Group leader optimization algorithm (GLO) (Daskin & Kais, 2011)	2011	Influence of group leaders on group members
Social emotional optimization algorithm (SEOA) (Xu et al., 2010)	2010	Emotion guided behavior
Politics-based algorithms		
Political Optimizer (PO) (Askari et al., 2020)	2020	Multi-party political system
Greedy politics optimization (GPO) (Melvix, 2014)	2014	Political strategies adopted by politicians
Election campaign algorithm (ECA) (Lv et al., 2010)	2010	Election campaign by candidates
Parliamentary optimization algorithm (POA) (Borji, 2007)	2007	Competition for parliamentary head elections

recombination and selection. A few more algorithms from this category are: Evolutionary Strategy (ES) (Huning, 1976), Genetic Programming (GP) (Koza, 1992), Fast Evolutionary Programming (FEP) (Yao, Liu, & Lin, 1999), Memetic Algorithm (MA) (Moscato, 1989) and Biogeography Based Optimization (BBO) (Simon, 2008).

2. **Swarm-based algorithms**: These algorithms are normally inspired by the collective intelligent behavior of living organisms. Living creatures interact with each other in the real world to achieve optimal collective behavior. One of the founding algorithms in this branch is Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995). PSO is inspired by the food foraging behavior of flocks of the birds. The Ant Colony Optimization (ACO) (Dorigo & Di Caro, 1999) is another key algorithm in this category. ACO is inspired by the path seeking behavior of ants from their colony to the food source. A few well-known algorithms from this category are Bat Algorithm (BA) (Yang, 2010), Cuckoo Search (CS) (Yang & Deb, 2009), Firefly Algorithm (FA) (Yang, 2009), Artificial Bee Colony (ABC) (Karaboga & Basturk, 2007), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), Ant Lion Optimizer (ALO) (Mirjalili, 2015), Krill Herd (KH) (Gandomi & Alavi, 2012) and Grey-Wolf Optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014).
3. **Physics-based algorithms**: The researchers have also modelled the laws of physics that work in nature behind various phenomena to address the problems of optimization. Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983) is one of the basic algorithms in this category. SA is inspired by the metallurgical annealing process. Another very fundamental algorithm is the Quantum-inspired Genetic Algorithm (QGA) (Narayanan & Moore, 1996), which is a fusion of quantum computing and GA. A few more well-known examples are: Quantum-inspired Evolutionary Algorithm (QEA) (Han & Kim, 2002), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009), Big-Bang Crunch (BBC) (Erol & Eksin, 2006), Hysteretic Optimization (HO) (Zaránd, Pázmándi, Pál, & Zimányi, 2002), and Gravitational Interaction Optimization (GIO) (Flores, López, & Barrera, 2011).
4. **Human-based algorithms**: Human beings are regarded to be the smartest creature and they always find the best methods to fix their issues. In the literature, there are so many optimization algorithms that are influenced by human social behavior. One of the well-referred algorithms in this category of literature is Teaching–Learning Based Optimization (TLBO) (Rao, Savsani, & Vakharia, 2011). TLBO imitates student learning behavior in a classroom. A few more algorithms are Social-Based Algorithm (SBA) (Ramezani & Lotfi, 2013), Election Campaign Algorithm (ECA) (Lv et al., 2010), and Imperialist Competitive Algorithm (ICA) (Atashpaz-Gargari & Lucas, 2007).

In addition to the above state-of-the-art algorithms, dozens of meta-heuristics have been proposed in recent years. A few well-known recently proposed swarm-based meta-heuristics are Squirrel Search Algorithm (SqSA) (Jain, Singh, & Rani, 2019) inspired by the foraging behavior of flying squirrels, Seagull Optimization Algorithm (SOA) (Dhiman & Kumar, 2019) inspired by migration and attacking behavior of seagulls, Hitchcock Bird-Inspired Algorithm (HBIA) (Morais, Mourelle, & Nedjah, 2018) inspired by aggressive bird behavior portrayed by Hitchcock, Sea Lion Optimization (SLnO) algorithm (Masadeh et al., 2019) inspired by hunting behavior of sea lions, Emperor Penguins Colony (EPC) (Harifi, Khalilian, Mohammadzadeh, & Ebrahimnejad, 2019) inspired by spiral-like movement of colony of penguins, Sailfish Optimizer (SO) (Shadravan, Naji, & Bardsiri, 2019) inspired by hunting strategies of sailfishes, Bald Eagle Search (BES) (Alsattar, Zaidan, & Zaidan, 2019) inspired by hunting strategies of bald eagles, Naked-Mole Rate (NMR) (Salgotra & Singh, 2019) inspired by matting patterns of naked-mole rats, Harris Hawk Optimization (HHO) (Heidari et al., 2019) inspired by cooperative and chasing behavior of Harris' hawks, and Butterfly Optimization Algorithm (BOA) (Arora & Singh, 2018) inspired by food searching and matting behavior of butterflies. Moreover, a few recent human-based meta-heuristics are Team Game Algorithm (TGA) (Mahmoodabadi, Rasekh, & Zohari, 2018) inspired by team game strategies used in football, basketball, and volleyball, Ludo Game-based Swarm Intelligence (LGSI) (Singh, Elaziz, & Xiong, 2019) inspired by game playing rules of a board game called ludo, Poor

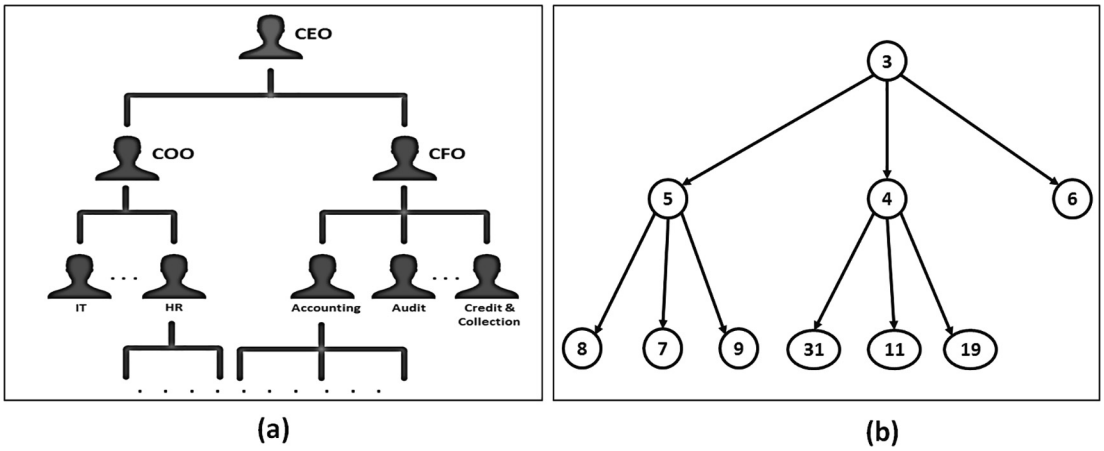


Fig. 1. Partial examples of corporate rank hierarchy and 3-ary min heap.

and Rich Optimization (PRO) (Moosavi & Bardsiri, 2019), Nomadic People Optimizer (NPO) (Salih & Alsewari, 2019) inspired by the behavior of nomadic people, and Deer Hunting Optimization Algorithm (DHOA) (Brammya et al., 2019) inspired by deer hunting strategies of human beings. Moreover, the current development in the field of optimization algorithms includes, but is not limited to, Slime Mould Algorithm (SMA) (Li, Chen, Wang, Heidari, & Mirjalili, 2020) inspired by oscillation mode of the slime in nature, Henry Gas Solubility Optimization (HGSO) (Hashim, Houssein, Mabrouk, Al-Atabany, & Mirjalili, 2019) inspired by the Henry's law, Equilibrium Optimizer (EO) (Faramarzi, Heidarinejad, Stephens, & Mirjalili, 2020) inspired by control volume mass balance models used to estimate both dynamic and equilibrium states, Chimp Optimization Algorithm (COA) (Khishe & Mosavi, 2020) inspired by the individual intelligence and sexual motivation of chimps in their group hunting, Political Optimizer (PO) (Askari, Younas, & Saeed, 2020) inspired by the multi-party political system, and Artificial Electric Field Algorithm (AEFA) (Yadav et al., 2019) inspired by Coulombs' law of electrostatic force and Newtons' law of motion.

The focus of this study is developing a novel meta-heuristic inspired by some social behavior of human-beings because the outstanding results of human behavior-based algorithms have taken the field of evolutionary and swarm intelligence to the next level. The algorithms based on human behavior can further be categorized as sport-based algorithms, human social interaction-based algorithms, colonization-based algorithms, and politics-based algorithms. A few well-known and recently proposed algorithms along with their source of inspirations from each subcategory of human behavior-based algorithms are presented in Table 1. One kind of social interaction among the human beings can be seen in the organizations where the individuals are arranged in a hierarchy called organizational chart or corporate rank hierarchy (CRH). An organization is a structured group of people because a group of individuals working for a common goal can not achieve their objective unless they organize themselves in a certain structure. This has given rise to the organization's hierarchical arrangements so that the individuals can become sufficiently efficient to achieve the organizational goals. Therefore, most organizations nowadays arrange their employees in the hierarchy of corporate officers called corporate rank hierarchy (CRH). The hierarchy of corporate ranks is a tree-like structure. The boss is appointed at the top rank (the root of the tree) and the staff are arranged in the tree-like parent–child nodes. Each parent node is considered as the head/manager and the children of that node are considered as the subordinates. It is the responsibility of each subordinate to interact and follow his/her immediate boss (the parent node). The ultimate goal of the hierarchy is accomplishing the business-related tasks in an optimal manner through interaction among the individuals. For illustration, an example of CRH is presented in Fig. 1(a). In the figure, the CEO is the boss and designated as the root node. COO and CFO are direct subordinates to the CEO: however, they are heading multiple departments. The individuals at the same level in the tree are considered as the colleagues.

The research and applications in expert and intelligent systems is one of the most flourishing sub-area of Artificial Intelligence (AI), which mimics some of the intelligent behaviors of human experts to solve real-life complex and huge problems (Zhao et al., 2019; Nabil, 2016). In certain cases, choices that need to be optimized are focused on multiple variables with enormous search spaces and meta-heuristics are the intelligent and efficient strategies for solving the optimization problems with these intractable and complex search spaces (Nabil, 2016). This motivates us to develop a novel human-behavior based meta-heuristic, which maps the concept of corporate rank hierarchy in a very distinctive manner to an optimization algorithm named as Heap Based Optimizer (HBO). In order to simulate CRH, we use the heap data structure. Heap is a non-linear tree-based data structure, mostly used for the implementation of priority queues. An example of 3 degree (3-ary) min–heap is presented in Fig. 1(b). The utilization of the heap data structure in the mapping of CRH helps to arrange the solutions in a hierarchy based on their fitness and use that arrangement in position-updating mechanism of the algorithm in a very unique way. The proposed algorithm smartly models 3 types of behavior of the employees, e.g., the interaction of subordinates with their immediate head, the interaction between the colleagues, and the self-contribution of individuals. The proposed algorithm can be adapted to solve a wide-range of engineering, industry, science and business related optimization problems arising in expert and intelligent systems. Many of the real-life resource scheduling, production planning, vehicle routing, network optimization, robotics-path planning, packaging problems and many other engineering and industry-related optimization problems of intelligent and expert systems can be solved using the proposed HBO. Moreover, this paper also contributes in the following respects:

1. To the best of our knowledge, the concept of corporate rank hierarchy has not been mapped yet. However, the arrangement of the individuals in an organizational chart allows the group of individuals to accomplish their task collectively in an optimized way. The mapping of the corporate rank hierarchy and interac-

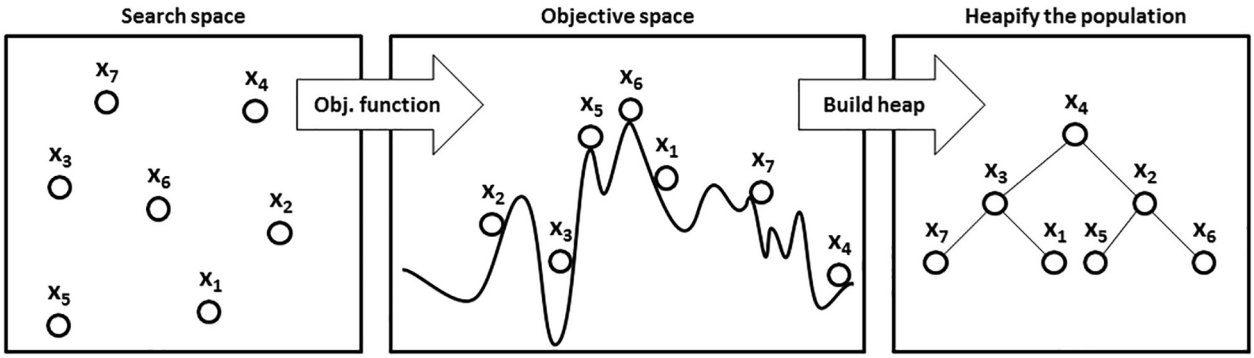


Fig. 2. An illustration of the modeling of the CRH with min–heap.

tion between the individuals in that hierarchy is a unique idea. The use of the heap data structure for mapping makes the proposed algorithm a novel approach.

2. The performance of the proposed algorithm is evaluated on 97 benchmark functions and 3 mechanical engineering optimization problems. The set of benchmarks includes unimodal functions, multimodal functions, and CEC-BC-2017 (Awad et al., 2017) test functions. The performance is compared with 7 well-known algorithms including the winner of CEC-BC-2017. It is shown through experiments and Friedman mean rank statistical test that HBO secures 1st rank for the unimodal functions as well as the multimodal functions. However, the rank of HBO for CEC-BC-2017 is 2nd after EBO-CMAR (Kumar, Misra, & Singh, 2017) (CEC-BC-2017 winner) but multiple comparison test based on Bonferroni method (Zar, 1999) shows the difference in ranks of HBO and EBO-CMAR is not statistically significant. Moreover, the overall rank of HBO is 1st for all 97 functions. Such an excellent performance on a comprehensive suite of benchmarks against well-known algorithms shows that HBO has excellent optimization capability and is expected to perform well on real-life optimization problems.
3. Another property of HBO is its simplicity. However, we anticipate that the variations in terms of improvements will be made in the future. Moreover, the mapping method encourages researchers to integrate other well-known data structures into existing or new optimization algorithms.

Section 2 describes the inspiration and mathematical modeling of CRH by using heap data structure, and Section 3 presents how the proposed algorithm can be implemented. In Section 4 the performance of proposed algorithm is evaluated by solving benchmark functions, and Section 5 asseses the applicability of proposed algorithm on mechanical engineering problems. The optimization capabilities, practical implications, and the future directions are briefly discussed in Section 6. Finally, Section 7 presents the concluding remarks.

2. Heap Based Optimizer (HBO)

In this section, we discuss the inspiration and present the mathematical model of the proposed Heap Based Optimizer (HBO).

2.1. Inspiration

The officials in a company or corporation are given corporate titles (designations). The titles define the job descriptions and responsibilities of the employees. Although the designations vary from corporation to corporation and from business to business, they are arranged in a hierarchy and are given many names such as corporate rank hierarchy (CRH), organizational chart tree, or corporate hierarchy structure, etc. A partial sample of CRH is presented in Fig. 1(a). The organizational structure is a set of methods that split the task into particular duties and coordinate it (Ahmady, Mehrpour, & Nikooravesh, 2016). The primary aim of this structure (hierarchy) is giving the formal activities an organized shape and achieving the end goals optimally. The upper levels usually represent the executives and members of the board, the middle levels represent the managers and supervisors, and the lower levels represent the workers. Individuals at the upper level are considered to be the heads/superiors for those at the lower level, individuals at the same level are considered to be the colleagues/co-workers and those who are at the lower levels are considered to be subordinated to the upper levels. The decision-making power is maintained at the higher levels in the centralized organizational structure (Jones, 2019) and individuals perform a job by making their effort, working with their colleagues, and responding to their immediate boss.

The mapping of the entire idea is split into four steps in this article: (i) Modeling the corporate rank hierarchy (CRH), (ii) mathematically modeling the interaction between the subordinates and the immediate boss, (iii) mathematically modeling the interaction between the colleagues, and (iv) self contribution of an employee to accomplish a task.

2.2. Modeling the corporate rank hierarchy

Considering the nature of CRH, we chose to model CRH with the heap data structure. By definition, the heap is a non-linear tree-shaped data structure with the following two properties: (i) Heap is a complete tree. A tree is called a complete tree if every level of the tree, except possibly the last level, is filled and all nodes in the last level are as far left as possible. (ii) In the case of min–heap, the key of every parent node is either smaller than or equal to the keys of its children and in case of max-heap, the key of every parent node is either greater than or equal to the keys of its children.

The whole CRH is considered as the population. Each formal designation is treated as the search agent. In implementation phase, a search agent corresponds to a heap node. The fitness of the search agent is considered as the key of the node in the heap and the index of the search agent in population is considered as the value of the node in the heap. The process of CRH modelling through a heap data structure is shown in Fig. 2, where x_i denotes ith search agent of the population. The curve in the objective space represents the landscape of an assumed objective function and the search agents are drawn on the fitness landscape according to their fitness. By using the fitness of each search agent as a key, a heap is constructed. As can be seen in the Fig. 2 the nodes find their positions in the heap according to their fitness. For instance, x_4 is the

best solution in the population, so is the root of the heap. It should be noted that min–heap has to be used for minimization and max-heap for maximization.

2.3. Mathematical modeling of the interaction with immediate boss

The rules and policies are imposed from the upper levels in a centralized organizational structure and subordinates follow their immediate boss. By assuming that each parent node is an immediate boss to its children, this behaviour can be modelled by updating the position of each search agent $\vec{x_i}$ with reference to its parent node B by using the following equation:

$$x_i^k(t+1) = B^k + \gamma\lambda^k|B^k - x_i^k(t)| \tag{1}$$

where t denotes the current iteration, k in superscript denotes the kth component of a vector, and $|\ |$ computes the absolute value. λ^k is the kth component of vector $\vec{\lambda}$, which is randomly generated as follows:

$$\lambda^k = 2r - 1 \tag{2}$$

where r is a random number generated according to the uniform distribution from the range $[0, 1]$. In Eq. (1), γ is a carefully designed parameter, which is computed as follows:

$$\gamma = \left|2 - \frac{(t \bmod \frac{T}{C})}{\frac{T}{4C}}\right| \tag{3}$$

where t denotes the current iteration, T represents the total number of iterations, and C is a user-defined parameter and explained below. With the course of iterations, γ reduces linearly from 2 to 0 and after reaching 0, it starts to increase back to 2 with iterations. However, it is the parameter C that determines how many cycles γ will complete in T iterations. For illustration see Fig. 3, where part (a) of the figure shows how γ changes with iterations. For better illustration, two scenarios are depicted. We set $C = 2$ in the first scenario, and $C = 5$ in the second scenario; however, in both scenarios T is set to 200. Fig. 3(b) highlights the region where $\gamma\lambda^k$ from Eq. (1) can be positioned at any location depending upon the value of random parameter λ^k. For example in scenario 1 ($C = 2$) at $t = 75, \gamma\lambda^k$ can have any value in the range $[-1, 1]$ depending upon the value of λ^k and in scenario 2 ($C = 5$) at $t = 75, \gamma\lambda^k$ can have any value in the range $[-1.5, 1.5]$ depending upon the value of λ^k. From Fig. 3, we can learn C should play a crucial role in Eq. (1) because C controls the rate of variation in the values of $\gamma\lambda^k$.

To see the impact of the parameter C on the performance of HBO, we solved several unimodal and multimodal benchmark functions by varying C from the lowest value to a high value. For instance, the impact of the value of C on the performance of HBO for Sphere, Generalized Penalized, Ackley, and Griewank benchmark functions is shown in Fig. 4. We can learn from the combined impact on the curves of unimodal and multimodal functions that the strong variation or the high value of C allows search agents to escape the local optima at the expense of weak exploitation. For example, if the total iterations are 1300, the performance for the multimodal functions (Generalized Penalized, Ackley, and Griewank) at small values of C is not very good because multimodal functions have many local optima and the algorithm might not be in a position to escape them. However, the results for the multimodal functions improve if the value of C is increased but the performance stabilizes after $C = \frac{T}{42}$. Based on these simulations we can fix the minimum value of C for the set of experiments on multimodal functions. Furthermore, the unimodal functions do not have local optima and the parameter C may behave differently for these functions. For instance, we are presenting the curve for Sphere function in Fig. 4. The performance for the Sphere (unimodal) func-

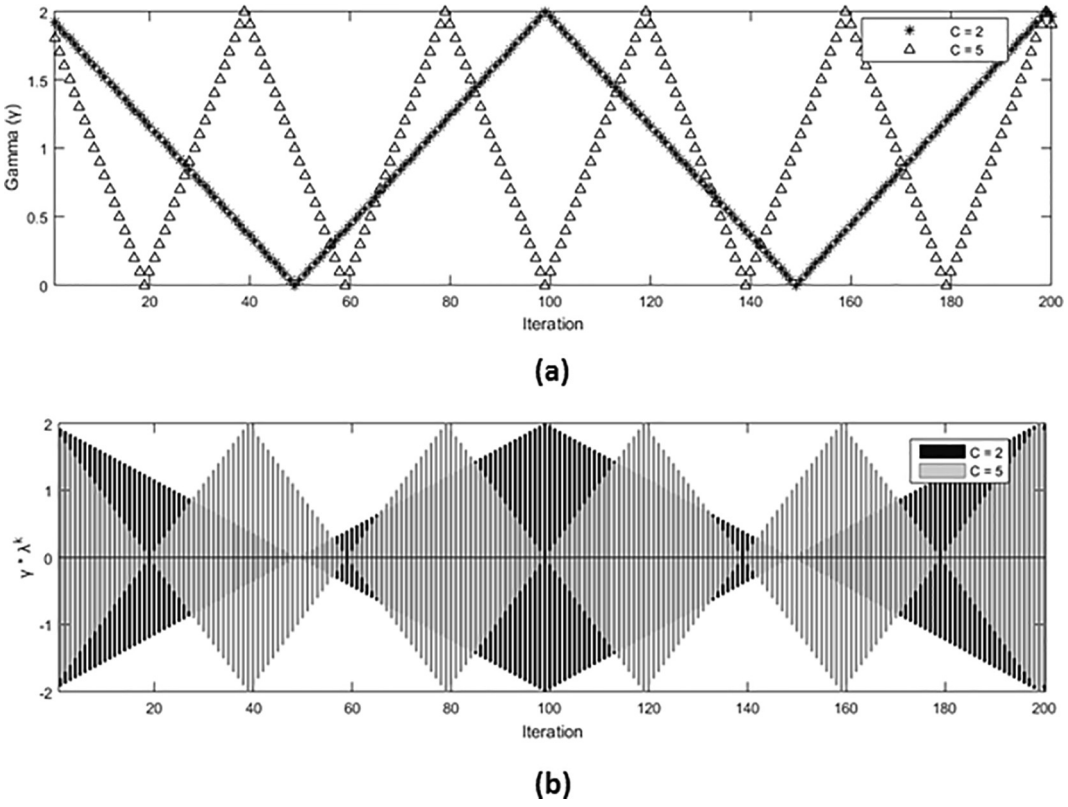


Fig. 3. Illustration of the impact of C and T on values of γ. (a) depicts two scenarios for two different values of C and (b) highlights the region where $\gamma\lambda^k$ from Eq. (1) can be positioned at any location, in both scenarios.

tion is best at $\frac{T}{25}$ and the performance starts deteriorating after that. By performing this experiment for many other functions we decided to calculate the balanced value of C as follows:

$$C = \lfloor T/25 \rfloor \tag{4}$$

The position updating mechanism of Eq. (1) is depicted in Fig. 5. By assuming $D = |B^k - x_i^k(t)|$, Fig. 5(a) highlights how $\gamma\lambda^k$ scales D up or down. For instance, in 40th iteration, $\gamma\lambda^k$ lies in the range $[-2, 2]$, which scales D in the range $[-2D, 2D]$. However, in 25th iteration $\gamma\lambda^k|B^k - x_i^k(t)|$ gives value in the range $[-0.5D, 0.5D]$ because $\gamma\lambda^k$ lies in $[-0.5, 0.5]$. Similarly, in any iteration, the range of $\gamma\lambda^k|B^k - x_i^k(t)|$ can be determined from Fig. 5(a). The role of $\gamma\lambda^k|B^k - x_i^k(t)|$ in Eq. (1) is to decide the area of the region to search around B^k. In Fig. 5 (b) the regions around B^k, at different values of t, are highlighted. For instance, the size of the region to search around B^k in 40th and 80th iterations is twice the distance $|B^k - x_i^k(t)|$. The size of the region to search around B^k in 10th, 30th, 50th, and 70th iterations is equal to the distance $|B^k - x_i^k(t)|$. However, in 15th, 25th, 55th, and 65th iterations the size of the region to search around B^k is half the distance $|B^k - x_i^k(t)|$. The discussion can be concluded as the $\gamma\lambda^k$ makes a search agent capable of escaping local optima when it scales D up and exploit the region around B^k when it scales D down.

2.4. Mathematical modeling of the interaction between colleagues

The officials with the same rank are considered to be the colleagues. They interact with each other in order to fulfill the official tasks. In heap, we assume the nodes at the same level are colleagues and each search agent $\vec{x}_i$ updates it position with reference to its randomly selected colleague $\vec{S}_r$ according to the equation given below:

$$x_i^k(t+1) = \begin{cases} S_r^k + \gamma\lambda^k|S_r^k - x_i^k(t)|, & f(\vec{S}_r) < f(\vec{x}_i(t)) \\ x_i^k + \gamma\lambda^k|S_r^k - x_i^k(t)|, & f(\vec{S}_r) \geqslant f(\vec{x}_i(t)) \end{cases} \tag{5}$$

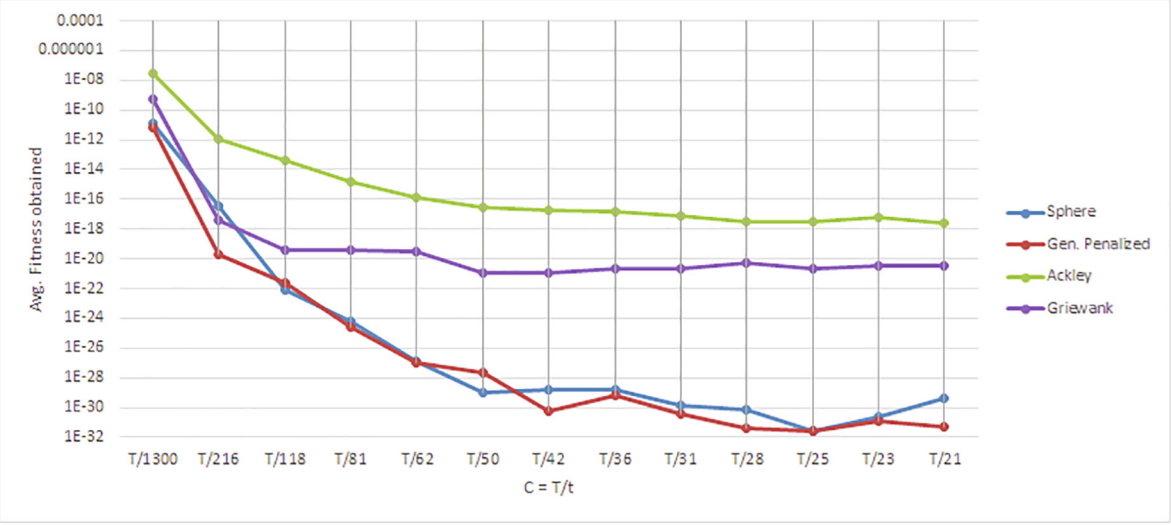


Fig. 4. Impact of the parameter C on the performance of HBO for 1 unimodal and 3 multimodal functions, where the number of dimensions (variables) = 30. T denotes the total iterations, t denotes iterations per cycle, and C denotes total number of cycles. Note: In this experiment T is assumed to be equal to 1300.

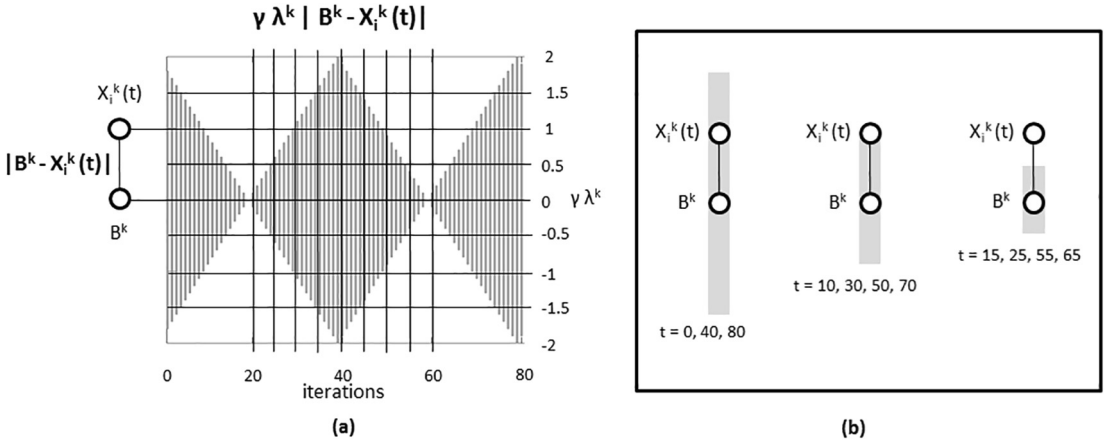


Fig. 5. Illustration of the position updating mechanism of Eq. (1). (a) illustrates how $\gamma\lambda^k$ scales $|B^k - x_i^k(t)|$ in any iteration and (b) highlights the regions around B^k which HBO searches in different iterations.

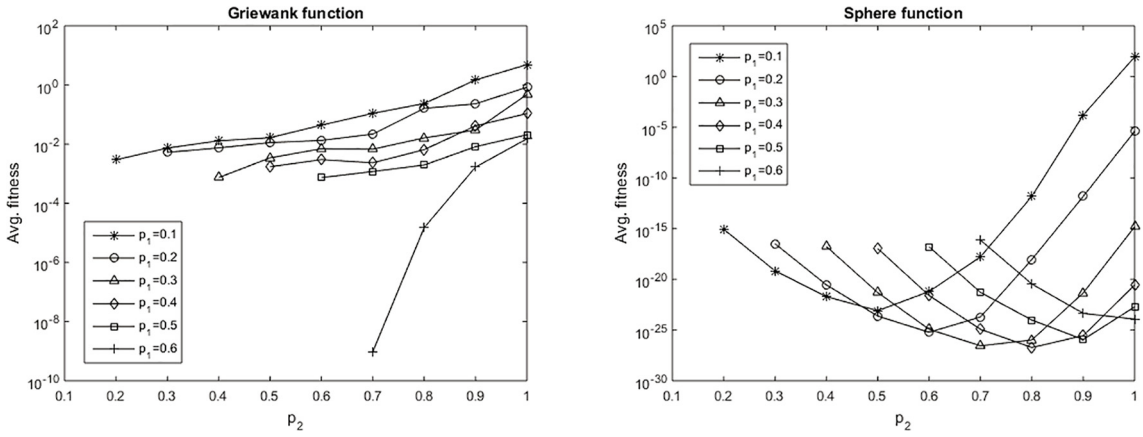


Fig. 7. Impact of different distinct combinations of p_1 and p_2 on exploration and exploitation of griewank and sphere functions.

where f denotes the objective function and computes the fitness of the search agent. The position updating mechanism of Eq. (5) is very similar to Eq. (1); however, Eq. (5) enables the search agent to explore the region around S_r^k if $f(\vec{S_r}) < f(\vec{x_i}(t))$ and allows to explore the region around x_i^k otherwise. For illustration see Fig. 6. This behavior promotes both exploration and exploitation. The random selection of the colleagues incorporates the diversity and always searching around good solutions promotes exploitation.

2.5. Modeling of the self contribution of an employee

This phase maps the concept of self-contribution of an employee. The mapping of this phase is kept very simple: however, we suggest a few variations in Section 6. We are modeling this behavior by retaining the previous position of the employee in the next iteration, as expressed below:

$$x_i^k(t+1) = x_i^k(t) \tag{6}$$

In Eq. (6), the search agent $\vec{x_i}$ does not change its position for its kth design variable in the next iteration. This behavior allows us to regulate the rate of change of a search agent. The next subsection discusses how exploration can be controlled with this equation.

2.6. Putting it all together

In this subsection, we discuss how to merge the position updating equations, modelled in previous subsections, into one equation. A major challenge is determining the probabilities of selection for all three equations as their probabilities of selection play a significant role in balancing exploration and exploitation. A roulette wheel is designed to balance these probabilities, which is split into three proportions p_1, p_2, and p_3. The selection of the proportion p_1 allows a search agent to update its position using Eq. (6). The limit of p_1 is computed as follows:

$$p_1 = 1 - \frac{t}{T} \tag{7}$$

where t denotes current the iteration and T denotes maximum number of iterations. The selection of proportion p_2 allows a search agent to update its position using Eq. (1). The limit of p_2 is computed as follows:

$$p_2 = p_1 + \frac{1-p_1}{2} \tag{8}$$

Finally, the selection of p_3 represents updating position using Eq. (5) and limit of p_3 is computed as follows:

$$p_3 = p_2 + \frac{1-p_1}{2} = 1 \tag{9}$$

A general positions updating mechanism of HBO is presented in the following equation:

$$x_i^k(t+1) = \begin{cases} x_i^k(t), & p \leqslant p_1 \\ B^k + \gamma\lambda^k|B^k - x_i^k(t)|, & p > p_1 \text{ and } p \leqslant p_2 \\ S_r^k + \gamma\lambda^k|S_r^k - x_i^k(t)|, & p > p_2 \text{ and } p \leqslant p_3 \text{ and } f(\vec{S_r}) < f(\vec{x_i}(t)) \\ x_i^k + \gamma\lambda^k|S_r^k - x_i^k(t)|, & p > p_2 \text{ and } p \leqslant p_3 \text{ and } f(\vec{S_r}) \geqslant f(\vec{x_i}(t)) \end{cases} \tag{10}$$

where p is a randomly generated number in the range $[0, 1]$. It is important to discuss why we are calculating p_1, p_2 and p_3 as mentioned above. Two well-known functions sphere and griewank are used to analyze the impact of distinct combinations of p_1 and p_2 on exploration and exploitation of these functions. See Fig. 7 for illustration. Since griewank is a multimodal function with regularly distributed and widespread local minima, the performance for this function can only be increased if the algorithm can explore its search space. From griewank function in Fig. 7, we can learn that the performance improves if the value of p_1 increases or in other words the participation of Eq. (6) increases; however, the performance starts decreasing if p_2 increases by fixing p_1 at any point. It should be noted that by increasing p_2 the participation of Eq. (1) increases but participation of Eq. (5) decreases. Conclusively, both Eq. (6) and Eq. (5) enhance exploration. However, the role of Eq. (1) will be prominent for sphere function. Since sphere is a unimodal function, the performance for this function can only be enhanced if the algorithm can exploit promising areas once they are discovered. From sphere function in Fig. 7, we can learn that

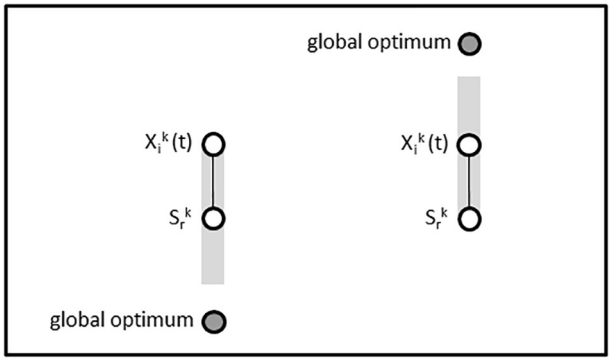


Fig. 6. Illustration of the position updating mechanism of both cases of Eq. (5).

the performance increases if both p_1 and p_2 increase but after a certain limit the performance begins to deteriorate by raising p_2 any further because the participation of Eq. (5) in the form of exploitation starts getting compromised. Conclusively, both Eq. (1) and Eq. (5) promote exploitation and after assigning a handsome proportion to p_1 for exploration the rest of the proportion should be equally divided for Eq. (1) and Eq. (5).

The whole discussion concludes the argument that Eq. (6) enhances exploration, Eq. (1) enhances exploitation and convergence, and Eq. (5) promotes both the exploration and exploitation. Based on these observations, p_1 is kept higher in beginning and is linearly reduced over the course of iterations, which reduces exploration with iterations and enhances exploitation with iterations. After computing p_1, the rest of the span is divided into two equal parts, which makes attraction towards the boss and colleagues equally probable.

3. Implementation of HBO and complexity overhead analysis

In this section, all the important steps of HBO and their implementation-related details are discussed. Furthermore, it is justified that integrating heap into the implementation of HBO does not affect the time and space complexity of the proposed algorithm asymptotically.

3.1. Steps of HBO

1. Definition and initialization of the parameters: Initialize general parameters, such as the size of the population (N), number of design variables/dimensions (D), maximum number of iteration (T), and ranges of the design variables (L_i, U_i). The algorithm specific parameter C can be computed by using Eq. (4).
2. Initialization of the population: Generate a random population P of N search agents, each consisting of D dimensions. The representation of population P is presented below:

$$\mathrm{P} = \begin{bmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \dots & x_1^D \\ x_2^1 & x_2^2 & x_2^3 & \dots & x_2^D \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^1 & x_N^2 & x_N^3 & \dots & x_N^D \end{bmatrix}$$

3. Building the heap: Heap can generally be a d-ary tree; however, we use 3-ary (ternary) heap to implement CRH in this paper. Although heap is a tree-shaped data structure, it can be implemented efficiently using an array because it has the property of completeness. The following are some significant d-ary heap-based operations needed to implement HBO.

- parent (i): Assuming the heap is implemented as an array, this function receives the index of a node and returns the index of the parent of that node. The formula to compute the index of the parent of node i is given below:

$$parent(i) = \lfloor \frac{i+1}{d} \rfloor \tag{11}$$

where $\lfloor\ \rfloor$ denotes the floor function, which returns the greatest integer less than or equal to the input. For example, $parent(13) = \lfloor \frac{14}{3} \rfloor = 4$, as depicted in Fig. 8.

- child (i,j): In a 3-ary heap a node can have maximum 3 children. In our mapping we can say, a boss may not have more than 3 direct subordinates. This function returns the index of the jth child of a node i. A constant-time mathematical formulation of this function is given below:

$$child(i,j) = d \times i - d + j + 1 \tag{12}$$

For example, as illustrated in Fig. 8, the index of the 2nd child of node 2 is computed as:

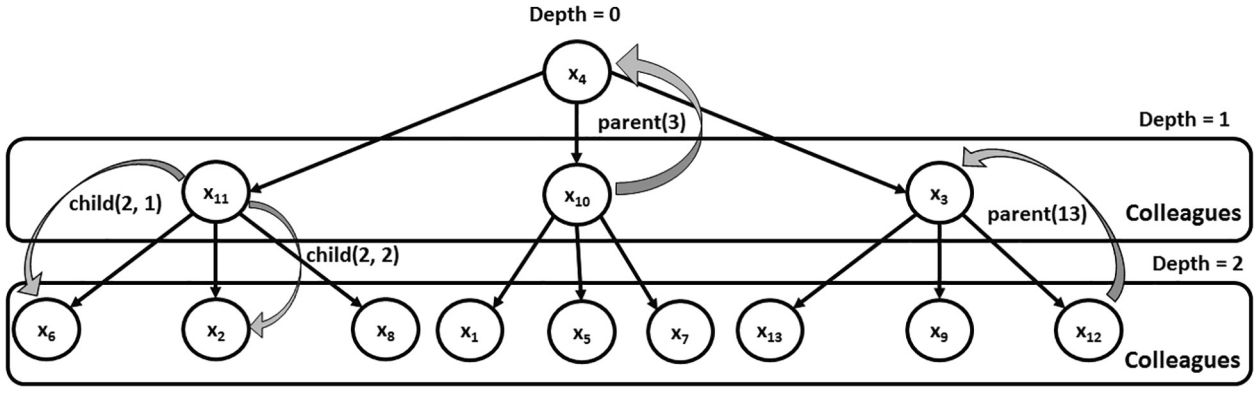
$$child(2,2) = 6 - 3 + 2 + 1 = 6$$

- depth (i): Considering the depth of the last level equals to 0, the depth of any node i can be computed in constant time by using the following formula:

$$depth(i) = \lceil log_d(d \times i - i + 1) \rceil - 1 \tag{13}$$

where $\lceil\ \rceil$ denotes the ceil function, which returns the smallest integer greater than or equal to the input. For example, the depth of node at index 27 in heap array is computed as:

$$depth(27) = \lceil log_3(81 - 27 + 1) - 1 \rceil = \lceil 2.6476 \rceil = 3$$



Indices	1	2	3	4	5	6	7	8	9	10	11	12	13
heap.value	4	11	10	3	6	2	8	1	5	7	13	9	12
heap.key	$f(x_4)$	$f(x_{11})$	$f(x_{10})$	$f(x_3)$	$f(x_6)$	$f(x_2)$	$f(x_8)$	$f(x_1)$	$f(x_5)$	$f(x_7)$	$f(x_{13})$	$f(x_9)$	$f(x_{12})$

Fig. 8. Illustration of different functions of the heap. Number of search agents = 13, where heap.value stores the indices of the search agents in the population and heap.key stores the fitness of the corresponding search agents.

- colleague (i): All nodes at the level of a node i are considered the colleagues of node i. This function returns the index of any randomly selected colleague of node i and it can be computed by generating any random integer in the range $[\frac{dd^{depth(i)-1}-1}{d-1}+1, \frac{dd^{depth(i)}-1}{d-1}]$. For example, the range for the colleagues of node 6 is highlighted in Fig. 8 and computed as follows:

$$\left[\frac{3\times 3^{depth(6)-1}-1}{3-1}+1, \frac{3\times 3^{depth(6)}-1}{3-1}\right] = [5, 13]$$

- Heapify_Up (i): It searches upward in the heap and inserts the node i at its correct location to maintain the heap property. The pseudo code for this operation is presented in Algorithm 1.

Algorithm 1. Heapify_Up (i)

Input: i (the index of the node we are trying to heapify)
▷ Assuming that the rest of the nodes fulfill the heap property
while $i \neq root$ *and* $heap[i].key < heap[parent(i)].key$ **do**
  swap($heap[i]$, $heap[parent(i)]$)
  $i \leftarrow parent(i)$
end

Finally, the algorithm to build the heap is presented in Algorithm 2. For a population of 13 search agents, a heap is built in Fig. 8 where *heap.value* stores the indices of the search agents in the population and *heap.key* stores the fitness of the corresponding search agents.

Algorithm 2. Build_Heap (P, N)

Input: P (population of search agents), N (population size)
for $i \leftarrow 1$ *to* N **do**
  $heap[i].value \leftarrow i$
  $heap[i].key \leftarrow f(x_i)$
  Heapify_Up (i)
end

Please note that all the operations presented above except *Heapify_Up*() and *Build_heap*() are constant time operations; however, the time complexity of *Heapify_Up*() is $O(log_d N)$ and time complexity of *Build_Heap*() is $O(N)$ (Zhu, Hu, & Zhu, 2019).

4. Repeated applications of position updating mechanism: Search agents update their positions repeatedly in accordance with earlier discussed equations and attempt to converge on the optimum global. The main body of HBO is presented in Algorithm 3.

Algorithm 3. HBO_Main_Body ()

for $t \leftarrow 1$ *to* T **do**
  Compute γ by using Eq. (3)
  Compute p_1 by using Eq. (7)
  Compute p_2 by using Eq. (8)
  for $I \leftarrow N$ *down to* 2 **do**
    $i \leftarrow heap[I].value$ ▷ i is the index of the search agent in population P corresponds to I^{th} node in heap. The heap is constructed in Algorithm 2.
    $bi \leftarrow heap[parent(I)].value$ ▷ bi is the index of the parent of I
    $ci \leftarrow heap[colleague(I)].value$ ▷ ci is the index of a random colleague of I
    $\vec{B} \leftarrow \vec{x}_{bi}$ ▷ $\vec{B}$ is the position vector of the parent of I
    $\vec{S} \leftarrow \vec{x}_{ci}$ ▷ $\vec{S}$ is the position vector of a random colleague of I
    for $k \leftarrow 1$ *to* D **do**
      $p \leftarrow rand()$
      $x^k_{temp} \leftarrow$ update $x^k_i(t)$ by using Eq. (10)
    end
    if $f(\vec{x}_{temp}) < f(\vec{x}_i(t))$ **then**
      $\vec{x}_i(t+1) \leftarrow \vec{x}_i(t)$
    end
    Heapify_Up (I)
  end
end
return $x_{heap[1].value}$

Table 2
Descriptions of unimodal fixed-dimension benchmark functions.

f.No.	Name	Vars	Range	f_{min}	f.No.	Name	Vars	Range	f_{min}
f1	Beale	2	[−4.5,4.5]	0	f6	Wayburn Seader 3	2	[−500,500]	19.10588
f2	Booth	2	[−10,10]	0	f7	Leon	2	[−1.2,1.2]	0
f3	Brent	2	[−10,10]	0	f8	Cube	2	[−10,10]	0
f4	Matyas	2	[−10,10]	0	f9	Zettl	2	[−5,10]	−0.00379
f5	Schaffer N. 4	2	[−100,100]	0.292579					

Table 3
Descriptions of unimodal variable-dimension benchmark functions.

f.No.	Name	Vars	Range	f_{min}	f.No.	Name	Vars	Range	f_{min}
f10	Sphere	30	[−100,100]	0	f18	Rosenbrock	30	[−30,30]	0
f11	Powell Sum	30	[−1,1]	0	f19	Brown	30	[−1,4]	0
f12	Schwefel's 2.20	30	[−100,100]	0	f20	Dixon and Price	30	[−10,10]	0
f13	Schwefel's 2.21	30	[−100,100]	0	f21	Powell Singular	30	[−4,5]	0
f14	Step	30	[−100,100]	0	f22	Xin-She Yang	30	[−20,20]	0
f15	Stepint	30	[−5.12,5.12]	−155	f23	Perm 0,D,Beta	5	[−Var,var]	0
f16	Schwefel's 2.22	30	[−100,100]	0	f24	Sum Squares	30	[−10,10]	0
f17	Schwefel's 2.23	30	[−10,10]	0					

Table 4
Descriptions of multimodal fixed-dimension benchmark functions.

f.No.	Name	Vars	Range	f_{min}	f.No.	Name	Vars	Range	f_{min}
f25	Egg Crate	2	[−5,5]	0	f39	Cross function	2	[−10,10]	0
f26	Ackley N.3	2	[−32,32]	−195.629	f40	Cross leg table	2	[−10,10]	−1
f27	Adjiman	2	[−1,2]	−2.02181	f41	Crowned cross	2	[−10,10]	0.0001
f28	Bird	2	[−2pi, 2pi]	−106.765	f42	Easom	2	[−100,100]	−1
f29	Camel 6 Hump	2	[−5,5]	−1.0316	f43	Giunta	2	[−1,1]	0.060447
f30	Branin RCOS	2	[−5,5]	0.397887	f44	Helical Valley	3	[−10,10]	0
f31	Goldstien Price	2	[−2,2]	3	f45	Himmelblau	2	[−5,5]	0
f32	Hartman 3	3	[0,1]	−3.86278	f46	Holder Table 2	2	[−10,10]	−19.2085
f33	Hartman 6	6	[0,1]	−3.32236	f47	Pen Holder	2	[−11,11]	−0.96354
f34	Cross-in-tray	2	[−10,10]	−2.06261	f48	Test Tube Holder	2	[−10,10]	−10.8723
f35	Bartels Conn	2	[−500,500]	1	f49	Shubert	2	[−10,10]	−186.731
f36	Bukin 6	2	[(−15,−5), (−5,−3)]	180.3276	f50	Shekel	4	[0, 10]	−10.5364
f37	Carrom Table	2	[−10,10]	−24.1568	f51	Three-Hump Camel	2	[−5,5]	0
f38	Chichinadze	2	[−30,30]	−43.3159					

Table 5
Descriptions of multimodal variable-dimension benchmark functions.

f.No.	Name	Vars	Range	f_{min}	f.No.	Name	Vars	Range	f_{min}
f52	Schwefel's 2.26	30	[−500,500]	−418.983	f61	Styblinski-Tang	30	[−5,5]	−1174.98
f53	Rastrigin	30	[−5.12,5.12]	0	f62	Griewank	30	[−100,100]	0
f54	Periodic	30	[−10,10]	0.9	f63	Xin-She Yang N. 4	30	[−10,10]	−1
f55	Qing	30	[−500,500]	0	f64	Xin-She Yang N. 2	30	[−2pi,2pi]	0
f56	Alpine N. 1	30	[−10,10]	0	f65	Gen. Penalized	30	[−50,50]	0
f57	Xin-She Yang	30	[−5,5]	0	f66	Penalized	30	[−50,50]	0
f58	Ackley	30	[−32,32]	0	f67	Michalewics	30	[0,pi]	−29.6309
f59	Trignometric 2	30	[−500,500]	0	f68	Quartic Noise	30	[−1.28,1.28]	0
f60	Salomon	30	[−100,100]	0					

3.2. Complexity overhead analysis

Most optimization algorithms use $O(ND)$ space to store population and use $O(N)$ space to store cost/fitness of all search agents. However, their overall space complexity can be expressed as $O(ND)$ where N denotes the total number of search agents in population and D denotes the number of dimensions of the problem. The space complexity of HBO is also bounded by $O(ND)$ because population consumes $O(ND)$ and heap consumes $O(N)$ space.

The time complexity analysis of most algorithms involves analyses of three components. (i) The time complexity of initialization of the population, generally bounded by $O(ND)$, (ii) time complexity of initial fitness evaluation, generally bounded by $O(NC_{0bj})$, where C_{obj} represents the cost of the objective function, and (iii) time complexity of the main loop, generally bounded by $O(TND + TNC_{0bj})$, where T denotes the total number of iterations. Since the cost of the objective functions varies from function to function, the general time complexity $T(N)$ of such algorithms can be expressed as follows:

Table 6
Descriptions of the benchmark functions from CEC 2017 (Awad et al., 2017).

f.No.	Name	Vars	Range	fmin
Unimodal functions				
f69	Shifted and Rotated Bent Cigar Function	10	[−100,100]	100
Multimodal functions				
f70	Shifted and Rotated Rosenbrock's Function	10	[−100,100]	300
f71	Shifted and Rotated Rastrigin's Function	10	[−100,100]	400
f72	Shifted and Rotated Expanded Scaffer's F6 Function	10	[−100,100]	500
f73	Shifted and Rotated Lunacek Bi$_R$*astriginFunction*	10	[−100,100]	600
f74	Shifted and Rotated Non-Continuous Rastrigin's Function	10	[−100,100]	700
f75	Shifted and Rotated Levy Function	10	[−100,100]	800
f76	Shifted and Rotated Schwefel's Function	10	[−100,100]	900
Hybrid functions (N is basic number of functions)				
f77	Hybrid Function 1 (N = 3)	10	[−100,100]	1000
f78	Hybrid Function 2 (N = 3)	10	[−100,100]	1100
f79	Hybrid Function 3 (N = 3)	10	[−100,100]	1200
f80	Hybrid Function 4 (N = 4)	10	[−100,100]	1300
f81	Hybrid Function 5 (N = 4)	10	[−100,100]	1400
f82	Hybrid Function 6 (N = 4)	10	[−100,100]	1500
f83	Hybrid Function 6 (N = 5)	10	[−100,100]	1600
f84	Hybrid Function 6 (N = 5)	10	[−100,100]	1700
f85	Hybrid Function 6 (N = 5)	10	[−100,100]	1800
f86	Hybrid Function 6 (N = 6)	10	[−100,100]	1900
Composite functions (N is basic number of functions)				
f87	Composite Function 1 (N = 3)	10	[−100,100]	2000
f88	Composite Function 2 (N = 3)	10	[−100,100]	2100
f89	Composite Function 3 (N = 4)	10	[−100,100]	2200
f90	Composite Function 4 (N = 4)	10	[−100,100]	2300
f91	Composite Function 5 (N = 5)	10	[−100,100]	2400
f92	Composite Function 6 (N = 5)	10	[−100,100]	2500
f93	Composite Function 7 (N = 6)	10	[−100,100]	2600
f94	Composite Function 8 (N = 6)	10	[−100,100]	2700
f95	Composite Function 9 (N = 6)	10	[−100,100]	2800
f96	Composite Function 10 (N = 3)	10	[−100,100]	2900
f97	Composite Function 11 (N = 3)	10	[−100,100]	3000

$$T(N) = \begin{cases} O(TND), & D > C_{0bj} \\ O(TNC_{0bj}), & Otherwise \end{cases} \quad (14)$$

On the contrary, the time complexity analysis of HBO involves analyses of four components: (i) The time complexity of initialization of the population bounded by $O(ND)$, (ii) time complexity of initial fitness evaluation bounded by $O(NC_{0bj})$, (iii) time complexity of Build_Heap bounded by $O(N)$, and (iv) time complexity of the main loop bounded by $O(TND + TNC_{0bj} + TNlog_dN)$, where additional $TNlog_dN$ is for calling heapify after updating each search agent. The overall time complexity $T(N)$ of HBO is also expressible through Eq. (14) because calling heapify for each search agent does not add any significant overhead to the overall time complexity. It should be noted that log_dN denotes the number of levels in the heap and the heaps we use to address most of the optimization problems only have 4 to 6 levels.

4. Experiments to evaluate the performance of HBO

A global optimization algorithm should be able to explore the search space to find promising areas and exploit the promising areas to converge on the global optimum. An algorithm requires a very nice equilibrium between exploration and exploitation to converge to the global optimum. In this section, we evaluate the performance and the behaviour of HBO from these three perspectives.

4.1. Experimental setup

To evaluate the performance of HBO, we are using 97 benchmark functions, which are classified in 5 groups: (i) Unimodal fixed-dimension benchmark functions, (ii) unimodal variable-dimension benchmark functions, (iii) multimodal fixed-dimension benchmark functions, (iv) multimodal variable-dimension benchmark functions, and (v) shifted, rotated, hybrid, and composite benchmark functions from CEC-BC-2017 (Awad et al., 2017). The names of these functions and their other related properties are presented in Tables 2, 3, 4, 5, and 6, respectively. In each table, Vars denotes the number of dimensions (design variables) of the functions, Range defines the lower and upper bounds of the design variables, and f_{min} represents the global minimum of the functions. Moreover, Table 2 describes the fixed-dimension unimodal benchmark functions, Table 3 presents variable-dimension unimodal benchmark functions, Table 4 describes fixed-dimension multimodal benchmark functions, Table 5 presents variable-dimension multimodal benchmark functions, and Table 6 presents shifted, rotated, hybrid, and composite benchmark functions from CEC-BC-2017. The 2D landscapes of few of the functions from each category are presented in Fig. 9. The researchers have used a suite of 23 classical benchmark functions in many literature articles (Mirjalili & Lewis, 2016; Mirjalili et al., 2014; Mirjalili, 2016; Rashedi et al., 2009); however, we extended their count to 97 in this study for a thorough assessment of the proposed algorithm. The performance of HBO is compared with 7 highly-referred state-of-the-art algorithms in the literature. The names of the algorithms are gravitational search algorithm (GSA) (Rashedi et al., 2009), particle swarm optimization (PSO) (Eberhart & Kennedy, 1995), sine cosine algorithm (SCA) (Mirjalili, 2016), moth flame optimization (MFO) (Mirjalili, 2015), multi-verse optimizer (MVO) (Mirjalili, Mirjalili, & Hatamlou, 2015), cuckoo search (CS) (Yang & Deb, 2009), and effective butterfly optimizer using covariance matrix adapted retreat phase (EBO-CMAR) (Kumar et al., 2017). The parameter settings of all these algorithms are taken from

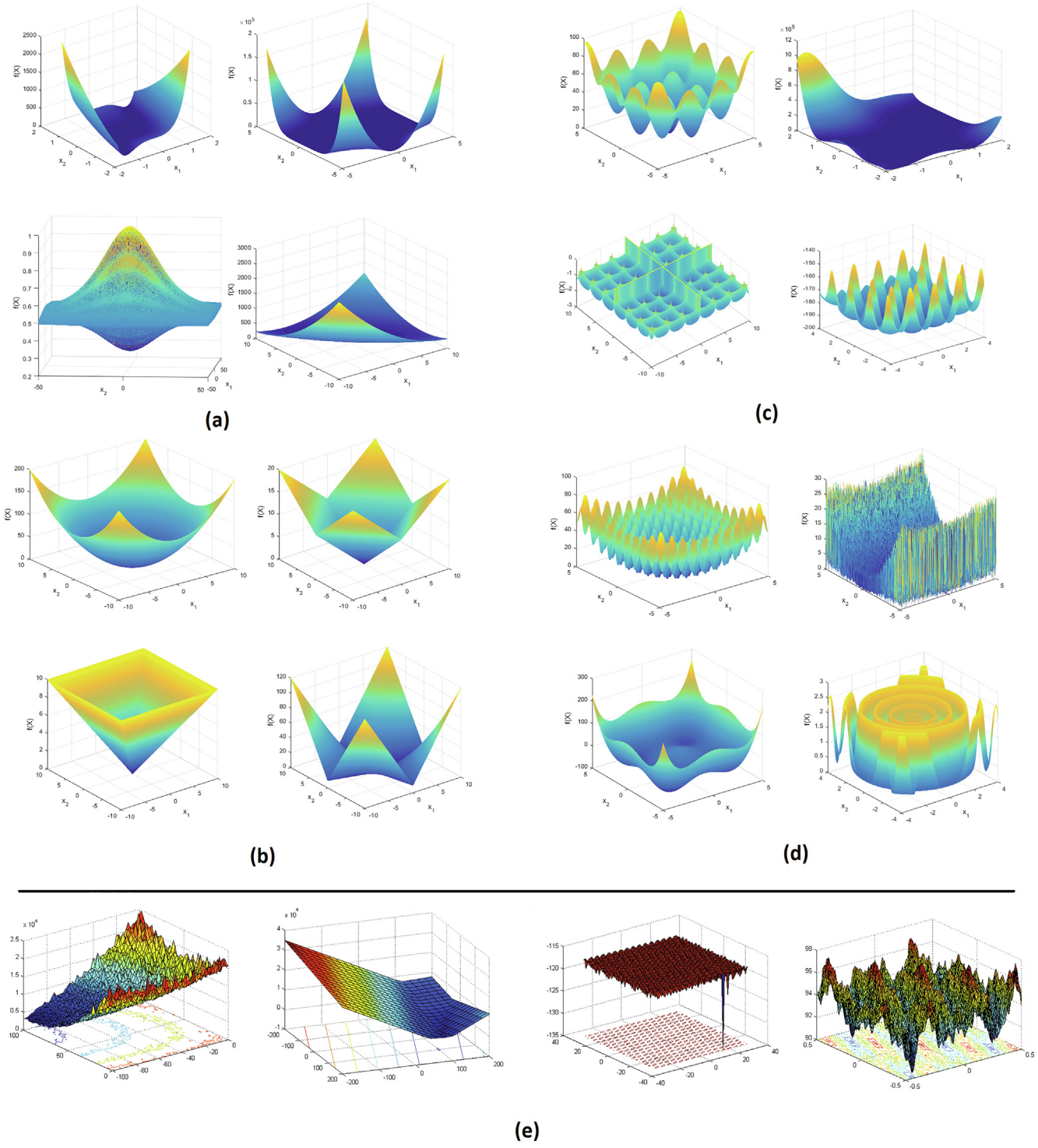


Fig. 9. 2D landscapes of a few benchmark functions from all 5 categories. (a) Fixed-dimension unimodal functions, (b) Variable-dimension unimodal functions, (c) Fixed-dimension multimodal functions, (d) Variable-dimension multimodal functions, and (e) Rotated-Shifted functions.

their original papers and are presented in Table 7. All the algorithms are coded in MATLAB programming software and simulations are run on a Core i7-4650U with 8 GB RAM. The MATLAB code will be released at GitHub after acceptance of the paper. The codes of all algorithms except PSO are published by their original authors. All results are produced by executing each algorithm 30 times for each benchmark function with randomly initialized populations.

4.2. Evaluation of the exploitation capability of HBO

The area around the good solutions is considered promising for the global optimum which is why many algorithms search the area around good solutions by attracting the poor search agents to them. HBO follows the same rule but in a different way. In HBO, a hierarchical approach is used in which each child exploits the area around its parent, as depicted in Fig. 10 (a). Two experiments

Table 7
Algorithms used for comparative analysis and their parameter settings. NFEs denotes the number of objective function evaluations.

Algorithm	Parameters setting	NFEs
GSA (Rashedi et al., 2009)	$N = 50, \alpha = 20, G_0 = 100, k = [N \rightarrow 1]$	50,000
PSO (Eberhart & Kennedy, 1995)	$N = 50, c1 = 2, c2 = 2, w = [0.9 \rightarrow 0.2]$	50,000
SCA (Mirjalili, 2016)	$N = 50, a = 2, [r1, r2, r3, r4]$ from corresponding eq.	50,000
MFO (Mirjalili, 2015)	$N = 30$	50,000
MVO (Mirjalili et al., 2015)	$N = 60, WEP_{max} = 1, WEP_{min} = 0.2$	50,000
CS (Yang & Deb, 2009)	$N = 20, pa = 0.25$	50,000
EBO-CMAR (Kumar et al., 2017)	$PS_{1,max} = 18D, PS_{1,min} = 4D, PS_{2,max} = 46.8D, PS_{2,min} = 10D, H = 6$ $\sigma = 0.3, CS = 100, prob_{ls} = 0.1, cfe_{ls} = 0.25 * FE_{max}$	50,000
HBO	$N = 40, [C, p_1, p_2]$ from corresponding equations.	50,000

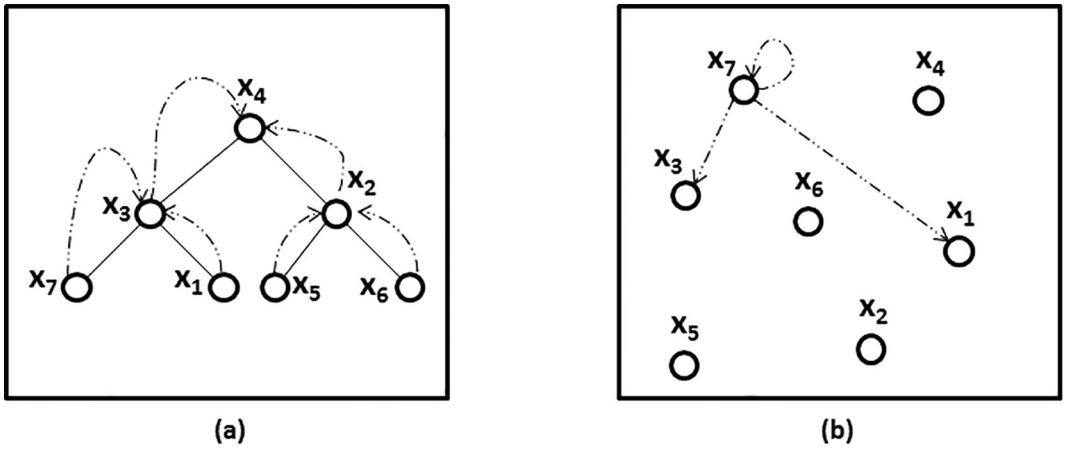


Fig. 10. (a) Depicts the hierarchical position updating approach and (b) illustrates the search agent's own impact and the impact of boss and colleagues on its position updating.

Table 8
Comparison of the performance of HBO with other algorithms on unimodal fixed-dimension functions.

f.No.	Stats	HBO	GSA	PSO	SCA	MFO	MVO	CS	EBO-CMAR
f1	Med	**0**	6.29E−21	**0**	7.23E−05	**0**	1.6E−08	**0**	**0**
	Mean	**0**	9.48E−21	**0**	7.88E−05	2.35E−32	1.98E−08	**0**	**0**
	Std	0	7.41E−21	0	5.6E−05	1.01E−31	1.44E−08	0	0
f2	Med	**0**	8.29E−21	**0**	0.000148	**0**	1.36E−07	**0**	**0**
	Mean	**0**	1.2E−20	**0**	0.000313	**0**	1.77E−07	**0**	**0**
	Std	0	1.3E−20	0	0.000331	0	1.61E−07	0	0
f3	Med	**1.38E−87**	1.51E−05	**1.38E−87**	**1.38E−87**	**1.38E−87**	**1.38E−87**	**1.38E−87**	**1.38E−87**
	Mean	**1.38E−87**	4.09E−05	**1.38E−87**	**1.38E−87**	**1.38E−87**	**1.38E−87**	**1.38E−87**	**1.38E−87**
	Std	6.8E−103	6.78E−05	4.6E−103	4.6E−103	4.6E−103	4.6E−103	4.6E−103	2.4E−103
f4	Med	5.1E−107	4.68E−22	1.3E−95	1.7E−132	2.6E−142	3.55E−09	1.21E−55	**0**
	Mean	9.21E−78	6.88E−22	7.36E−93	2.1E−121	2.75E−66	5.03E−09	1.26E−52	**0**
	Std	5.05E−77	7.43E−22	3.08E−92	9.8E−121	1.38E−65	5.21E−09	4.33E−52	0
f5	Med	**0.292579**	0.301431	**0.292579**	**0.292579**	**0.292579**	**0.292579**	**0.292579**	**0.292579**
	Mean	**0.292579**	0.306436	**0.292579**	**0.292579**	0.292687	**0.292579**	**0.292579**	**0.292579**
	Std	7.14E−17	0.014464	7.93E−17	1.76E−07	0.000288	6.19E−07	6.73E−11	9.2E−09
f6	Med	**19.10588**	**19.10588**	**19.10588**	19.11139	**19.10588**	19.10677	**19.10588**	**19.10588**
	Mean	**19.10588**	**19.10588**	**19.10588**	19.11828	**19.10588**	19.10717	**19.10588**	**19.10588**
	Std	1.45E−14	3.4E−15	9.17E−15	0.014647	6.76E−15	0.001452	7.47E−15	2.37E−15
f7	Med	8.53E−17	0.00257	1.5E−22	6.5E−05	0.001156	1.25E−08	**0**	7.36E−21
	Mean	3.59E−13	0.002669	1.45E−20	0.000133	0.004254	2.48E−08	**0**	3.90E−12
	Std	1.29E−12	0.002082	3.86E−20	0.000196	0.005276	3.4E−08	0	1.21E−08
f8	Med	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Std	0	0	0	0	0	0	0	0
f9	Med	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**
	Mean	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**	**−0.00379**
	Std	1.76E−18	9.9E−19	1.33E−18	2.89E−10	1.33E−18	2.9E−08	1.33E−18	0

are performed to practically assess the exploitative capability of HBO. It should be noted that the unimodal functions are used in both experiments because unimodal functions are deemed a helpful source for assessing the algorithm's exploitative capacity. In the first experiment, the results for fixed-dimension unimodal benchmark functions are compared with other algorithms in Table 8 and

Table 9
Comparison of the performance of HBO with other algorithms on unimodal variable-dimension functions.

f.No.	Stats	HBO	GSA	PSO	SCA	MFO	MVO	CS	EBO-CMAR
f10	Med	**4.6E−28**	2.4E−17	2.1E−12	0.00029	3.1E−10	0.19031	7.8E−05	8.3E−16
	Mean	**8.5E−27**	2.4E−17	4.8E−11	0.00737	800	0.21839	0.00011	8.99E−16
	Std	3.6E−26	8.5E−18	2.1E−10	0.02329	2768.87	0.06965	0.00011	2.71E−16
f11	Med	**5.3E−70**	5E−18	4.9E−25	2.2E−15	3.2E−31	8.6E−08	4.4E−17	5.73E−33
	Mean	**1.3E−64**	2.7E−17	6.6E−22	2E−10	1.2E−27	8.8E−08	1.1E−15	6.53E−32
	Std	7.3E−64	6.2E−17	2.8E−21	8E−10	5E−27	5.9E−08	2.9E−15	1.72E−31
f12	Med	**2E−18**	2.3E−08	3.6E−06	1.5E−05	100	2.62075	0.01978	4.56E−06
	Mean	**2.8E−18**	2.3E−08	6.8E−06	5.9E−05	76.0004	2.8031	0.02831	5.27E−06
	Std	3.8E−18	3.2E−09	8.9E−06	0.00013	83.0658	0.79795	0.03062	2.23E−06
f13	Med	1.13756	**3.4E−09**	0.41017	11.7186	67.6111	0.62211	10.5873	0.000477
	Mean	1.21633	0.01449	0.40665	13.7426	66.4658	0.65522	11.5805	**0.000802**
	Std	0.8007	0.07246	0.11811	8.47197	8.71094	0.20993	3.53693	0.000709
f14	Med	**3.1E−28**	1.9E−17	2.1E−12	4.11205	3.3E−10	0.19494	0.00012	8.25E−16
	Mean	**1.3E−26**	2E−17	1.5E−10	4.20256	3588.09	0.19032	0.00017	8.07E−16
	Std	5.4E−26	6.2E−18	5.3E−10	0.4928	6365.06	0.02881	0.00018	2.04E−16
f15	Med	**−155**	−119	−133	−107	**−155**	−147	**−155**	−122
	Mean	**−155**	−118.96	−132.16	−106.16	**−155**	−147.8	−154.56	−106.1
	Std	0	2.93655	10.455	4.61591	0	2.39792	1.00333	3.12291
f16	Med	**6.6E−19**	58.5087	9.7E−06	2.5E−06	400	1.1E+19	1E+10	1.82E−05
	Mean	**1.5E−18**	63.6029	0.0002	7.5E−05	448	1.1E+21	1E+10	3.22E−05
	Std	1.7E−18	48.0521	0.00067	0.0003	204.369	3.6E+21	0	2.62E−05
f17	Med	2.7E−59	**3.1E−88**	2.1E−19	0.01581	8.6E−19	2E−15	7.6E−06	8.12E−37
	Mean	2.8E−50	**4.3E−88**	2.8E−16	1711.85	3.2E−11	1.6E−14	0.00151	3.94E−35
	Std	1.5E−49	4.6E−88	1.2E−15	8248.58	1.6E−10	4.9E−14	0.00631	1.32E−34
f18	Med	76.1145	26.0798	25.9212	31.1826	555.187	31.8312	29.517	**9.933503**
	Mean	63.8433	28.9477	3643.9	73.3545	3209394	265.897	48.8975	**9.769767**
	Std	33.5034	13.9235	17994.4	103.248	1.6E+07	531.457	29.1874	1.136834
f19	Med	**5.3E−31**	4.2E−17	31	2E−08	12	0.00053	6.9E−07	3.43E−15
	Mean	**2.5E−30**	4.3E−17	39	7.8E−07	14.48	0.00055	1.3E−06	3.87E−15
	Std	5.3E−30	1.3E−17	24.1039	2.8E−06	9.36358	0.00018	1.3E−06	1.91E−15
f20	Med	**0.66667**	**0.66667**	0.67049	0.72082	95.7188	0.85047	0.69866	**0.666667**
	Mean	**0.66673**	0.67272	96.1583	2.6131	40292.9	2.075	1.01916	**0.666667**
	Std	0.00028	0.03027	134.03	4.85844	83888.4	2.71328	0.63911	3.01E−12
f21	Med	0.00166	0.01148	390.268	0.01223	211.189	0.32012	0.0071	**5.12E−09**
	Mean	0.00186	0.02005	690.979	2.52091	892.32	0.32042	0.0102	**8.81E−09**
	Std	0.00074	0.02257	812.839	12.164	1296.1	0.15215	0.00737	1.34E−08
f22	Med	**4E−232**	3.6E−41	**4E−232**	6E−199	**4E−232**	1E−177	**4E−232**	**4.3E−232**
	Mean	**4E−232**	1.6E−35	**4E−232**	5E−189	**4E−232**	2E−159	**4E−232**	**4.3E−232**
	Std	0	5E−35	0	0	0	8E−159	0	0
f23	Med	1.13066	348.325	**0.07123**	19.9472	0.52853	0.22093	1.26415	2.016307
	Mean	1.4738	512.377	**0.18433**	21.5403	7.27586	0.39812	1.48717	2.163735
	Std	1.44998	469.867	0.27907	13.2359	15.7645	0.48897	1.32319	2.010001
f24	Med	**6.1E−29**	1.9E−16	8.1E−10	2.6E−05	300	0.15061	7.2E−06	1.03E−12
	Mean	**2E−28**	2E−16	68	0.00023	564	0.18363	1.3E−05	1.60E−12
	Std	5E−28	4.3E−17	98.8264	0.00052	670.746	0.10162	1.2E−05	1.21E−12

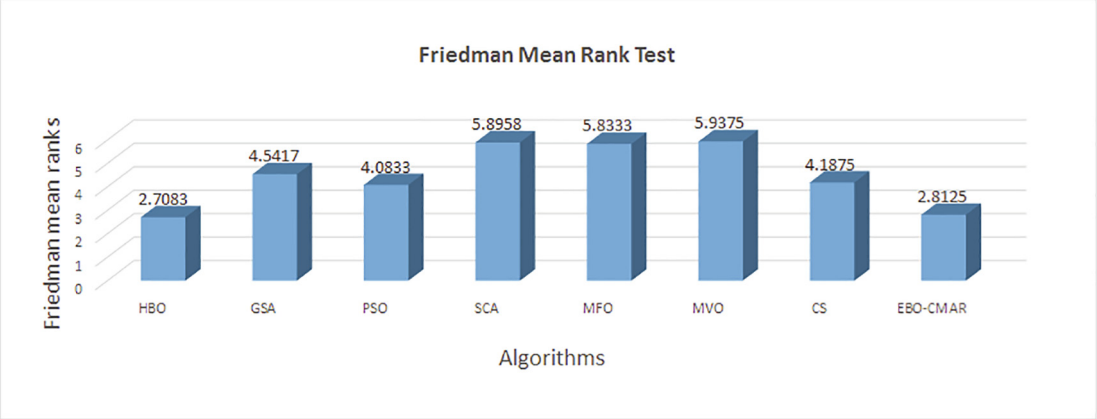


Fig. 11. Result of Friedman Mean Rank Test for all unimodal functions. Smaller bars/outcomes on the bars denote the higher ranks.

in the second experiment, the results for variable-dimension unimodal benchmark functions are compared with other algorithms in Table 9. In the case of fixed-dimension functions, HBO and a few other algorithms share 1st position for all functions except $f4$ and $f7$; however, the results for $f4$ and $f7$ are also very good and stochastically equivalent to the other good performers. In the case of variable-dimension functions, HBO outperforms the other algorithms by securing 1st position for all functions except

Table 10
Comparison of the performance of HBO with other algorithms on multimodal fixed dimension functions.

f.No.	Stats	HBO	GSA	PSO	SCA	MFO	MVO	CS	EBO-CMAR
f25	Med	**0**	8.12E−20	1E−129	4.4E−158	**0**	2.26E−07	7.73E−52	**0**
	Mean	**0**	1.09E−19	7.8E−128	3.5E−150	**0**	2.94E−07	1.09E−49	**0**
	Std	0	1.16E−19	1.7E−127	1.7E−149	0	2.6E−07	1.91E−49	0
f26	Med	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**
	Mean	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**	**−195.629**
	Std	5.78E−14	5.8E−14	5.8E−14	6.38E−05	5.8E−14	2.9E−06	5.8E−14	3E−14
f27	Med	**−2.02181**	−2.02138	**−2.02181**	**−2.02181**	**−2.02181**	**−2.02181**	**−2.02181**	**−2.02181**
	Mean	**−2.02181**	−2.02119	**−2.02181**	**−2.02181**	**−2.02181**	**−2.02181**	**−2.02181**	**−2.02181**
	Std	1.36E−15	0.000654	1.36E−15	7.6E−10	1.36E−15	3.49E−11	1.36E−15	4.68E−16
f28	Med	**−106.765**	**−106.765**	**−106.765**	−106.753	**−106.765**	**−106.765**	**−106.765**	**−106.765**
	Mean	**−106.765**	−106.616	**−106.765**	−106.745	**−106.765**	**−106.765**	**−106.765**	**−106.765**
	Std	2.95E−14	0.556386	8.7E−15	0.019459	2.8E−14	2.35E−06	4.1E−15	2.72E−14
f29	Med	**−1.03163**	**−1.03163**	**−1.03163**	−1.03162	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**
	Mean	**−1.03163**	**−1.03163**	**−1.03163**	−1.03162	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**
	Std	6.71E−16	5.55E−16	6.72E−16	1.29E−05	6.8E−16	5.75E−08	6.8E−16	7.4E−17
f30	Med	**0.397887**	**0.397887**	**0.397887**	**0.39818**	**0.397887**	**0.397887**	**0.397887**	**0.397887**
	Mean	**0.397887**	**0.397887**	**0.397887**	**0.398279**	**0.397887**	**0.397887**	**0.397887**	**0.397887**
	Std	0	0	0	0.000339	0	2.58E−08	0	0
f31	Med	**3**	**3**	**3**	3.000004	**3**	**3**	**3**	**3**
	Mean	**3**	**3**	**3**	3.00001	**3**	3.000001	**3**	**3**
	Std	1.04E−15	1.83E−15	9.46E−16	1.33E−05	1.68E−15	6.55E−07	9.64E−16	0
f32	Med	**−3.86278**	**−3.86278**	**−3.86278**	−3.85468	**−3.86278**	**−3.86278**	**−3.86278**	**−3.86278**
	Mean	**−3.86278**	**−3.86278**	−3.86247	−3.85551	**−3.86278**	**−3.86278**	**−3.86278**	**−3.86278**
	Std	2.71E−15	1.99E−15	0.001576	0.002786	2.27E−15	3.02E−07	2.27E−15	9.36E−16
f33	Med	**−3.322**	**−3.322**	−3.19957	−3.0753	−3.2031	−3.32199	**−3.322**	**−3.322**
	Mean	**−3.322**	**−3.322**	−3.22063	−3.03156	−3.23811	−3.26481	**−3.322**	−3.29822
	Std	1.36E−15	4.53E−16	0.113038	0.145542	0.06013	0.060747	4.53E−16	0.05013
f34	Med	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**
	Mean	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**	**−2.06261**
	Std	9.03E−16	9.06E−16	9.06E−16	5.31E−06	9.06E−16	6.14E−09	9.06E−16	4.68E−16
f35	Med	**1**	1.000004	**1**	**1**	**1**	1.00274	**1**	**1**
	Mean	**1**	1.00001	**1**	**1**	**1**	1.002914	**1**	**1**
	Std	0	1.31E−05	0	0	0	0.001642	0	0
f36	Med	**180.3276**	180.3277	**180.3276**	**180.3276**	**180.3276**	**180.3276**	NA	NA
	Mean	**180.3276**	180.3278	**180.3276**	**180.3276**	**180.3276**	**180.3276**	NA	NA
	Std	0	0.000329	0	0	0	0	NA	NA
f37	Med	**−24.1568**	**−24.1568**	**−24.1568**	−24.1435	**−24.1568**	**−24.1568**	**−24.1568**	**−24.1568**
	Mean	**−24.1568**	−24.1373	**−24.1568**	−24.1383	**−24.1568**	**−24.1568**	**−24.1568**	**−24.1568**
	Std	8.9E−15	0.043688	8.97E−15	0.015637	5.08E−15	1.6E−06	5.85E−15	5.92E−15
f38	Med	**−42.9444**	−42.9386	**−42.9444**	−42.9433	**−42.9444**	**−42.9444**	**−42.9444**	**−42.9444**
	Mean	**−42.9444**	−42.8605	−42.8728	−42.9429	**−42.9444**	**−42.9444**	**−42.9444**	**−42.9444**
	Std	3.61E−14	0.153608	0.167332	0.001715	2.9E−14	4.06E−06	2.9E−14	1.47E−14
f39	Med	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**
	Mean	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**	**4.85E−05**
	Std	1.38E−20	6.92E−21	6.92E−21	1.1E−10	6.92E−21	1.36E−13	6.92E−21	7.01E−21
f40	Med	**−0.08478**	−0.00762	**−0.08478**	−0.00037	−0.08284	−0.00035	−0.00436	**−0.08478**
	Mean	−0.08472	−0.00754	−0.15413	**−0.37981**	−0.10401	−0.00038	−0.00648	−0.26782
	Std	0.000356	0.000899	0.255101	0.482087	0.189542	0.000141	0.0054	0.378938
f41	Med	**0.00118**	0.013535	**0.00118**	0.413084	0.001207	0.227099	0.024604	**0.00118**
	Mean	0.001179	0.013084	**0.001012**	0.282322	0.008629	0.22724	0.029771	0.009655
	Std	6.56E−07	0.001716	0.000406	0.240743	0.015171	0.037235	0.01734	0.000448
f42	Med	**−1**	**−1**	**−1**	−0.99958	**−1**	−0.99999	**−1**	−0.98999
	Mean	**−1**	**−1**	**−1**	−0.99952	**−1**	−0.99999	**−1**	−0.98999
	Std	0	0	0	0.000378	0	8.9E−06	0	9.90E−06
f43	Med	**0.06447**	**0.06447**	**0.06447**	0.064476	**0.06447**	**0.06447**	**0.06447**	**0.06447**
	Mean	**0.06447**	**0.06447**	**0.06447**	0.064478	**0.06447**	**0.06447**	**0.06447**	**0.06447**
	Std	4.82E−17	4.25E−17	4.84E−17	6.06E−06	6.23E−17	7.09E−10	5.2E−17	3.91E−17
f44	Med	2.31E−29	0.002338	**1.53E−38**	0.000147	0.000153	1.75E−05	1.27E−33	2.19E−10
	Mean	9.77E−26	0.007471	**4.36E−34**	0.001891	0.001914	5.41E−05	1.64E−27	1.89E−09
	Std	2.77E−25	0.012988	2.02E−33	0.005523	0.003791	7.55E−05	8.17E−27	3.40E−09
f45	Med	**0**	5.05E−20	**0**	0.003713	**0**	1.44E−07	1.02E−23	**0**
	Mean	**7.89E−32**	8.85E−20	2.84E−31	0.004536	2.21E−31	2.39E−07	9.24E−22	6.89E−31
	Std	2.41E−31	8.59E−20	3.86E−31	0.004803	3.62E−31	2.74E−07	2.66E−21	3.39E−31
f46	Med	**−19.2085**	**−19.2085**	**−19.2085**	−19.2011	**−19.2085**	**−19.2085**	**−19.2085**	**−19.2085**
	Mean	**−19.2085**	−19.1934	−19.161	−19.1994	**−19.2085**	**−19.2085**	**−19.2085**	**−19.2085**
	Std	8.47E−15	0.02884	0.237557	0.00849	1.1E−14	3.32E−07	6.99E−15	3.35E−15
f47	Med	**−0.96353**	−0.96341	**−0.96353**	−0.96352	**−0.96353**	**−0.96353**	**−0.96353**	**−0.96353**
	Mean	**−0.96353**	−0.96336	**−0.96353**	−0.96352	**−0.96353**	**−0.96353**	**−0.96353**	**−0.96353**
	Std	0	0.000164	0	1.85E−05	0	6.19E−10	0	1.17E−16
f48	Med	**−10.8723**	−10.8612	**−10.8723**	**−10.8723**	**−10.8723**	**−10.8723**	**−10.8723**	**−10.8723**
	Mean	**−10.8723**	−10.8601	**−10.8723**	**−10.8723**	−10.8562	−10.8652	**−10.8723**	**−10.8723**
	Std	3.61E−15	0.012144	3.63E−15	4.13E−07	0.023801	0.009704	3.63E−15	1.87E−15
f49	Med	**−186.731**	−185.585	**−186.731**	−186.618	**−186.731**	**−186.731**	**−186.731**	**−186.731**
	Mean	**−186.731**	−184.279	**−186.731**	−186.549	**−186.731**	**−186.731**	**−186.731**	**−186.731**

(*continued on next page*)

Table 10 (*continued*)

f.No.	Stats	HBO	GSA	PSO	SCA	MFO	MVO	CS	EBO-CMAR
	Std	2.64E−14	3.674921	4.26E−14	0.167024	3.12E−14	7.45E−05	2.62E−10	2.02E−08
f50	Med	**−10.5364**	**−10.5364**	**−10.5364**	−4.91002	**−10.5364**	−10.5363	**−10.5364**	**−10.5364**
	Mean	**−10.5364**	**−10.5364**	−10.1038	−5.05406	−9.17268	−9.89116	**−10.5364**	−10.5365
	Std	1.81E−15	2.08E−15	1.497388	1.893045	2.810418	1.783183	8.22E−14	1.87E−15
f51	Med	**0**	7.03E−21	9.2E−129	9.8E−154	**0**	7.66E−09	6.28E−54	1.24E−59
	Mean	**0**	9.84E−21	1.4E−126	2.2E−146	**0**	1.22E−08	2.02E−50	3.94E−57
	Std	0	9.87E−21	4.5E−126	9.7E−146	0	1.05E−08	9.77E−50	3.33E−58

Table 11
Comparison of the performance of HBO with other algorithms on multimodal variable dimension functions.

f.No.	Stats	HBO	GSA	PSO	SCA	MFO	MVO	CS	EBO-CMAR
f52	Med	**1.27E−05**	327.8278	199.3841	280.8342	134.4251	152.8595	124.8529	173.779
	Mean	**2.368779**	325.6861	197.0912	280.2942	133.0182	153.686	121.3023	197.8038
	Std	3.680208	14.32713	34.56364	9.926151	22.691	19.59828	15.69451	24.93263
f53	Med	**1.492439**	16.41682	80.59118	3.602768	157.2026	92.58562	66.32891	11.93951
	Mean	**1.757761**	17.41178	79.8289	10.18856	157.5637	96.53296	66.89123	23.25561
	Std	1.244442	4.114998	21.80321	14.53029	33.24362	27.12155	11.0364	5.031696
f54	Med	1.242823	**1**	2.775754	4.20689	4.255549	1.002174	1.170081	**1**
	Mean	1.225406	**1**	2.775754	4.531994	4.113488	1.002159	1.16335	**1**
	Std	0.109622	1.02E−16	0.955203	1.744219	1.097844	0.000456	0.025051	1.24E−13
f55	Med	**7.98E−11**	1592117	**6.29E−11**	5111.529	3.03E−06	346.6131	1E+10	4.85E−07
	Mean	**4.96E−09**	2193799	**1.33E−09**	44938.17	1.29E−05	348.8602	1E+10	0.000414
	Std	1.35E−07	2192991	5.36E−09	165063.9	2.73E−05	107.5889	0	0.001591
f56	Med	**1.87E−12**	2.33E−09	5.97E−06	0.005094	4.440211	3.68881	4.801046	3.52E−06
	Mean	**7.03E−09**	**2.38E−09**	0.355371	0.019103	7.646744	3.758177	5.486771	9.63E−06
	Std	2.04E−08	4.63E−10	1.229393	0.047921	7.69664	1.351876	1.712033	1.44E−05
f57	Med	**5.28E−22**	2.86E−06	50742.82	1.28E−05	367131.9	0.033149	0.775464	3.1E−11
	Mean	**1.67E−19**	0.001128	1928062	0.000228	4481183	2.602246	8.208223	5.63E−11
	Std	4.77E−19	0.003176	6519140	0.000456	10848431	10.10839	19.11906	8.56E−11
f58	Med	**2.22E−14**	3.48E−09	1.4E−06	18.60399	18.98799	0.650862	1.935688	1.08E−05
	Mean	**2.73E−14**	3.43E−09	3.95E−06	13.92582	14.11539	0.735326	1.701354	1.10E−05
	Std	1.47E−14	6.89E−10	6.03E−06	7.874642	8.439071	0.619771	1.005471	4.05E−08
f59	Med	**1**	5152.369	5.933111	103.8615	121.266	182.0208	129.2541	5.485088
	Mean	**1**	6565.999	8.117298	107.4402	60130.07	182.1578	131.8499	8.300152
	Std	2.14E−16	4248.258	4.972892	20.73517	130692.8	27.56456	27.77208	4.818281
f60	Med	**0.199873**	1.308996	0.299873	0.199985	4.099873	0.499874	1.399901	**0.199873**
	Mean	0.229873	1.248939	0.347873	0.256104	6.071873	0.547873	1.378919	**0.179873**
	Std	0.046609	0.233003	0.06532	0.096212	3.701522	0.096264	0.268042	0.041404
f61	Med	**−1174.98**	−1097.23	−1076.03	−612.026	−1019.48	−1005.34	−1106.92	**−1174.98**
	Mean	**−1174.98**	−1107.13	−1085.08	−613.355	−1015.61	−1011.56	−1100.76	**−1174.98**
	Std	4.22E−14	22.28142	31.57904	31.91868	47.41921	38.73212	22.74872	0.027262
f62	Med	**0**	**0**	0.012316	0.00159	0.039202	0.034666	0.001853	1.96E−13
	Mean	**7.4E−18**	0.000493	0.011133	0.115381	0.758264	0.034398	0.005808	2.09E−13
	Std	2.82E−17	0.002206	0.00644	0.211554	1.383064	0.009681	0.009444	1.19E−13
f63	Med	8.45E−22	**6.67E−30**	9.39E−25	1.59E−10	1.20E−13	6.45E−16	6.71E−14	4.8E−13
	Mean	7.23E−20	**6.9E−30**	1.88E−14	1.69E−10	1.07E−13	6.52E−16	7.01E−14	4.26E−13
	Std	2.05E−19	1.9E−30	6.5E−14	8.18E−11	1.13E−13	1.68E−16	3.24E−14	1.82E−13
f64	Med	**5.29E−12**	**3.51E−12**	3.06E−11	3.47E−10	2.71E−11	1.79E−11	1.59E−11	1.9E−11
	Mean	**5.31E−12**	**3.53E−12**	3.06E−11	3.82E−10	2.61E−11	2.33E−11	1.54E−11	9.87E−11
	Std	7.41E−13	5.33E−14	1.34E−12	1.95E−10	3.03E−12	1.51E−11	3.68E−12	3.96E−12
f65	Med	**1.25E−28**	2.35E−18	4.92E−12	2.378227	0.010987	0.034323	3.919759	5.66E−15
	Mean	**1.17E−27**	0.000549	0.004395	3.027752	16402510	0.039241	5.786021	2.18E−14
	Std	**3.63E−27**	0.002457	0.005494	1.743096	82012548	0.019365	5.487841	3.43E−14
f66	Med	**5.43E−30**	1.56E−19	1.99E−14	0.655037	0.103669	1.234599	2.128034	1.35E−14
	Mean	**6.13E−29**	0.063639	0.004147	1.219443	0.37375	1.296764	2.120552	5.38E−14
	Std	2.2E−28	0.133857	0.020734	1.671748	0.535287	1.22462	0.85441	1.24E−13
f67	Med	**−27.895**	−27.6524	−17.8654	−7.93566	−25.3297	−15.1814	−18.0941	−24.4579
	Mean	**−27.8159**	−27.451	−17.9252	−7.93718	−25.1561	−14.8792	−18.1274	−24.4436
	Std	0.90429	0.827775	1.839035	0.83881	1.862249	1.662599	0.960404	1.055604
f68	Med	0.010402	0.018265	2.720734	0.016155	1.637257	0.010857	0.077495	**0.004267**
	Mean	0.010944	0.020008	4.873331	0.02002	5.650202	0.012286	0.0904	**0.004466**
	Std	0.003052	0.007638	6.195275	0.012613	9.489989	0.004913	0.039609	0.001445

$f13, f17, f18, f21$ and $f23$; however, the performance for these 5 functions is also comparable to the others. From the outcomes, we can conclude that HBO has excellent exploitation capability and can deliver better outcomes in challenging environments. There are a few reasons behind good exploitative behavior of HBO. First, the value of γ in the range $[-1, 1]$ allows search agents to exploit area around their bosses/colleagues. Second, a gradual increase in the selection probability of Eq. (1) and Eq. (5) allows the population to evolve towards the better solutions. Third, a search agent is allowed to update its position only if the next position is better than the previous position.

4.2.1. Statistical analysis of the results for unimodal functions

To see the overall rank of each algorithm, we have performed Friedman mean rank test. Friedman test is a non-parametric statistical test developed by Milton Friedman (Friedman, 1937; Milton,

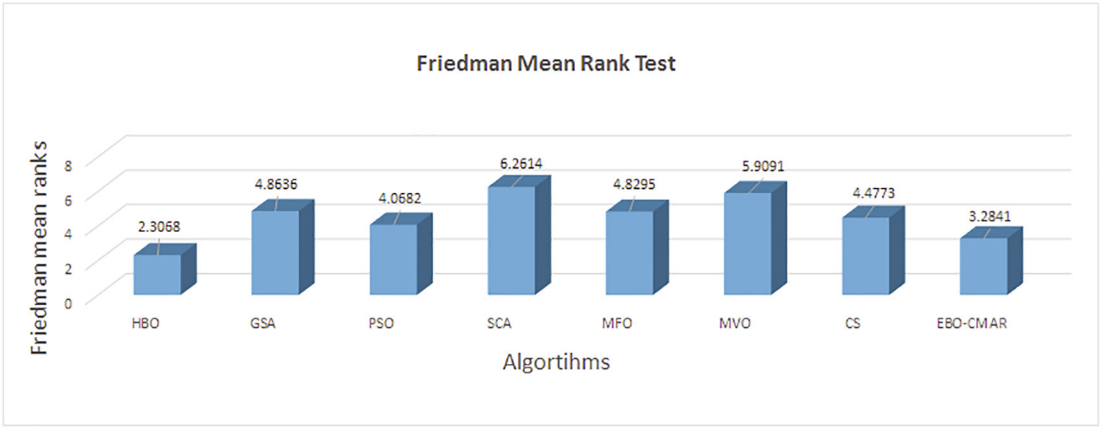


Fig. 12. Result of Friedman Mean Rank Test for all multimodal functions. Smaller bars/outcomes on the bars denote the higher ranks.

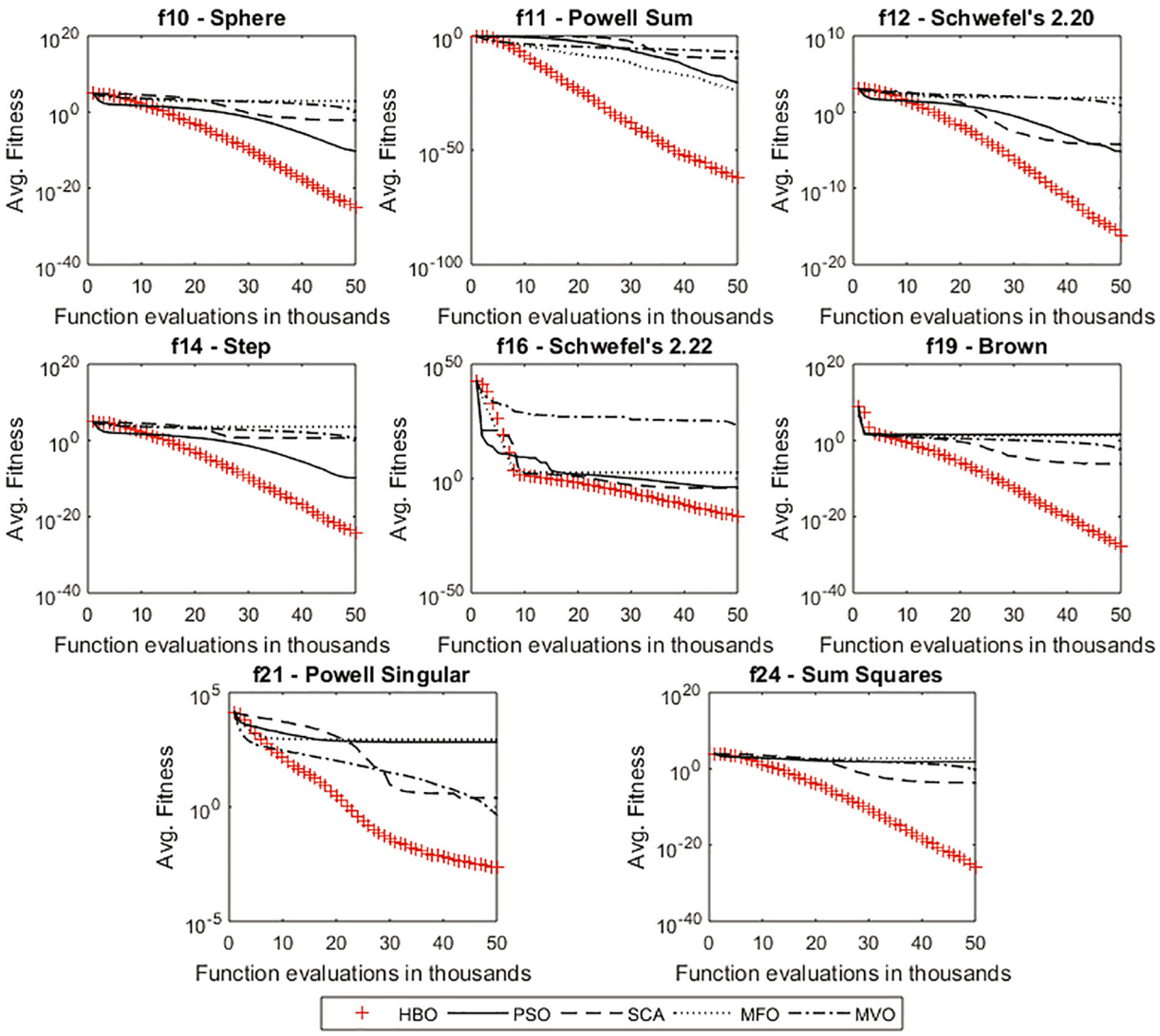


Fig. 13. Comparison of convergence curves of HBO with other algorithms for a few variable-dimension unimodal benchmark functions.

1939; Friedman, 1940). Friedman test involves sorting each block of rows together, then considering the ranked column values. The mean ranks of all algorithms given by Friedman test for unimodal functions are presented in Fig. 11. Smallest bar or lowest value on the bar in the figure indicates the best mean rank. The results show that HBO secures 1st rank and EBO-CMAR (winner of CEC-BC-2017) secures 2nd rank; however, the difference between the ranks of HBO and EBO-CMAR is very minute and insignificant. Con-

clusively, HBO has comparatively good exploitative capability, which is little better than the winner of CEC-BC-2017 and significantly better than the others.

4.3. Evaluation of the exploration capability of HBO

By comparing the landscapes of multimodal and unimodal benchmark functions presented in Fig. 9, we can easily find that reaching the global optimum for multimodal functions is tougher than the unimodal functions because multimodal functions have many local optima and an optimization algorithm needs to escape them. A search agent may escape local optimum if force is applied to the search agent from different directions to pull it out of the local optimum. This is how HBO attains exploration. Before to discuss it in detail, let the exploration capacity of HBO first be evaluated by optimizing multimodal benchmark functions and comparing efficiency with other algorithms. First experiment is performed for fixed-dimension multimodal benchmark functions. Although, these functions have fixed number of dimensions (design variables) and can not be scaled up or down but they offer a very challenging landscapes to optimize, for illustration see Fig. 9 (c). The results of HBO and all other algorithms for the fixed-dimension multimodal benchmarks are presented in Table 10. The findings obviously demonstrate that in most instances, HBO reaches the global optimum. If we compare the results then HBO shares 1st position for all functions except $f40$, $f41$ and $f44$. Performance for such functions, however, is also similar to the algorithms that give the best results for these functions. Second experiment is performed for variable-dimension multimodal benchmark functions. The dimensionality, in addition to the local optima, makes these functions more complex and more difficult to optimize. The capacity of HBO to solve variable-dimension multimodal functions is compared with other algorithms in Table 11. The findings indicate that HBO is able to explore the search space tremendously. In all instances, HBO outperforms competitors except $f54$, $f63$ and $f68$; however, the performance for these three functions is also competitive. The sources of excellent capacity for exploration are discussed below:

- As discussed in Section 2, Eq. (6) allows search agents to converge gradually, thus stops them from converging to some local optimum rapidly.
- In HBO, unlike many well-known optimization algorithms, neither all search agents update their position with reference to a common good solution (global leader) nor a search agent updates the position with reference to only one better solution. In fact, each search agent follows its own boss, as depicted in Fig. 10(a), which allows each search agent to explore vicinity around a different good solution. Moreover, a search agent, for its different design variables, might be following a different solution depending upon the position updating equation selected probabilistically, see Fig. 10(b) for illustration. In

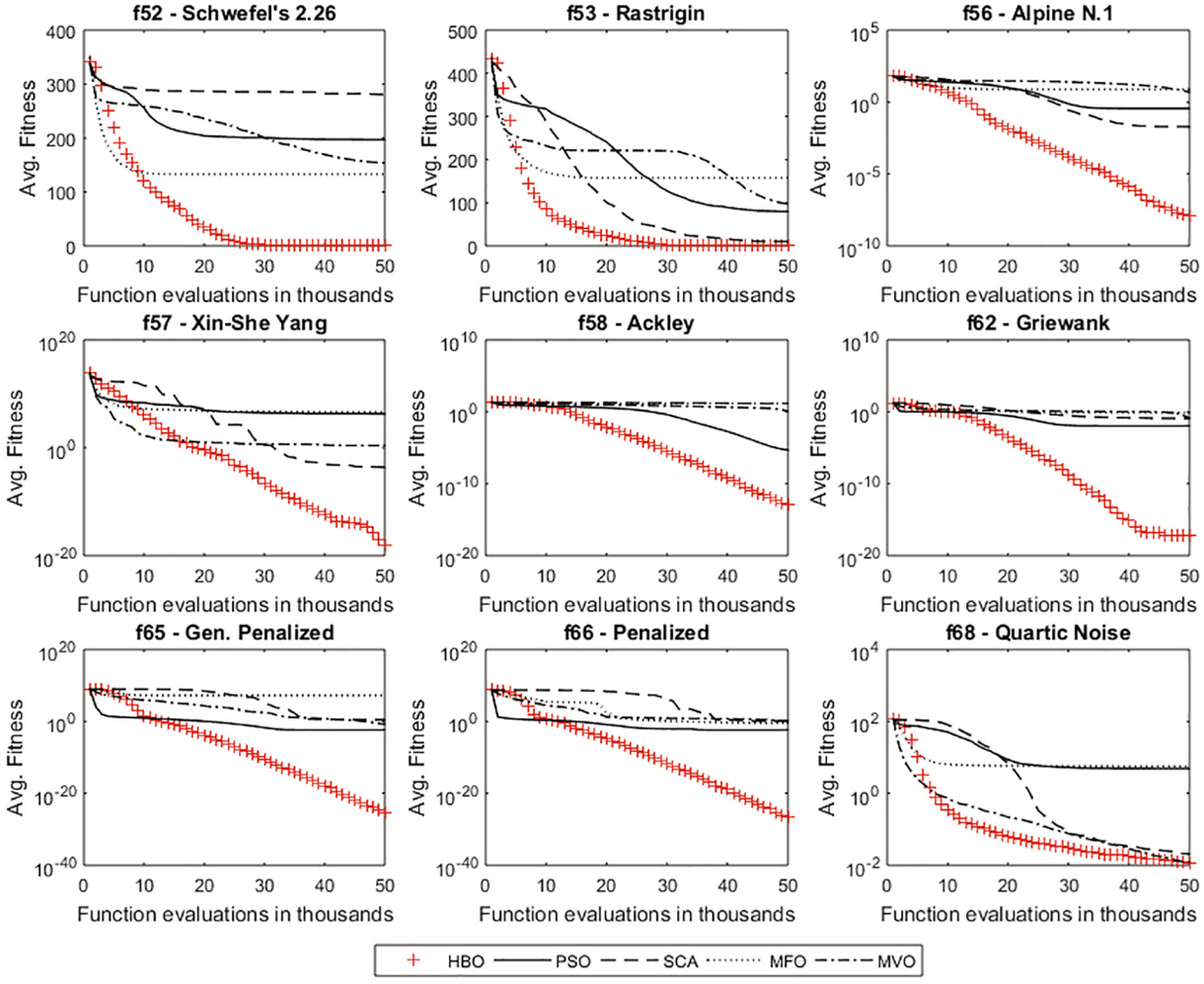


Fig. 14. Comparison of convergence curves of HBO with other algorithms for a few variable-dimension multimodal benchmark functions.

Fig. 10(b), it is shown that x_7 is updating its one design variable with reference to its boss x_3, other with reference to its colleague x_1, and one another is keeping unchanged.

- Another source of exploration in HBO is the definition of γ. Whenever it gives value in the range $[-2,-1]$ or $[1,2]$ the search agent has a chance to jump out of the vicinity bounded by the distance between the search agent and the reference solution, which enhances the exploration capability of HBO. It has already been depicted in Fig. 5(b). Furthermore, the repetitions of these ranges for γ are increased if the value of C is increased.

4.3.1. Statistical analysis of the results for multimodal functions

To determine the overall rank of each algorithm for the multimodal functions, the results of Friedman mean rank test are reported in Fig. 12. Smallest bar or lowest value on the bar in the figure indicates the best mean rank. The results show that HBO secures 1st rank and EBO-CMAR secures 2nd rank; however, in this case, the difference between the ranks of HBO and EBO-CMAR is not minute. The performance of HBO and most of the comparative algorithms for fixed-dimension multimodal functions is nearly equivalent; however, the real strength of HBO can be seen in the

Table 12
Comparison of the statistical results obtained for CEC-BC-2017 benchmark test functions.

f.No.	Stats	HBO	GSA	PSO	SCA	MFO	MVO	CS	EBO-CMAR
f69	Avg	535.7896	296.00	3959.60	8E+08	1.67E+08	8643.646	3.23E+08	**100**
	Std	543.1567	275.10	4456.60	1.84E+08	5.06E+08	5831.444	4.17E+08	0
f70	Avg	300.0249	10829.20	300.00	1543.853	11597.22	300.0254	301.2	**300**
	Std	0.116151	1620.74	0.00	1046.025	10006.07	0.10903	6.4E−06	0
f71	Avg	404.7788	406.60	405.94	437.8285	424.7637	405.7562	405.1247	**400**
	Std	0.428089	2.92	3.28	16.71067	34.25905	1.413572	0.086962	1.89E−14
f72	Avg	510.549	556.70	513.06	550.7501	529.4872	519.5746	518.3719	**500.3672**
	Std	3.634439	8.40	6.54	7.982532	11.92222	5.986546	5.916674	0.657057
f73	Avg	**600**	621.60	600.24	618.8765	602.241	600.7418	600.8636	**600**
	Std	4.8E−14	9.02	0.98	3.947706	1.977177	0.849693	0.651739	6.56E−14
f74	Avg	713.052	714.60	718.98	771.5427	735.5393	727.2502	730.2516	**710.8196**
	Std	4.027615	1.55	5.10	8.233241	8.114764	4.702083	4.755411	0.28215
f75	Avg	808.7198	820.50	811.39	839.3499	837.0564	820.2996	820.6072	**800.2247**
	Std	3.934154	4.69	5.47	6.499727	13.03039	8.663515	4.640893	0.413797
f76	Avg	**900**	**900.00**	**900.00**	996.687	1007.864	900.2162	902.9494	**900**
	Std	0	0.00	0.00	41.0157	176.4655	0.566283	3.35952	0
f77	Avg	1552.687	2694.60	1473.30	2259.251	1814.364	1744.32	1640.57	**1099.353**
	Std	156.713	297.62	214.97	166.7697	282.0393	263.1552	147.4421	85.70686
f78	Avg	1102.227	1134.70	1110.50	1200.436	1156.355	1130.25	1103.58	**1100**
	Std	1.231383	10.45	6.28	83.65854	78.4699	24.21214	1.241033	2.74E−11
f79	Avg	81747.03	702723.00	14532.00	11910003	1344229	616701.9	1413.652	**1287.319**
	Std	91097.37	42075.40	11260.00	11005843	2863313	473060.5	64.29277	100.1878
f80	Avg	2961.075	11053.00	8601.10	31047.57	14740.09	12208.16	1309.985	**1303.104**
	Std	2557.818	2110.55	5123.60	18855.22	12992.74	6129.402	2.498851	2.691863
f81	Avg	1453.975	7147.50	1482.10	1638.34	4613.985	1459.757	1413.601	**1400.035**
	Std	74.8672	1489.52	42.46	99.61163	2946.199	24.65836	6.198319	0.061189
f82	Avg	1604.459	18001.00	1714.30	2177.631	14499.43	1610.813	1501.966	**1500.149**
	Std	202.2075	5498.67	282.89	229.2363	11276.97	145.9304	0.658341	0.199345
f83	Avg	1602.001	2149.70	1860.00	1741.876	1799.57	1768.661	1610.443	**1600.643**
	Std	2.739227	105.80	127.65	80.46611	146.3447	142.4687	10.97171	0.322247
f84	Avg	1702.797	1857.70	1761.60	1785.585	1777.122	1789.585	1730.826	**1700.411**
	Std	4.303241	108.32	47.50	28.72761	45.19119	61.28143	7.935544	0.384295
f85	Avg	3771.107	8720.50	14599.00	178737.5	19869.11	21441.57	1807.308	**1802.265**
	Std	1154.932	5060.10	11852.20	149895.5	11934.01	13764.76	3.7134	6.24546
f86	Avg	1977.153	13670.00	2602.80	3376.863	10397.75	1929.977	1902.082	**1900.043**
	Std	157.9963	19168.00	2185.02	2901.632	12201.66	19.16336	0.351679	0.03392
f87	Avg	**2000.021**	2272.30	2085.10	2087.696	2074.901	2044.661	2028.09	2000.156
	Std	0.079201	81.72	62.25	22.736	46.66931	34.67326	6.540017	0.16453
f88	Avg	2250.372	2357.70	2281.70	2236.762	2319.647	2286.316	2256.678	**2230.587**
	Std	49.59053	28.20	54.02	53.59605	37.17628	59.06308	57.30963	49.24987
f89	Avg	**2299.039**	2300.00	2314.80	2368.054	2304.205	2347.236	2301.879	2300
	Std	4.916489	0.07	66.10	19.87323	4.59443	138.0584	15.86097	1.05E−11
f90	Avg	2613.876	2736.50	2620.80	2659.471	2630.681	2616.139	2619.442	**2601.974**
	Std	4.341088	39.14	9.23	8.751258	8.697182	5.697303	3.511326	1.764107
f91	Avg	2633.722	2742.20	2692.20	2760.82	2762.662	2727.99	**2567.618**	2578.898
	Std	95.03502	5.52	108.20	68.05451	10.18015	80.7585	78.64748	108.3447
f92	Avg	**2912.622**	2937.50	2924.00	2963.651	2950.647	2951.075	2917.312	2920.656
	Std	19.81313	15.36	25.02	17.28965	31.27117	42.92945	19.69569	24.0413
f93	Avg	2885.041	34407.50	2952.10	3073.043	3022.139	2900.134	**2842.893**	2890
	Std	62.5852	628.73	249.66	25.57284	92.7532	0.037098	94.2884	42.1637
f94	Avg	**3090.815**	3259.50	3116.20	3103.917	3095.412	3099.225	3091.353	3091.296
	Std	1.619507	41.66	24.99	2.122732	2.880396	20.45328	1.276217	2.486536
f95	Avg	3174.83	3459.40	3315.90	3294.704	3387.949	3301.581	**3100.129**	3111.573
	Std	73.69574	33.84	121.83	72.75564	58.63187	132.742	0.278911	36.59639
f96	Avg	3177.423	3449.50	3203.80	3247.233	3224.653	3178.9	3191.396	**3138.36**
	Std	13.69572	171.33	52.26	15.19993	64.38182	39.23594	32.11374	3.858333
f97	Avg	25686.73	1303361.00	350650.00	1024036	832852.4	304191.9	4071.321	**3403.194**
	Std	14554.84	363843.00	504857.00	928491.9	712542.7	616893.3	266.4059	14.72898

case of variable-dimension benchmark functions in which HBO mostly outperform the other algorithms including EBO-CMAR. Conclusively, HBO demonstrates good exploration capability especially in the case of variable-dimension benchmark functions.

4.4. Evaluation of convergence capability

Convergence is the state which reaches when the entire population is settled at a single point and/or stops improving. If an algorithm lacks the ability for exploration, it can converge prematurely to some local optimum; however, by lacking exploitative capability an algorithm might not even converge at a single point. An excellent balance between exploration and exploitation is needed to prevent premature convergence and reach the global optimum. In HBO, that balance is attained through the parameters p_1, and p_2. The definition of p_1 allows HBO to avoid premature convergence by starting from the highest probability and linearly decreasing to 0 with the course of iterations. The high value of p_1 stops search agent to quickly converge but as the proportion for p_1 decreases and the proportions for p_2 and p_3 increase the attraction towards the better solutions also increases, which finally results in convergence.

To practically examine the convergence capability of HBO, its convergence curves for 8 unimodal variable-dimension functions are plotted and compared with PSO, SCA, MFO and MVO in Fig. 13. HBO, as compared to the others, is little steady in the beginning because of the higher value of p_1 but it keeps improving smoothly with the iterations and becomes better than the others after 5 to 10 thousand function evaluations, which roughly implies $p_1 < 0.85$. To see the impact of local optima on the convergence of HBO, its convergence curves for 9 multimodal variable-dimension functions are compared in Fig. 14. For multimodal functions the behavior of HBO can be categorised in 3 types, which are discussed below:

- For $f56$, $f57$, $f65$ and $f66$, the behavior of HBO is well-balanced. HBO keeps improving linearly from the very beginning without getting trapped in any local optimum and outperforms the others.
- In the cases of $f58$ and $f62$, HBO progresses steadily for first 10 to 15 thousand function evaluations because of being in the exploration phase and improves quickly afterwards as the rate of exploitation starts increasing significantly.
- Finally, for the functions $f52$, $f53$ and $f68$, HBO makes a curve showing the rapid convergence in the first half of the iterations. If the landscapes of these functions are analyzed, with so many local optima, they have very challenging surfaces. For these functions, a rapid convergence requires a strong ability to explore the search space that HBO demonstrates.

4.5. Performance evaluation of HBO on CEC-BC-2017 test functions

The suite of benchmark functions designed for CEC-BC-2017 includes 30 unimodal, multimodal, composite, and hybrid func-

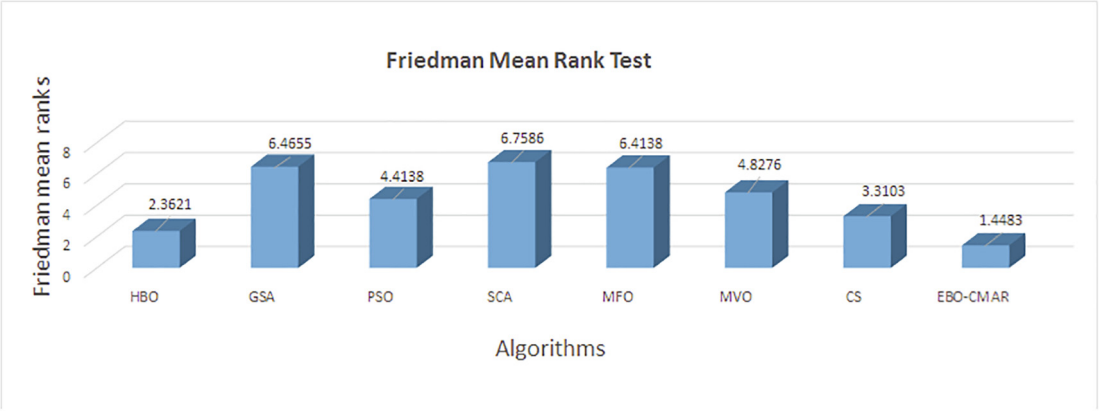


Fig. 15. Result of Friedman Mean Rank Test for all CEC-BC-2017 functions. Smaller bars/outcomes on the bars denote the higher ranks.

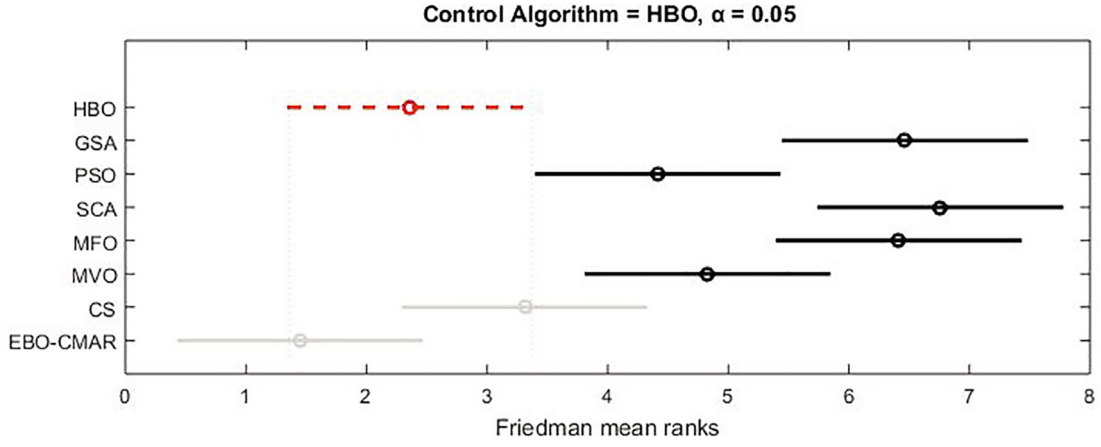


Fig. 16. Considering HBO as the reference/control algorithm, the multiple comparison test based on Bonferroni method at significance level of 0.05.

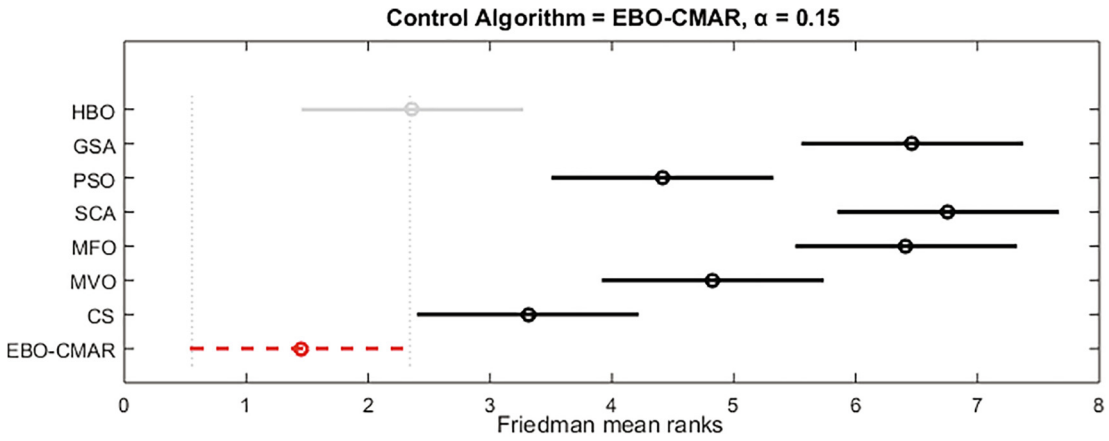


Fig. 17. Considering EBO-CMAR as the reference/control algorithm, the multiple comparison test based on Bonferroni method at significance level of 0.15.

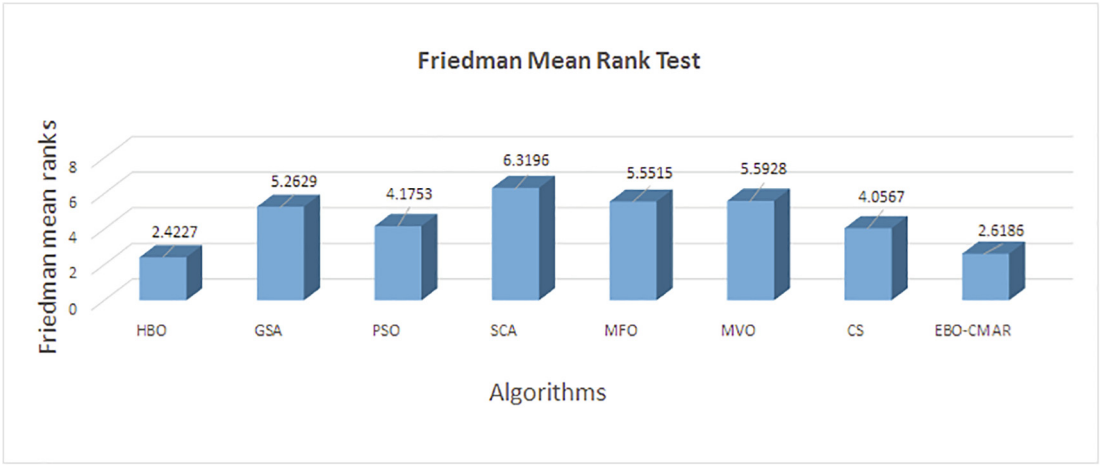


Fig. 18. Result of Friedman Mean Rank Test for all 97 functions. Smaller bars/outcomes on the bars denote the higher ranks.

tions; however the second function has been removed due to the unstable behavior. These functions have very challenging landscapes and in this section, we evaluate the performance of HBO on these functions against the same algorithms, which we have used in previous experiments. It is important to mention here that the set of comparative algorithms also includes the winner of CEC-BC-2017 called EBO-CMAR. The details of the functions including their names, number of design variables, range of the variables, and global optimum are presented in Table 6. The average and standard deviation of 30 runs of the algorithms for all functions of CEC-BC-2017 are presented in Table 12.

The good performance of HBO is evident from the results. Although, HBO could not beat EBO-CMAR but HBO is better than the other algorithms and has comparable results to EBO-CMAR in many cases. For instance, HBO achieves 1st rank for $f73, f76, f87, f89, f92$, and $f94$. Furthermore, HBO performs better than EBO-CMAR for $f93$. Moreover, the results of HBO does not have any significant difference from the results of EBO-CMAR for $f70, f71, f74, f78, f83$, and $f84$. To have better insights, Friedman mean rank test is performed and the results are reported in Fig. 15. The results of Friedman test shows that HBO attains 2nd rank after EBO-CMAR.

The Friedman rank test gives the average ranking of the algorithms; however, it is important to see the statistical significance of the difference between the ranks of the algorithms. Using the

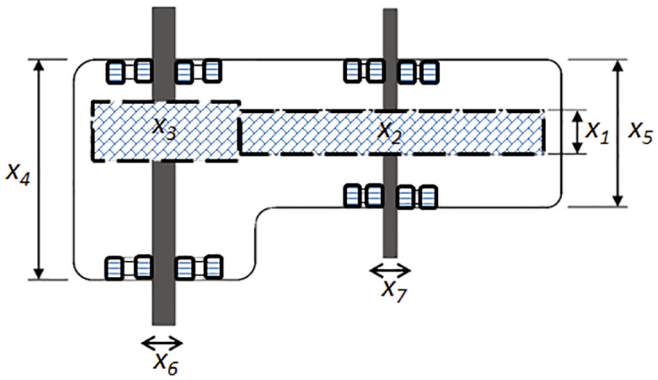


Fig. 19. Schematic views of speed reducer design.

statistics provided by the Friedman rank test, we perform multiple comparison test based on Bonferroni method (Zar, 1999) to find the statistical significance of the difference in the results. According to the Bonferroni method, the difference between two algorithms is statistically significant if the difference in average ranking of the algorithms is larger than comparison interval defined by the significance level (α) of the test. Two pair-wise com-

parison tests are performed. The first test is performed by considering HBO as a reference/control algorithm at the significance level of 0.05 and second test is performed by considering EBO-CMAR as the reference/control algorithm at the significance level of 0.15. The results of the first and second tests are depicted in Figs. 16 and 17, respectively. In both figures, the mean ranks of the algorithms are highlighted by the circles and comparison intervals are highlighted through the bars. The dotted red circle and bar denote the mean rank and comparison interval of the control algorithm, respectively. The bars in plain black are of the algorithms having a statistically significant difference from the control algorithm and the bars in plain grey are of the algorithms having a statistically insignificant difference from the control algorithm. The comparison in Fig. 16 shows that the rank of HBO is statistically equivalent to CS and EBO-CMAR at the significance level of 0.05; however, the comparison in Fig. 17 shows that the rank of EBO-CMAR is statistically equivalent to HBO at the significance level of 0.15. It is worth mentioning here that if the significance level is set to 0.05 the rank of EBO-CMAR becomes statistically equivalent to CS as well but for any of the considered significance levels, the difference in ranks of HBO and EBO-CMAR is insignificant. Conclusively, according to Friedman mean rank test HBO attains 2nd best rank; however, the difference in ranks of HBO and EBO-CMAR is not statistically significant from the perspective of HBO as well as EBO-CMAR.

4.6. Overall mean ranks of the algorithms

In this subsection, the Friedman test is performed to find the overall mean rank of the algorithms for all 97 benchmark functions including unimodal, multimodal, and CEC-BC-2017 benchmark functions. The results of Friedman test are plotted in Fig. 18. As we can see, the mean rank value of HBO is minimum, which indicates that HBO attains 1st rank.

5. Demonstration of applicability on mechanical engineering problems

In order to demonstrate the applicability of the proposed algorithm on real world problems, we picked 3 well-known engineering problems: Speed reducer design problem, multiple disc clutch brake design problem, and rolling element bearing design problem. In the literature, these problems have extensively been used to evaluate the performance of optimization algorithms on real world problems. As the engineering problems are constrained optimization problems, a simple constraint handling technique called death penalty (scalar penalty function) (Coello, 2002) is used. In this approach, solutions are penalized on violating a constraint.

5.1. Speed reducer design problem

This problem was presented by Golinski (1970). The problem of speed reducer design deals with designing a simple gear box that is used between the engine and the propeller in light aircraft. The goal is to minimize the weight of a speed reducer so that the engine and propeller can rotate efficiently. The problem involves constraints on stresses in the shafts, transverse deflection of the shafts, surface stress and bending stress of the gear teeth. Schematic view of the problem is depicted in Fig. 19 and the mathematical definition of the problem is presented below:

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) \\
& -1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) \\
& +0.7854\left(x_4x_6^2 + x_5x_7^2\right) \\
\text{subjectto} \quad & g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leqslant 0, \quad g_2(x) = \frac{397.5}{x_1x_2^2x_2^2} - 1 \leqslant 0 \\
& g_3(x) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \leqslant 0, \quad g_4(x) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \leqslant 0 \\
& g_5(x) = \frac{\left[\left(745x_4/x_2x_3\right)^2 + 16.9\times10^6\right]^{1/2}}{110x^3} - 1 \leqslant 0 \\
& g_6(x) = \frac{\left[\left(745x_5/x_2x_3\right)^2 + 157.5\times10^6\right]^{1/2}}{85x_7^3} - 1 \leqslant 0 \\
& g_7(x) = \frac{x_2x_3}{40} - 1 \leqslant 0, \quad g_8(x) = \frac{5x_2}{x_1} - 1 \leqslant 0 \\
& g_9(x) = \frac{x_1}{12x_2} - 1 \leqslant 0, \quad g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leqslant 0 \\
& g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leqslant 0 \\
\text{where} \quad & 2.6 \leqslant x_1 \leqslant 3.6, \quad 0.7 \leqslant x_2 \leqslant 0.8, \quad 17 \leqslant x_3 \leqslant 28, \\
& 7.3 \leqslant x_4 \leqslant 8.3, \quad 7.3 \leqslant x_5 \leqslant 8.3, \quad 2.9 \leqslant x_6 \leqslant 3.9, \\
& 5.0 \leqslant x_7 \leqslant 5.5
\end{aligned}
\tag{15}
$$

Table 13
Comparison of statistical results obtained from various algorithms for the speed reducer design problem.

Algo	Statistical results				NFEs
	Worst	Mean	Best	Std	
SC	3009.964736	3001.758264	2994.744241	4.000000	54,456
PSO-DE	2996.348204	2996.348174	2996.348167	0.000006	54,350
DELC	**2994.471066**	**2994.471066**	**2994.471066**	0.000000	30,000
DEDS	**2994.471066**	**2994.471066**	**2994.471066**	0.000000	30,000
HEAA	2994.752311	2994.613368	2994.499107	0.070000	40,000
MDE	NA	2996.367220	2996.356689	0.008200	24,000
ABC	NA	2997.058000	2997.058000	0.000000	30,000
WCA	2994.505578	2994.474392	**2994.471066**	0.007400	15,150
HBO	**2994.471066**	**2994.471066**	**2994.471066**	0.000000	**9,010**

Table 14
Comparison of the best solution obtained by all algorithms for speed reducer design problem.

DV	DEDS	DELC	HEAA	MDE	PSO-DE	WCA	HBO
X1	3.50000	3.50000	3.50002	3.50001	3.50000	3.50000	3.50000
X2	0.70000	0.70000	0.70000	0.70000	0.70000	0.70000	0.70000
X3	17.00000	17.00000	17.00001	17.00000	17.00000	17.00000	17.00000
X4	7.3 + 9	7.3 + 9	7.30043	7.30016	7.30000	7.30000	7.30000
X5	7.71532	7.71532	7.71538	7.80003	7.80000	7.71532	7.71532
X6	3.35021	3.35021	3.35023	3.35022	3.35021	3.35021	3.35021
X7	5.28665	5.28665	5.28666	5.28669	5.28668	5.28665	5.28665
f(X)	**2994.471066**	**2994.471066**	2994.499107	2996.356689	2996.348167	**2994.471066**	**2994.471066**

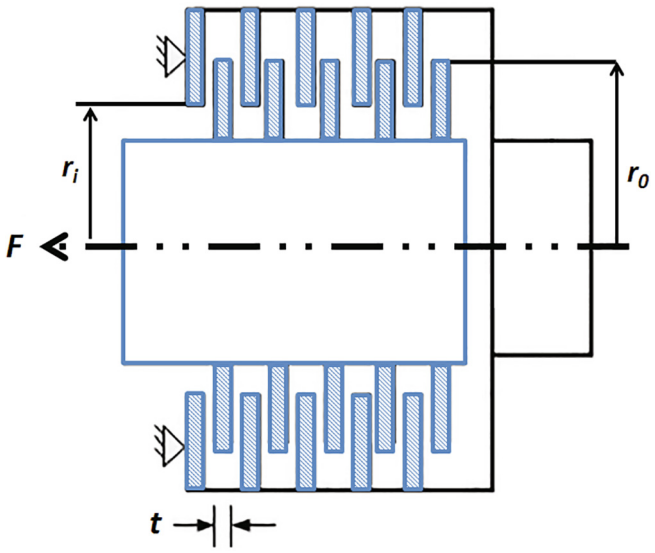


Fig. 20. Schematic views of multiple disc clutch brake design problem.

By executing the proposed Algorithm 30 times in a row, the statistical results are produced. In Table 13, the results are compared with some well-known algorithms used to solve the problem of speed reducer design. The names of these algorithms are: Society and civilization (SC) algorithm (Ray & Liew, 2003), hybrid PSO and DE (PSO-DE) (Liu, Cai, & Wang, 2010), differential evolution with level comparison (DELC) (Wang & po Li, 2009), differential evolution with dynamic stochastic selection (DEDS) (Zhang, Luo, & Wang, 2008), hybrid evolutionary algorithm and adaptive constraint-handling (HEAA) (Wang, Cai, Zhou, & Fan, 2008), modified differential evolution (MDE) (Mezura-Montes, Velázquez-Reyes, & Coello, 2006), artificial bee colony (ABC) (Karaboga & Basturk, 2007), and water cycle algorithm (WCA) (Eskandar, Sadollah, Bahreininejad, & Hamdi, 2012). The results demonstrate that HBO shares 1st position with DELC and DEDS; however, HBO evaluates the objective function 9,010 times which is much lesser than the function evaluations of DEDS and DELC. In order to regenerate the results, it is recommended to use population of size = 40, iterations = 230 and a search agent should be penalized by a fitness value $1E+16$ on violating a constraint.

The values of the design variables (DV) of the best solutions obtained by all algorithms are presented in Table 14. Speed reducer design problem involves 11 constraints and HBO satisfies all of them with values $g = (-0.0739, -0.1979, \ -0.4991, -0.9046, -9.089E-12, -2.459E-12, -0.7025, -6.08E-13, -0.7958, -0.0513, -1.3095E-11)$.

5.2. Multiple disc clutch brake design problem

This problem deals with minimizing the mass of a multiple disc clutch brake. The problem belongs to the category of discrete optimization, which involves 5 discrete design variables: inner radius $r_i \in \{60, 61, \ldots, 80\}$, outer radius $r_o \in \{90, 91, \ldots, 110\}$, thickness of the disc $t \in \{1, 1.5, \ldots, 3\}$, actuating force $F \in \{6000, 610, \ldots, 1000\}$ and number of friction surfaces $Z \in \{2, 3, \ldots, 9\}$. The schematic view of the problem is illustrated in Fig. 20 and the mathematical definition of the problem is given below:

$$
\begin{aligned}
&\underset{x}{\text{minimize}} \quad f(\vec{z}) = \pi(r_0^2 - r_i^2)t(Z+1)p \\
&\text{subjectto} \quad g_1(\vec{z}) = r_0 - r_i - \Delta r \geqslant 0, \quad g_2(\vec{z}) = l_{max} - (Z+1)(t+\delta) \geqslant 0 \\
&\qquad\qquad g_3(\vec{z}) = p_{max} - p_{rz} \geqslant 0, \quad g_4(\vec{z}) = p_{max} v_{stmax} - p_r z v_{st} \geqslant 0 \\
&\qquad\qquad g_5(\vec{z}) = v_{stmax} - v_{st} \geqslant 0, \quad g_6(\vec{z}) = T_{max} - T \geqslant 0 \\
&\qquad\qquad g_7(\vec{z}) = M_h - sM_s \geqslant 0, \quad g_8(\vec{z}) = T \geqslant 0 \\
&\text{where} \quad \mu = 0.5, l_z = 55 kgmm^2, n = 250 rpm, \Delta r = 20 mm, M_s = 40 Nm, p_{max} = 1 MP_a, \\
&\qquad\qquad T_{max} 15 s, F_{max} = 1000 N, s = 1.5, M_f = 3 Nm, v_{srmax} = 10, l_{max} = 30 mm, \\
&\qquad\qquad r_{imin} = 60 \tfrac{m}{s}, r_{0min} = 90, r_{0max} = 110, r_{imax} = 80, t_{max} = 3, F_{min} = 600, \\
&\qquad\qquad Z_{min} = 2, Z_{max} = 9, t_{min} = 1.5
\end{aligned}
\tag{16}
$$

In literature, this problem has been solved by using ABC (Karaboga & Basturk, 2007), TLBO (Rao et al., 2011), non-dominated sorting genetic algorithm (NSGA-II) (Deb, Agrawal, Pratap, & Meyarivan, 2000) and adaptive particle swarm optimization (APSO) (Zhan, Zhang, Li, & Chung, 2009). By considering the importance of the number of function evaluations (NFEs) in solving real world problems, two sets of statistical results are obtained. First set of results, labeled as HBO1, is obtained for iterations = 25 and population size = 20, which is to prove the superiority of HBO over the competitors and second set of results is obtained for iterations = 50, which demonstrates the capability of HBO to always converge at the best location. The statistical results are presented and compared in Table 15, which demonstrate the superiority of HBO over the other algorithms. The results of HBO1 shows that our proposed algorithm gives better results using fewest

Table 15
Comparison of statistical results obtained from various algorithms for the multiple disc clutch brake design problem.

Algo	Statistical results				NFEs
	Worst	Mean	Best	Std	
ABC	0.3528640	0.3247510	0.3136570	5.40E−01	>900
TLBO	0.3920710	0.3271660	0.3136570	6.70E−01	>900
NSGA-II	N/A	N/A	0.4704000	N/A	N/A
APSO	0.7163130	0.5068290	0.3371810	9.67E−02	2000
HBO1	**0.3396313**	**0.3164207**	**0.3136566**	7.36E−03	**495**
HBO2	**0.3136566**	**0.3136566**	**0.3136566**	5.85E−17	**970**

Table 16
Comparison of the best solution obtained by all algorithms for multiple disc clutch brake design problem.

DV	Best solution DV					f(X)
	X1	X2	X3	X4	X5	
NSGA-II	70	90	1.5	1000	3	0.4704
TLBO	70	90	1	810	3	0.313656
NSGA-II	70	90	1.5	1000	3	0.4704
APSO	76	96	1	840	3	0.3371821
HBO1	70	90	1	1000	3	**0.3136566**
HBO2	70	90	1	1000	3	**0.3136566**

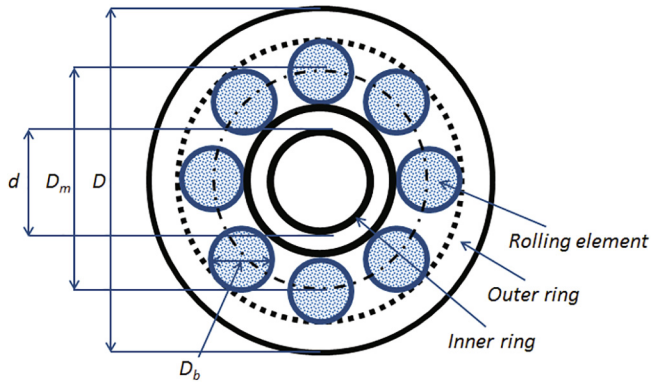


Fig. 21. Schematic views of Rolling element bearing design problem.

objective function evaluations; however, HBO2 shows that the proposed algorithm can always converge at the best position by evaluating the objective function 970 times.

For the sake of illustration the values of design variables obtained by all algorithms are compared in Table 16. Moreover, HBO fulfills all the constraints with values $g = (0, 24, 0.900528161, 9.790581597, -7.89469659, 3.352706711, 60.625, 11.64729329)$. In order to regenerate the results, it is recommended to use penalty $= 1.00E+07$ for both HBO1 and HBO2.

5.3. Rolling element bearing design problem

This problem deals with maximizing the dynamic load bearing capacity of rotating machinery. The design of rolling element bearing involves ten variables. Five of them are used in the objective function, known as the diameter of the balls D_b, mean diameter D_m, number of balls Z (a discrete variable), curvature radius coefficient of inner raceway groove $fi = \frac{r_i}{D_b}$ and curvature radius coefficient of outer raceway groove $fo = \frac{r_o}{D_b}$, where r_i and r_o represent the inner and outer ring groove curvature radii, respectively. Rest of the five parameters $(K_{D_{min}}, K_{D_{max}}, \epsilon, e, \eta)$ are used in constraints. The schematic diagram of rolling element bearing design is given in Fig. 21 and its mathematical definition is presented below:

$$\begin{aligned}
&\underset{x}{\text{maximize}} && f(\vec{x}) = C_d = \begin{cases} f_c Z^{2/3} D_b^{1.8} & \text{if} D_b, \leqslant 25.4mm \\ 3.467 f_c Z^{2/3} D_b^{1.4}, & otherwise \end{cases} \\
&\text{subjectto} && g_1(\vec{x}) = \frac{\phi_0}{2Sin^{-1}(D_b/D_m)} - Z + 1 \geqslant 0, \quad g_2(\vec{x}) = 2D_b - K_{D_{min}}(D-d) \geqslant 0 \\
& && g_3(\vec{x}) = K_{D_{min}}(D-d) - 2D_b \geqslant 0, \quad g_4(\vec{x}) = \eta B_w - D_b \leqslant 0 \\
& && g_5(\vec{x}) = D_m - 0.5(D+d) \geqslant 0, \quad g_6(\vec{x}) = (0.5+e)(D+d) \geqslant 0 \\
& && g_7(\vec{x}) = 0.5(D - D_m - D_b) - \epsilon D_b \geqslant 0 \\
& && g_8(\vec{x}) = f_i - 0.515 \geqslant 0, g_9(\vec{x}) = f_0 - 0.515 \geqslant 0 \\
&\text{where} && f_c = 37.91\{1 + [1.04(\tfrac{1-\gamma}{1+\gamma})^{1.72}(\tfrac{f_i(2f_0-1)}{f_0(2f_i-1)})^{0.41}]^{10/3}\}^{-0.3}[\tfrac{f_i^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}](\tfrac{2f_i}{2f_i-1})^{0.41} \\
& && \phi_0 = 2\pi - 2\cos^{-1}(\tfrac{[(D-d)/2-3T/4]^2+[D/2-T/4-D_b]^2-[d/2+T/4]^2}{2[(D-d)/2-3T/4][D/2-T/4-D_b]}), \\
& && \gamma = D_b/D_m, f_i = r_i/D_b, f_0 = r_0/D_b, T = D-d-2D_b, D = 160, d = 90, B_w = 30, \\
& && 0.4 \leqslant K_{D_{min}} \leqslant 0.5, 0.6 \leqslant K_{D_{max}} \leqslant 0.7, 0.3 \leqslant \epsilon \leqslant 0.4, 0.02 \leqslant e \leqslant 0.1, 0.6 \leqslant \eta \leqslant 0.85, \\
& && 0.5(D+d) \leqslant D_m \leqslant 0.6(D+d), 0.15(D-d) \leqslant D_b \leqslant 0.45(D-d)
\end{aligned} \tag{17}$$

This problem has been solved in the literature by using GA4 (Gupta, Tiwari, & Nair, 2007), ABC (Karaboga & Basturk, 2007), TLBO (Rao et al., 2011), WCA (Eskandar et al., 2012), MBA (Sadollah, Bahreininejad, Eskandar, & Hamdi, 2013), PSO-TVAC (Muneender & Vinodkumar, 2012) and DAPSO-GA (Zhu et al., 2019). The performance of HBO is compared with these algorithms in Table 17. Two sets of outcomes are produced, one showing the superiority over the competitors and the other showing the capacity to converge in the best position. First set of outcomes (HBO1) is generated by setting population size = 20 and iterations = 100; however, for second set of results (HBO2) the iterations are increased to 400 to achieve best results. The results demonstrate that HBO1 gives best statistics by evaluating the objective function fewer times than the others. In order to further improve the results and always converge at the same best position, the number of functions evaluations are increased to 7620. It is important to highlight here, the best objective value found by WCA is infeasible because Z denoted as x_3 is a discrete variable and represents the number of balls; however; WCA treats it as a continuous variable and finds its value as 11.00103, which is not possible. The values of design variables found by all algorithms and their comparison is presented in Table 18.

Both sets of results fully satisfy the constraints. The values of constraints for the best solution obtained by HBO1 is as $g = (0.000921088, 8.682122802, 4.709300943, -3.17371473, 0.719391861, 16.53579039, 0.001450935, 0, 8.00457E-07, 3.93907E-05, 2.22423E-05)$ and for the best solution obtained by HBO2 is as $g = (0.001159423, 12.1532798, 6.153256824, -3.390871387, 0.722718447, 11.16033474, 1.3E-14, 0, 0, 0, 0)$. Please note the value of the 4th constraint is negative because it has to be lesser or equal to 0. In order to regenerate the results, it is recommended to use penalty $1E+12$ for HBO1 and $1E+21$ for HBO2 (if the problem is transformed into a minimization problem).

6. Optimization capabilities, practical implications, and research directions

In this study, a novel meta-heuristic inspired by corporate rank hierarchy and the interaction between the individuals in that hierarchy is proposed. Unlike many other meta-heuristics, the relative difference in the fitness of the search agents is utilized to arrange them in a hierarchy and the concept of min/max heap is utilized to make interaction possible between them and preserve their relative difference. Moreover, a parameter called Gamma (γ) is designed, which helps to avoid premature convergence by enabling the algorithm to escape local optima. The proposed algorithm has demonstrated very good optimization capabilities, such as:

- **High exploration in early iterations**: It is due to the higher probability of selection of the self-contribution equation (Eq. 6) and diversity incorporation by updating the position of a search agent with respect to a randomly selected colleague (node at the same level in heap) through the Eq. (5).

Table 17
Comparison of statistical results obtained from various algorithms for the rolling element bearing design problem.

Algo	Statistical results				NFEs
	Worst	Mean	Best	Std	
MBA	NA	NA	81843.68625	2.12E+02	15100
PSO-TVAC	41,130.57	76,442.50	81,859.74	1.19E+04	3750
DAPSO-GA	79,834.78	81,066.43	81,859.81	7.43E+02	3650
GA4	NA	NA	81843.3	NA	225,000
ABC	78897.81	81496	81859.74	6.90E−01	10,000
TLBO	80807.85	81438.98	81859.74	6.60E−01	10,000
WCA	83942.71	83847.16	85538.48	4.88E+02	3,950
HBO1	**84433.04**	**85197.34**	**85532.57**	4.02E+02	**1,920**
HBO2	**85533.18**	**85533.18**	**85533.18**	2.15E−06	**7,620**

Table 18
Comparison of the best solution obtained by all algorithms for rolling element bearing design problem.

Algo	Best solution DV										f(x)
	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	
MBA	125.7153	21.4233	11	0.515	0.515	0.488805	0.627829	0.300149	0.097305	0.64095	81,843.69
PSO-TVAC	125.7191	21.4256	11	0.515	0.515	0.4169	0.7	0.3	0.1	0.6001	81,859.74
DAPSO-GA	125.7191	21.4256	11	0.515	0.515	0.4	0.7	0.3	0.0474	0.6	81,859.81
GA4	125.7171	21.4230	11	0.5150	0.5150	0.4159	0.6510	0.3000	0.0223	0.7510	81843.3
TLBO	125.7191	21.4256	11	0.5150	0.5150	0.4243	0.6339	0.3000	0.0689	0.7995	81859.74
WCA	125.7212	21.4233	**11.0010**	0.5150	0.5150	0.4015	0.6590	0.3000	0.0400	0.6000	85538.48
HBO1	125.7193919	21.42322	11	0.515	0.515001	0.488062	0.679368	0.300013	0.069021	0.608317	**85532.57**
HBO2	125.7227184	21.4233	11	0.515	0.515	0.438476	0.699998	0.3	0.047532	0.601081	**85533.18**

- **Emergence of exploitation capability**: HBO emerges exploitative behavior with the course of iterations due to the linear increase in the probability of selection of Eq. (8) designed for interaction with immediate boss/parent node (node in the upper level in heap).
- **Mechanism to balance the exploration and exploitation**: The balance between exploration and exploitation is attained by decreasing the probability of selection (Eq. 7) of the self-contribution equation and increasing the probability of selection (Eq. 8) of the equation for interaction with the parent node.
- **Convergence assurance**: The convergence is assured by leading all solutions directly or indirectly towards the root node of the heap. Moreover, the step towards a random colleague is only accepted if it improves the current position of the search agent.
- **Escaping local optima and premature convergence avoidance**: In addition to the controlled applicability of equations, the parameter gamma (Eq. 3) helps search agents to escape local optima and avoid premature convergence by cyclically rotating in the range [0, 2] because when its value ranges in [1, 2] the search agent may jump out of the local vicinity.

In addition to above good optimization capabilities, we find through experiments that the convergence speed of HBO compromises for some functions, which is due to the nature of the parameter γ and looking for a good balance between the exploration and exploitation. As a part of our future work, we plan to work on the improvement of the convergence speed without compromising the above-stated qualities of the proposed algorithm. However, a few suggestions for the researchers to come up with an improved variant are presented below:

- The range of the parameter γ can be squeezed to $[0, 1]$ with the course of iterations. This may increase the convergence speed but at the expense of poor premature convergence avoidance capability in the later iterations.
- In HBO, the self-contribution phase is kept very simple in which the current position of the search agent is retained; however, one may improve it by giving random walk to the search agent or incorporating the concept of levy flight etc. It is also possible to model the concept of self-contribution differently.
- Every search agent updates its position in each iteration except the search agent associated with the root node called the boss. The convergence speed or even the overall performance may be improved by giving some kind of movement to the root node as well.
- The probabilities of selection of different equations may be calculated in some other way giving better convergence speed.

The meta-heuristics have demonstrated excellent performance on complex optimization problems. Based on the balanced optimization capabilities, we suggest research community to use HBO for the feature selection to remove redundancy and useless information from the datasets, error minimization in regression problems, optimizing the architecture of a neural network, training of the neural network, maximizing the margin in support vector machines, tuning the parameters for several machine learning algorithms, solving the engineering optimization problems such as welded beam design problem, reinforced concrete beam problem, pressure vessel design problem, tension–compression spring problem, three-bar truss design problem, speed-reducer design problem, and stepped cantilever beam problem etc. However, the practical implications would include the slower convergence speed which might not be afforded in real-time optimization or time-constrained optimization problems.

The mapping of HBO is kept very simple in this paper. However, there are a lot of potential areas, which should be worked on to come up with a variant of the proposed algorithm. A few directions are given below:

- It is possible to try binary or higher degree heaps.
- One can investigate the interaction with random officials or interaction between the subordinates of the same boss rather than the colleagues.
- The heap can be reconstructed after updating the entire population instead of calling heapify right after updating each search agent.

Hybridization of HBO with other well-known existing optimization algorithms and development of a multi-objective version of HBO would also be the good directions for the researchers. Finding real-world applications where the advantages of applying HBO are evident and taking into account their strengths and limitations can also be a significant future direction.

7. Conclusion

This paper proposes a novel human-behaviour based meta-heuristic, which utilizes a heap data structure to map the concept of corporate rank hierarchy (CRH) to solve optimization problems. In the area of nature-inspired optimization algorithms, the use of heap data structure to map an inspiration is a unique idea in itself. To promote and maintain an appropriate balance between exploration and exploitation, the mathematical equations are carefully derived from three major CRH activities. Moreover, to escape local optima and avoid premature convergence without compromising the exploitation capability of the proposed algorithm, a self-adaptive parameter is designed, which periodically incorporates the exploration and exploitation capabilities. To verify the effectiveness of the proposed algorithm, we have compared it with 7 state-of-the-art and high-performance optimization algorithms including the winner of CEC-BC-2017 for solving a wide set of diverse benchmark functions including unimodal, multimodal, and CEC-BC-2017 benchmark functions. The experimental results for the unimodal and multimodal functions show that HBO outperforms the comparative algorithms in most cases and attains 1st

rank according to Friedman mean rank test. Moreover, the performance of HBO for CEC-BC-2017 is shown to be comparable to EBO-CMAR and better than the other algorithms. Although, HBO attains 2nd rank for CEC-BC-2017 functions it is shown by using the Bonferroni method that the rank of HBO is not significantly different from EBO-CMAR. Furthermore, the overall rank of HBO for all 97 benchmark functions is shown to be 1st according to the Friedman mean rank test. Finally, the ability to solve real-world problems is demonstrated by solving three mechanical constrained engineering problems. The results provide some evidence that HBO can be used effectively to solve real-world optimization problems. Given the preliminary nature of the proposed algorithm, we firmly believe that many extensions, improvements and evaluations are anticipated in the future. The superiority of HBO nominates it for solving real-world optimization problems with challenging landscapes arising in intelligent and expert systems.

CRediT authorship contribution statement

Qamar Askari: Conceptualization, Methodology, Software, Writing - original draft, Formal analysis, Investigation. **Mehreen Saeed:** Supervision, Writing - review & editing, Validation, Project administration. **Irfan Younas:** Supervision, Writing - review & editing, Validation, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmadi-Javid, A. (2011). Anarchic society optimization: A human-inspired method. In *2011 IEEE congress of evolutionary computation (CEC)*. IEEE.

Ahmady, G. A., Mehrpour, M., & Nikooravesh, A. (2016). Organizational structure. *Procedia - Social and Behavioral Sciences, 230*, 455–462.

Alsattar, H. A., Zaidan, A. A., & Zaidan, B. B. (2019). Novel meta-heuristic bald eagle search optimisation algorithm. *Artificial Intelligence Review*.

Arora, S., & Singh, S. (2018). Butterfly optimization algorithm: A novel approach for global optimization. *Soft Computing, 23*(3), 715–734.

Askari, Q., Younas, I., & Saeed, M. (2020). Political optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowledge-Based Systems*, 105709.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *2007 IEEE congress on evolutionary computation*. IEEE.

Awad, N.H., P.S.J.L.B.Q., Ali, M. Z. (2017). Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. In *2017 IEEE congress on evolutionary computation (CEC)*.

Balochian, S., & Baloochian, H. (2019). Social mimic optimization algorithm and engineering applications. *Expert Systems with Applications, 134*, 178–191.

Borji, A. (2007). A new global optimization algorithm inspired by parliamentary political competitions. In *Mexican international conference on artificial intelligence* (pp. 61–71). Heidelberg: Springer, Berlin.

Brammya, G., Praveena, S., Preetha, N. S. N., Ramya, R., Rajakumar, B. R., & Binu, D. (2019). Deer hunting optimization algorithm: A new nature-inspired meta-heuristic paradigm. *The Computer Journal*.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191*(11–12), 1245–1287.

Crawford, B., Soto, R., Cabrera, G., Salas-Fernández, A., & Paredes, F. (2019). Using a social media inspired optimization algorithm to solve the set covering problem. In *International conference on human-computer interaction* (pp. 43–52). Springer.

Daskin, A., & Kais, S. (2011). Group leaders optimization algorithm. *Molecular Physics, 109*(5), 761–772.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel problem solving from nature PPSN VI* (pp. 849–858). Heidelberg: Springer, Berlin.

Dhiman, G., & Kumar, V. (2019). Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowledge-Based Systems, 165*, 169–196.

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (pp. 1470–1477). IEEE, Vol. 2.

Eberhart, R., & Kennedy, J. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (pp. 1942–1948). IEEE, Vol. 4.

Erol, O. K., & Eksin, I. (2006). A new optimization method: Big bang–big crunch. *Advances in Engineering Software, 37*(2), 106–111.

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm – a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures, 110–111*, 151–166.

Fadakar, E., & Ebrahimi, M. (2016). A new metaheuristic football game inspired algorithm. In *2016 1st Conference on swarm intelligence and evolutionary computation*. (CSIEC). IEEE.

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems, 191* 105190.

Flores, J. J., López, R., & Barrera, J. (2011). Gravitational interactions optimization. In *Lecture notes in computer science* (pp. 226–237). Heidelberg: Springer, Berlin.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*(1), 86–92.

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17* (12), 4831–4845.

Golinski, J. (1970). Optimal synthesis problems solved by means of nonlinear programming and random methods. *Journal of Mechanisms, 5*(3), 287–309.

Gupta, S., Tiwari, R., & Nair, S. B. (2007). Multi-objective design optimisation of rolling bearings using genetic algorithms. *Mechanism and Machine Theory, 42* (10), 1418–1443.

Han, K.-H., & Kim, J.-H. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation, 6* (6), 580–593.

Harifi, S., Khalilian, M., Mohammadzadeh, J., & Ebrahimnejad, S. (2019). Emperor penguins colony: A new metaheuristic algorithm for optimization. *Evolutionary Intelligence, 12*(2), 211–226.

Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., & Mirjalili, S. (2019). Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems, 101*, 646–667.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.

Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*(1), 66–73.

Huning, A. (1976). *ARSP: Archiv für Rechts- und Sozialphilosophie/ Archives for Philosophy of Law and Social, 62*(2), 298–300.

Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation, 44*, 148–175.

Jones, G. R. (2019). *Organizational theory, design, and change*. Upper Saddle River, NJ: Pearson.

Karaboga, D., & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In *Lecture notes in computer science* (pp. 789–798). Heidelberg: Springer, Berlin.

Kashan, A. H. (2014). League championship algorithm (LCA): An algorithm for global optimization inspired by sport championships. *Applied Soft Computing, 16*, 171–200.

Khishe, M., & Mosavi, M. (2020). Chimp optimization algorithm. *Expert Systems with Applications, 149*, 113338.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection (Complex adaptive systems). A Bradford Book.

Kumar, A., Misra, R. K., & Singh, D. (2017). Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 1835–1842). IEEE.

Kumar, M., Kulkarni, A. J., & Satapathy, S. C. (2018). Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology. *Future Generation Computer Systems, 81*, 252–272.

Lampinen, J., & Storn, R. (2004). Differential evolution. In *New optimization techniques in engineering* (pp. 123–166). Heidelberg: Springer, Berlin.

Li, S., Chen, H., Wang, M., Heidari, A. A., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems*.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*(2), 629–640.

Lv, W., Xie, Q., Liu, Z., Zhang, X., Luo, S., & Cheng, S. (2010). Election campaign algorithm. In *2010 2nd International Asia conference on informatics in control, automation and robotics (CAR 2010)*. IEEE.

Mahmoodabadi, M., Rasekh, M., & Zohari, T. (2018). Tga: Team game algorithm. *Future Computing and Informatics Journal, 3*(2), 191–199.

Masadeh, R., A., B., & Sharieh, A. (2019). Sea lion optimization algorithm. *International Journal of Advanced Computer Science and Applications, 10*(5).

Melvix, J. L. (2014). Greedy politics optimization: Metaheuristic inspired by political strategies adopted during state assembly elections. In *2014 IEEE international advance computing conference (IACC)*. IEEE.

Mezura-Montes, E., Velázquez-Reyes, J., & Coello, C. C. (2006). Modified differential evolution for constrained optimization. In *2006 IEEE international conference on evolutionary computation* (pp. 25–32). IEEE.

Milton, F. (1939). A correction: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 34*(205), 109.
Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.
Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.
Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2015). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*(2), 495–513.
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
Moosavian, N., & Roodsari, B. K. (2014). Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. *Swarm and Evolutionary Computation, 17*, 14–24.
Moosavi, S. H. S., & Bardsiri, V. K. (2019). Poor and rich optimization algorithm: A new human-based and multi populations algorithm. *Engineering Applications of Artificial Intelligence, 86*, 165–181.
Morais, R. G., Mourelle, L. M., & Nedjah, N. (2018). Hitchcock birds inspired algorithm. In *Computational collective intelligence* (pp. 169–180). Springer International Publishing.
Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms.
Muneender, E., & Vinodkumar, D. M. (2012). Particle swarm optimization with time varying acceleration coefficients for congestion management. In *2012 IEEE conference on sustainable utilization and development in engineering and technology (STUDENT)*. IEEE.
Nabil, E. (2016). A modified flower pollination algorithm for global optimization. *Expert Systems with Applications, 57*, 192–203.
Narayanan, A., & Moore, M. (1996). Quantum-inspired genetic algorithms. In *Proceedings of IEEE international conference on evolutionary computation*. IEEE.
Ramezani, F., & Lotfi, S. (2013). Social-based algorithm (SBA). *Applied Soft Computing, 13*(5), 2837–2856.
Rao, R., Savsani, V., & Vakharia, D. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*(3), 303–315.
Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.
Ray, T., & Liew, K. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation, 7*(4), 386–396.
Razmjooy, N., Khalilpour, M., & Ramezani, M. (2016). A new meta-heuristic optimization algorithm inspired by FIFA world cup competitions: Theory and its application in PID designing for AVR system. *Journal of Control, Automation and Electrical Systems, 27*(4), 419–440.
Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing, 13*(5), 2592–2612.
Salgotra, R., & Singh, U. (2019). The naked mole-rat algorithm. *Neural Computing and Applications, 31*(12), 8837–8857.
Salih, S. Q., & Alsewari, A. A. (2019). A new algorithm for normal and large-scale optimization problems: Nomadic people optimizer. *Neural Computing and Applications*.
Satapathy, S., & Naik, A. (2016). Social group optimization (SGO): A new population evolutionary optimization technique. *Complex & Intelligent Systems, 2*(3), 173–203.
Shadravan, S., Naji, H., & Bardsiri, V. K. (2019). The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence, 80*, 20–34.
Shastri, A. S., Jagetia, A., Sehgal, A., Patel, M., & Kulkarni, A. J. (2019). Expectation algorithm (exa): A socio-inspired optimization methodology. In *Socio-cultural Inspired Metaheuristics* (pp. 193–214). Springer.
Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702–713.
Singh, P. R., Elaziz, M. A., & Xiong, S. (2019). Ludo game-based metaheuristics for global and engineering optimization. *Applied Soft Computing, 84* 105723.
Wang, L., & po Li, L. (2009). An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization, 41*(6), 947–963.
Wang, Y., Cai, Z., Zhou, Y., & Fan, Z. (2008). Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization, 37*(4), 395–413.
Xu, Y., Cui, Z., & Zeng, J. (2010). Social emotional optimization algorithm for nonlinear constrained optimization problems. In *Swarm, Evolutionary, and Memetic Computing* (pp. 583–590). Heidelberg: Springer, Berlin.
Yadav, A. et al. (2019). Aefa: Artificial electric field algorithm for global optimization. *Swarm and Evolutionary Computation, 48*, 93–108.
Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *Stochastic algorithms: Foundations and applications* (pp. 169–178). Heidelberg: Springer, Berlin.
Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74). Springer, Berlin Heidelberg.
Yang, X.-S., & Deb, S. (2009). Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)* (pp. 210–214). IEEE.
Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*(2), 82–102.
Zar, J. H. (1999). *Biostatistical analysis*. Pearson Education India.
Zaránd, G., Pázmándi, F., Pál, K. F., & Zimányi, G. T. (2002). Using hysteresis for optimization. *Physical Review Letters, 89*(15).
Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences, 178*(15), 3043–3074.
Zhan, Z.-H., Zhang, J., Li, Y., & Chung, H.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*(6), 1362–1381.
Zhao, F., Qin, S., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2019). A two-stage differential biogeography-based optimization algorithm and its performance analysis. *Expert Systems with Applications, 115*, 329–345.
Zhao, W., Wang, L., & Zhang, Z. (2019). Supply-demand-based optimization: A novel economics-inspired algorithm for global optimization. *IEEE Access, 7*, 73182–73206.
Zhu, H., Hu, Y., & Zhu, W. (2019). A dynamic adaptive particle swarm optimization and genetic algorithm for different constrained engineering design optimization problems. *Advances in Mechanical Engineering, 11*(3). 168781401882493.