Received January 28, 2020, accepted March 17, 2020, date of publication March 30, 2020, date of current version April 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984023

Hybrid Nature-Inspired Optimization Algorithm: Hydrozoan and Sea Turtle Foraging Algorithms for Solving Continuous Optimization Problems

DARANAT TANSUI AND ARIT THAMMANO
Computational Intelligence Laboratory, Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
Corresponding author: Arit Thammano (arit@it.kmitl.ac.th)

This work was supported by the King Mongkut's Institute of Technology Ladkrabang.

ABSTRACT In this paper, we develop a hybrid optimization algorithm inspired by the reproduction processes of hydrozoans and the foraging behavior of sea turtles for solving continuous optimization problems. Our hybrid algorithm combines the exploration capability of the hydrozoan algorithm with the exploitation capability of the sea turtle foraging algorithm. Moreover, a new adaptive crossover operator was introduced and integrated into the hybrid algorithm to further enhance exploration capability. Our hybrid algorithm was evaluated and compared to the individual algorithms and 12 state-of-the-art algorithms. Results on 21 standard benchmark functions showed that our algorithm was very effective and was among the best of the group, specifically it converged faster than the individual algorithms on most functions and reached optimal or near-optimal results on all functions.

INDEX TERMS Hybrid algorithm, hydrozoan, nature-inspired algorithm, optimization, sea turtle.

I. INTRODUCTION

Optimization is an applied science that determines the best values of parameters so as to minimize or maximize an objective function of a problem, subject to constraints on the variable values. Many real-life problems, when modeled mathematically, turn out to be optimization problems. Discovering global optimal solutions is needed in many fields, for example, science, engineering, economics and finance [1], [2].

Methods to solve optimization problems can be classified either as exact or approximate. Exact algorithms often require unacceptably large computational resources and time to find the optimum; good approximate algorithms can find near-optimal solutions using only a fraction of the resources and time. Approximate algorithms are further divided into two main groups: heuristic and metaheuristic algorithms. The metaheuristic algorithms optimize the solution of a problem by iteratively improving a candidate solution with regard to a given quality measure. The goal is to efficiently explore the search space in order to find near–optimal solutions [3], [4]. Over the past decade, nature-inspired metaheuristic algorithms have been widely used to find solutions for many complex problems in engineering and computer science [5]. Examples include Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Flower Pollination Algorithm (FPA), Invasive Weed Optimization (IWO), Biogeography-Based Optimization (BBO), Bat Algorithm (BA) and Firefly Algorithm (FA). GA is an optimization algorithm based on the Darwin's model of natural selection and evolution [6]. PSO was motivated by the social behavior of living organisms, for example, flocking birds and schooling fish. FPA was inspired by flower pollination [7]. IWO is a metaheuristic algorithm that was inspired by the spreading process in weed colonization. BBO was inspired by the emigration and immigration of living organisms into and out of various habitats based on biophysical theories of distribution of living organisms. The outstanding feature of BBO is the sharing of information between solutions, which enables them to converge rapidly to the global optimum [8].

Each of the metaheuristic algorithms has their own strengths and drawbacks; there is no method that can effectively solve all optimization problems [9]. The performance

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

of the metaheuristic algorithms depends on their ability to explore widely and find deep solutions. Therefore, many researchers have often combined two or more metaheuristic algorithms to form hybrid ones, which improve overall performance by using the strength of one to compensate for limitations of another. As a result, near global optimal solutions can be obtained more efficiently than by using one algorithm alone. The structure of a hybrid metaheuristic algorithm usually consists of a main algorithm, reinforced by an auxiliary metaheuristic. The main algorithm is an iterative generation process, that directs an auxiliary metaheuristic to explore and exploit the search space. Several hybrid algorithms are discussed in the following paragraphs.

Genetic algorithm (GA) is a widely recognized, commonly used optimization method [10]. Basically, a GA is a population-based stochastic search approach that mimics natural selection in the biological world. However, a GA has two major drawbacks: low convergence speed and easily being trapped in local optima. To remedy these drawbacks, Mahmoodabadi and Nemati [11] introduced an adaptive genetic algorithm (AGA) with new crossover and mutation operators. The new crossover operator was based on the combination of the traditional crossover and the particle swarm optimization, while the mutation operator used the concepts of the sliding mode control. The performance of this algorithm was tested using both unimodal and multimodal test functions and was found to be better than some well-established GAs. Farnad *et al.* [4] combined a GA, a particle swarm optimization (PSO) and a symbiotic organisms search (SOS) into a new hybrid method, capable of solving continuous optimization problems. The GA created offspring which inherited a better genetic structure from their parents. This helped PSO to find a better experience for each organism and this experience helped the SOS to catch a better survival opportunity in the symbiotic interaction. Together, these three component algorithms reduced the execution time and improved the solution quality. Zhang *et al.* [12] described a new hybrid PSO-GA for optimizing the performance of a biodiesel engine. Their algorithm had a two-step process, "PSO update step" followed by "GA step." The hybrid converged faster and performed better than the original GA and PSO.

Particle swarm optimization (PSO) is a widely used population-based algorithm for solving continuous optimization problems. In spite of its simplicity and efficiency, PSO often converges prematurely [13]. To overcome this drawback, many researchers hybridized PSO with other algorithms. Hakli and Uğuz [14] hybridized PSO with Levy flight. The new hybrid algorithm, called LFPSO, searched in the large spaces more effectively due to the long jumps made by particles. In this method, a jump limit was defined for each particle. The jump limit was increased, if the particles were unable to sufficiently improve at the end of an iteration. Its performance exceeded that of other PSO variants in solution quality and robustness. Javidrad and Nazari [15] hybridized PSO with simulated annealing (SA) to integrate the good exploration capability of PSO with the good exploitation capability of SA. This hybrid technique was evaluated on three criteria: stability of solution, average number of function evaluations in successful runs and average number of function evaluations, considering all successful and unsuccessful runs. On these criteria, their hybrid PSO-SA performed better than the standard PSO and Shieh's PSO-SA method. Garg [16] described another PSO-GA hybrid, in which the genetic operators, crossover and mutation, were embedded in the standard PSO to attain a balance between the exploration and exploitation. The performance on several engineering problems was better than other reported methods.

The Flower Pollination Algorithm (FPA) was developed by Yang [17], inspired by the pollination of flowering plants. Pollination occurred in two major ways: abiotic pollen dispersion, in which wind or rain spreads pollen locally and biotic pollen dispersion, in which living organisms that visit flowers carry pollen to places in the global search space. Nabil [18] developed a metaheuristic algorithm that combined FPA with the Clonal Selection Algorithm (CSA). In each iteration, the best fourteen solutions from the population were cloned proportionally to their fitness by the clonal operator. Cloning enhanced the exploitation ability. Their algorithm was able to find more accurate solutions than five other well-known optimization algorithms. Zhou *et al.* [19] modified FPA to an Elite Opposition-based Flower Pollination Algorithm (EOFPA). To enhance the exploitation and exploration abilities of FPA, two optimization strategies were incorporated into the FPA: a global elite opposition-based learning which enhanced population diversity and a local self-adaptive greedy strategy which enhanced exploitation ability. EOFPA was able to find accurate solutions quickly, with a high degree of stability.

The Invasive Weed Optimization (IWO) of Mehrabian and Lucas [20] was motivated by the ecological process of weeds colonization and distribution; it was an efficient and robust optimizer. However, it suffered from low convergence. Naidu and Ojha [21] hybridized IWO with a Quadratic Approximation (QA) operator to form QAIWO, which converged faster. QAIWO performed significantly better than IWO and GA. Cai *et al.* [22] hybridized IWO and Differential Evolution (DE). IWO was used as a local refinement step to adaptively exploit local regions around solutions with high fitness. DE, a global stochastic search algorithm, was used to find more promising solutions among elite solutions, refined by IWO. An adaptive weighted sum fitness assignment and polynomial distribution managed the reproduction and the local dispersion of IWO. More recently, a new hybrid of IWO, called IWO/BBO [8], was presented. IWO/BBO featured three new components: (i) migration; (ii) gradient descent; and (iii) mutation. In the original IWO, each individual plant used its own features to randomly distribute new seeds over the search space. There was no sharing of features among individuals. In IWO/BBO, on the contrary, a migration operator from biogeography-based optimization (BBO) provided the hybrid with a feature-sharing capability. This modification improved the quality of the distributed seeds.

Gradient descent improved the local search ability of IWO, while mutation increased the population diversity.

Biogeography-based optimization (BBO) was originally suggested by Simon [23]. Biogeography investigated the distribution of organisms in an ecosystem and the relations between them. Migration and mutation were the two main operators. The migration operator provided exploitation ability, while the exploration of the search space was based on the mutation operator. As in other biology-based algorithms, BBO had features to share information among solutions and to maintain good solutions from one iteration to the next. Chen *et al.* [24] embedded the covariance matrix based migration (CMM) into BBO to relieve BBO dependence upon the coordinate system. In CMM-BBO, the original coordinate system was rotated into an eigenvector-based coordinate system, in which solutions shared their information more efficiently. Gong *et al.* [25] developed a real-coded biogeography-based optimization (RCBBO) to extend the original BBO, so that each individual was directly encoded as a floating-point entity for solving global continuous optimization problems. Three mutation operators, Gaussian mutation, Cauchy mutation, and Lévy mutation, were also integrated into BBO to enhance its exploration ability and improve population diversity.

Yang's Bat Algorithm (BA) [26] was inspired by the social behavior of bats and their ability to use echolocation to sense distance. Liu *et al.* [27] incorporated the chaos strategy and Extremal Optimization (EO) algorithm into the BA to enhance the local search capability and the ability to escape from local optima. Yildizdan and Baykan [28] combined the Differential Evolution (DE) algorithm with the modified BA to balance the exploration and exploitation capability. The resulting hybrid algorithm showed superior performance to the standard BA in all tests. Ramli *et al.* [29] added adaptive dimension and inertia weight modifications to BA. The search performance and convergence speed of the modified BA significantly improved.

The Firefly Algorithm (FA) [30] was influenced by the flashing behavior of fireflies to attract one another. Rizk-Allah *et al.* [31] combined the FA with an ant colony optimization (ACO) algorithm. This hybrid was initialized using the ACO algorithm. Then the firefly algorithm was put to work as a local search to refine the solutions found by the ants. Ritthipakdee *et al.* [32] presented Firefly Mating Algorithm (FMA) for solving continuous optimization problems, in which GA was used as the core, incorporating a new mating pair selection method, which was inspired by the natural mating behavior of fireflies. The new mating pair selection method helped to improve convergence.

Several other interesting metaheuristic algorithms presented recently are reviewed next. Khalilpourazari and Khalilpourazary [33] used a Multi-Objective Dragonfly Algorithm (MODA) to simultaneously optimize final surface roughness, time and cost of grinding. The quality of the non-dominated Pareto optimal solutions, obtained from MODA, was significantly better than existing approaches. Pasandideh and Khalilpourazari [34] devised a Sine Cosine Crow Search Algorithm (SCCSA) based on two metaheuristics, Crow Search Algorithm and Sine Cosine Algorithm. SCCSA combined the concepts and operators of the two algorithms to ensure that all search agents followed other solutions and no low-quality random solution was generated. Significant improvements over the individual algorithms were shown. Khalilpourazari and Khalilpourazary [35] presented Sine Cosine Whale Optimization Algorithm (SCWOA), a hybrid of Sine Cosine Algorithm and Whale Optimization Algorithm, to optimize the parameters of a multi-pass milling process. SCWOA outperformed several previous works. Khalilpourazari and Khalilpourazary [36] devised a hybrid of Water Cycle Algorithm (WCA) and Moth-Flame Optimization (MFO) for solving numerical and constrained engineering optimization problems. In this hybrid, the spiral movement of the MFO was introduced into WCA to enhance its exploitation ability. Moreover, the streams in WCA were allowed to update their positions using a random walk, which helped improve exploration. Khalilpourazari and Pasandideh [37] used an exact method, called Sequential Quadratic Programming (SQP), and two hybrid metaheuristic algorithms, named Sine Cosine Crow Search Algorithm (SCCSA) and Water Cycle Moth-Flame Optimization (WCMFO) algorithm, to solve multi-item multi-constrained economic order quantity model. SQP solved small size problems efficiently; however, WCMFO was the best method for medium and large problems. Tansui and Thammano [38] developed Hydrozoan Algorithm (HA) based on the life cycle of hydrozoans, especially regeneration and transplantation, which included the formation of new animals, that are genetically different from their parents, as well as new ones, that are genetically identical to their parents. HA was specifically designed to find the best solutions for continuous optimization problems. It was benchmarked against GA and PSO on 20 standard benchmark functions. Even though HA was more successful than GA and PSO, HA did not find the global optima for most of the benchmark functions. Tansui and Thammano [39] presented Sea Turtle Foraging Algorithm (STFA), which imitated the foraging behavior of sea turtles moving toward a food source using an odor trail of dimethyl sulfide (DMS), which faded with distance and time. Further, turtle movement also depended on the direction and speed of the ocean current. The performance test on five unimodal functions revealed very good results, achieving optimal solutions in 4 out of 5 unimodal functions. However, STFA suffered from local optima stagnation due to its poor exploration capability.

To summarize, many individual metaheuristic algorithms have some weaknesses, for example, low robustness, premature convergence and little or no use of prior knowledge. Hybrids of metaheuristic algorithms have achieved better performance in finding optimal or near-optimal solutions than individual algorithms.

The algorithm described here is a hybrid of HA and STFA. This algorithm combined the advantages of both algorithms, viz. the good exploitation capability of STFA and the good exploration capability of HA.

The rest of this article is organized as follows. Section II describes the theoretical background. Section III explains our algorithm in detail. Section IV discusses the experimental results. Section V concludes with our contributions and indicates several directions for future research.

II. THEORETICAL BACKGROUND

A. HYDROZOAN ALGORITHM

The hydrozoan algorithm (HA) was inspired by reproduction in the hydrozoan life cycle. Hydrozoans are very small predators; most live in saltwater, but some species are found in freshwater. The hydrozoan has a very complex life cycle, which includes both sexual and asexual reproduction. A typical hydrozoan life cycle has three stages: the motile planula larva, the polyp, and the pelagic medusa [40]–[42]. Planula larvae are developed from sexual reproduction between medusae. The planula larvae then navigate through the water until they reach a hard surface and grow into polyps. Polyps reproduce asexually by developing small, genetically identical buds that protrude from the parent polyps. When the buds mature, they bud off to become independent medusae. In budding, the onset of budding and the bud development are controlled by two morphogens, called an activator and an inhibitor. These morphogens interact with each cell and control the increment and decrement of a growth parameter [43]. Once the medusae are released from the polyps, they gradually mutate. Some are stronger than others. Then mating pairs form and reproduce sexually.

HA imitates the nature of these three main stages of hydrozoan life cycle to make the search more effective. The operators used in HA are:

- one type of mutation operator is used to imitate the planula larvae swimming through the water,
- clonal selection is used to imitate asexual reproduction of the polyps and
- another type of mutation operator and the crossover operator are used to mimic medusa behavior.

The pseudocode of the HA is shown in Algorithm 1 and described below:

Step 1: Randomly generate the positions of N hydrozoans.

$$H_i(0) = \left[h_i^1, h_i^2, \ldots, h_i^D\right] \tag{1}$$

where i = 1 to N.

Step 2: Evaluate the fitness of each hydrozoan at its current position.

Step 3: The growth factor of each hydrozoan, $G_i(t)$, is determined from the ratio of the activator value (Ac_i) to the inhibitor value (In_i) as shown in (2)–(4).

$$G_i(t) = \frac{Ac_i}{In_i} \tag{2}$$

$$Ac_i = [f(H_i)]^{\beta} \tag{3}$$

$$In_i = [f(H_i)]^{\alpha} \tag{4}$$

where β is a significance factor of the activator and α is a significance factor of the inhibitor.

Step 4: The median of a set of growth factors $\{G_1(t), G_2(t), \ldots, G_N(t)\}$ is calculated.

Step 5: Calculate the number of buds to be dissected for each hydrozoan i by using (5) and (6).

$$DM_i = G_i(t) - Median \tag{5}$$

$$Bud_i = \begin{cases} 0; & if\ DM_i < 0 \\ 1; & if\ DM_i = \min^+ \\ 3; & if\ DM_i = \max^+ \\ 2; & otherwise \end{cases} \tag{6}$$

where $\min^+$ is the smallest positive DM_i and $\max^+$ is the largest positive DM_i.

Step 6: Clone each hydrozoan according to its Bud_i value.

Step 7: Mutate all newly created medusae by increasing or decreasing values of the randomly selected genes of the medusa by a small fraction.

Step 8: Evaluate the fitness of all mutated medusae.

Step 9: Select pairs of parents from the pool of medusae by using a roulette wheel selection method. In the roulette wheel selection, the probability of the individual i to be selected is defined:

$$P_i = \frac{f(M_i)}{\sum_{q=1}^{Q} f(M_q)} \tag{7}$$

where $f(M_i)$ is the fitness value of the individual i at time t.

Step 10: Perform multi-point crossover to produce new offspring.

Step 11: Mutate the newly created offspring by randomly changing the value of one or more genes in the offspring.

Step 12: Evaluate the fitness of all offspring. Then the group of offspring and the group of hydrozoans are merged. The best N individuals are included in the next-generation population.

Step 13: Repeat steps 3 – 12 until any stopping criteria is met.

B. SEA TURTLE FORAGING ALGORITHM (STFA)

The power of nature-inspired algorithms comes from the fact that they imitate the best characteristics of living individuals. Some nature-inspired algorithms were based on the swarming behavior of social animals [44], [45], for example, ants, bees, and birds, whereas some are inspired by solitary animals. STFA belongs to the latter group. In nature, sea turtles are not swarming animals; they independently forage for food. Their foraging behavior is very interesting and effective. Sea turtles are long lived and skilled ocean navigators, forever migrating throughout their lives. Their unique migrating and foraging behaviors inspired us to construct an algorithm that imitates their behaviors. Normally, sea turtles move in a straight path from one point to the next [46]. However, as they travel in the open sea, ocean currents affect their movements. During their long migration, sea turtles feed on small animals, sea grasses, algae and phytoplankton. In nature, sea grasses, algae and phytoplankton release a substance, that disintegrates into strong smelling dimethyl sulfide (DMS), to regulate the climate over the ocean to support their survival. Since DMS is volatile and capable of crossing the sea-air boundary, it accumulates in the air above areas that are abundant in sea grasses, algae and phytoplankton. Sea turtles can detect DMS and find areas with high concentrations of prey [47], [48]. Sea turtles move toward the food source, that releases the strongest odor. Their movement may be active and direct, but it may also be passive, assisted by the ocean current. The pseudocode of the STFA algorithm is in Algorithm 2 and described briefly below:

Step 1: Randomly initialize the positions of N sea turtles in a D-dimensional search space.

$$T_i(0) = \left[t_i^1, t_i^2, \ldots, t_i^D\right] \tag{8}$$

where i = 1 to N.

Step 2: Randomly initialize the velocity of each turtle, $V_i(0) = \left[v_i^1, v_i^2, \ldots, v_i^D\right]$. The velocity in each dimension of turtle i, v_i^d, is constrained to be within $\left[v\_\min^d, v\_\max^d\right]$:

$$v\_\max^d = \lambda\left[XUB^d - XLB^d\right] \tag{9}$$

$$v\_\min^d = -v\_\max^d \tag{10}$$

where XUB^d and XLB^d are the upper and lower bounds of the dth dimension of the search space and λ is a real number in [0, 1].

Step 3: Randomly generate the positions of M food sources.

$$K_j = \left[k_j^1, k_j^2, \ldots, k_j^D\right] \tag{11}$$

where j = 1 to M. Then the fitness values of all food sources are determined.

Step 4: Evaluate the fitness of all turtles. Then the strongest turtle is determined from:

$$I = \arg\max_i \left[f(T_i(t))\right] \tag{12}$$

where $f(T_i(t))$ is the fitness of turtle i at time t.

Step 5: Calculate the ocean current velocity at the position of each turtle, $VC_i = \left[vc_i^1, vc_i^2, \ldots, vc_i^D\right]$:

$$VC_i(t) = \gamma\left[T_I(t) - T_i(t)\right] \tag{13}$$

Step 6: Update the velocity of each sea turtle:

$$V_i(t+1) = V_i(t) + VC_i(t) + \left[\frac{f(T_i(t)) - f(T_i(t-1))}{f(T_i(t-1))}\right]\left[T_i(t) - T_i(t-1)\right] \tag{14}$$

where $T_i(t)$ is the position of turtle i at time t and $f(T_i(t))$ is the fitness of the turtle i at time t.

Step 7: Calculate the strength of the DMS odor from the food source j that is sensed by the turtle i, $C_{ij}(t)$, by comparing the fitness of the turtle with the fitness of the food source. If the turtle fitness is higher than that of the food source, then the strength of odor from that food source is taken as zero. On the other hand, if the turtle fitness is lower than that of the food source, then the strength of odor from that food source is determined by:

$$C_{ij}(t) = \frac{f(K_j)}{\sum_{q=1}^{M} f(K_q)} e^{-\left[\frac{d_{ij}^2}{2\sigma^2(t)}\right]} \tag{15}$$

where $f(K_j)$ is the fitness of the food source j. d_{ij} is the distance between turtle i and food source j. $\sigma(t)$ controls how far the DMS odor spreads; it decreases exponentially with time:

$$\sigma(t) = \sigma_0 e^{-\left[\frac{t}{T}\right]} \tag{16}$$

Step 8: Identify the best food source for turtle i. The best food source is the one that has the highest value of $C_{ij}(t)$ among all food sources.

$$J = \arg\max\left[C_{ij}\right] \tag{17}$$

Step 9: Update the position of each turtle.

$$T_i(t+1) = T_i(t) + \eta V_i(t+1) + C_{iJ}(t)\left[K_J - T_i(t)\right] \tag{18}$$

Step 10: Check the stopping criteria. If any one of them is met, the algorithm will be terminated. If not, two conditions are checked: i) If the value of t/T is an integer, go back to step 3; ii) If the value of t/T is not an integer, go back to step 4.

III. HA-STFA

The HA–STFA algorithm combines the advantages of the hydrozoan and the sea turtle foraging algorithms—the sea turtle foraging algorithm is embedded in the hydrozoan algorithm. The main idea is to provide a better balance between exploration (the advantage of HA) and exploitation (the advantage of STFA) capabilities. HA is based on clonal selection, crossover and mutation—three operators which are known to be efficient for exploring the problem space. STFA, which is based on local search, is embedded in HA to exploit the promising areas found by HA. In addition, two new features are introduced in this new hybrid algorithm: i) an adaptive two-point crossover operator and ii) an adaptive mutation probability, which is directly proportional to the rank of each individual in the population.

The detailed steps for the HA-STFA are set out in Algorithm 3 and described below:

Step 1: An initial population of N hydrozoans is randomly generated.

Step 2: Evaluate the fitness of each hydrozoan, $f(H_i)$.

Step 3: Calculate the growth factor for the hydrozoan i, $G_i(t)$, by using (2).

Step 4: Determine the median of a set of growth factors $\{G_1(t), G_2(t), \ldots, G_N(t)\}$. In the calculation of the median, the growth factors must first be ranked (sorted in ascending order) then the median is the one in the middle. More accurately, when a set of growth factors has an odd number of

Algorithm 1 Pseudocode of the Hydrozoan Algorithm

```
1: Objective function: min or max f(x);
2: Randomly create an initial population of N hydrozoans;
3: t = 1;
4: while (t < MaxGeneration)
5:     Evaluate the fitness values of all N hydrozoans;
6:     Calculate the growth factors of all hydrozoans using (2);
7:     Calculate the median of the growth factors of all hydrozoans;
8:     Calculate the number of buds to be dissected for each hydrozoan using (6);
9:     Create medusae by cloning each hydrozoan;
10:    Mutate all newly created medusae and evaluate their fitness;
11:    Select pairs of parents from the medusae pool by using the roulette wheel selection;
12:    Crossover the selected pairs of parents with an multi-point crossover operator;
13:    Evaluate the fitness of all offspring;
14:    Combine the group of offspring with the group of hydrozoans;
15:    Sort the combined list according to their fitness values;
16:    Select the top N individuals;
17:    t = t + 1;
18: end while
19: Output the best individual;
```

Algorithm 2 Pseudocode for the Sea Turtle Foraging Algorithm

```
1: Objective function: min or max f(x);
2: Randomly initialize the positions and velocities of N sea turtles;
3: Randomly create M food sources and evaluate their fitness;
4: t = 1;
5: while (t < MaxGeneration)
6:     Evaluate the fitness of all N turtles;
7:     Determine the strongest turtles in the population;
8:     for i = 1: N
9:         Calculate the velocity of the ocean current at the current position of the turtle i using (13);
10:        Update the velocity of the sea turtle i using (14);
11:        Calculate the DMS odor strength from each food source sensed by the turtle i using (15);
12:        Determine the food source with the highest DMS odor strength;
13:        Update the position of the turtle i using (18);
14:    end for
15:    t = t + 1;
16:    if (t mod T = 0)
17:        Randomly create M food sources and evaluate their fitness;
18:    end if
19: end while
20: Output the position of the strongest turtle;
```

members, the median is the value of the middle member; however, when a set has an even number of members, the median is the average of the values of two middle members.

Step 5: For each hydrozoan i, the number of buds to be dissected, $\mathrm{Bud_i}$, is determined by using (5) and (6).

Step 6: Each hydrozoan is cloned $\mathrm{Bud_i}$ times to produce a population of medusae. Then each newly created medusa $M_i = \left[m_i^1, m_i^2, \ldots, m_i^D\right]$ is mutated with a probability Pm1, which is defined by the user. The mutation is done by adding small random values to the values of the randomly selected genes of the medusa.

Step 7: Evaluate the fitness of each medusa. The strongest one is considered the global best.

Step 8: Calculate the velocity of the ocean current at the position of each medusa, $VC_i = \left[vc_i^1, vc_i^2, \ldots, vc_i^D\right]$, by using (13).

Step 9: Calculate DMS odor strength from the food source j that is perceived by medusa i, $\mathrm{C_{ij}(t)}$. This perceived odor strength and the ocean current velocity, calculated in the last step, influence the movement of the medusa to its next position. In finding $\mathrm{C_{ij}(t)}$, the medusa fitness is compared with the fitness of the food source. If the medusa fitness is higher than that of the food source, then the odor strength from that food source is set to zero. On the other hand, if the medusa fitness is lower than that of the food source, then the odor strength from that food source is:

$$C_{ij}(t) = \frac{f(K_j)}{\sum_{q=1}^{M} f(K_q)} e^{-\left[\frac{d_{ij}^2}{2\sigma^2(t)}\right]} \tag{19}$$

where $f(K_j)$ is the fitness of food source j, d_{ij} is the distance between medusa i and the food source j, $\sigma(t)$ controls the spread of the DMS; it decreases with time (i.e. the odor slowly fades away) following (16).

Step 10: For medusa i, identify the food source with the highest $C_{ij}(t)$ among all food sources, using (17).

Step 11: Update the position of the medusa:

$$M_i(t+1) = M_i(t) + \eta VC_i(t) + C_{iJ}(t)\left[K_J - M_i(t)\right] \tag{20}$$

where $C_{iJ}(t)$ is the odor strength of food source J.

Step 12: Select pairs of parents from the medusa pool using the roulette wheel selection.

Step 13: Each selected pair is mated to generate an offspring. The crossover operator used here yields only one offspring that inherits more genes from the higher fitness parent than from the lower fitness parent. The number of genes that each parent contributes to the offspring is determined according to (21) and (22).

$$n_h = \left[\frac{a-b}{a}\right]\left[\frac{D}{2}\right] + \left[\frac{D}{2}\right] \tag{21}$$

$$n_l = D - n_h \tag{22}$$

where n_h and n_l are the number of genes that the higher fitness parent and the lower fitness parent contribute to the offspring respectively. a is the fitness of the stronger parent and b is the fitness of the weaker parent. D is the dimension of the search space.

Afterward, the offspring is created by combining the first n_h genes of the stronger parent with the last n_l genes of the weaker parent.

Step 14: The generated offspring are mutated as follows: i) sort the offspring according to fitness; ii) calculate the mutation probability for each gene of offspring i, $\mathrm{Pm2_i}$, using (23); iii) generate a random number in the range of [0, 1] for each gene of the offspring i and compare it with $\mathrm{Pm2_i}$; iv) if the random number is less than or equal to $\mathrm{Pm2_i}$, mutate the gene; otherwise, keep the gene as it is.

$$Pm2_i = \frac{r_i}{N} \tag{23}$$

where N is the population size; $\mathrm{r_i}$ is the rank of offspring i; the best offspring is assigned a rank of 1 while the worst offspring is assigned a rank of N.

Step 15: The combined list of offspring and the hydrozoans are sorted by fitness. Then the top N individuals are selected for the next generation.

Step 16: Steps 3 – 15 are repeated until any stopping criteria is met.

Note that when solving the constrained optimization problems, the updated position of the medusa in steps 6 and 11 is checked to ensure it is still within the boundary. If the element in any dimension of the new position is out of bounds, it is replaced with a random number between the lower and upper boundaries of that dimension.

IV. RESULTS AND DISCUSSION

Twenty-one standard benchmark functions shown in Table 1 were used to show the performance of HA-STFA. These functions were in two categories: unimodal and multimodal. The first category included five simple unimodal functions: Sphere (*f2*), Zakharov (*f*6), Dixon and Price (*f8*), Rotated Hyper-Ellipsoid (*f18*), and Matyas (*f19*). The dimensions of these five functions was 30. The complexity of these unimodal functions at this high dimension were comparable to that of a multimodal function. The second category included sixteen complex high-dimensional multimodal functions, that had two or more local optima: Ackley (*f1*), Griewank (*f3*), Rastrigin (*f4*), Rosenbrock (*f5*), Michalewicz (*f7*), Levy (*f9*), Cross-in-Tray (*f10*), Drop-Wave (*f11*), Eggholder (*f12*), Holder table (*f13*), Schaffer function N. 2 (*f14*), Shubert (*f15*), Schaffer function N. 4 (*f16*), Beale (*f17*), Styblinski –Tang (*f20*), and De Jong function N. 5 (*f21*). Table 1 shows the test function details.

Our hybrid HA-STFA was compared to two sets of algorithms:

i) HA and STFA – This comparison determined whether the hybrid HA-STFA performed better than individual HA and STFA.

ii) Twelve state-of-the-art algorithms: Flower Pollination Algorithm (FPA), Modified Flower Pollination Algorithm (MFPA), Bat algorithm (BAT), Firefly algorithm (FF), Genetic Algorithm (GA), Simulated Annealing (SA), Bird Swarm Algorithm (BSA), Chicken Swarm Optimization (CSO), Grey Wolf Optimizer (GWO), Novel Bat Algorithm (NBA), Moth-Flame Optimization (MFO), Water Cycle Algorithm (WCA). The test results of FPA, MFPA, BAT, FF, GA and SA were obtained from [18], whereas for BSA, CSO, GWO, NBA, MFO and WCA, we used codes from Mathworks File Exchange [49]–[54].

In addition, five real-life engineering problems were used to test the efficiency of HA-STFA in solving complex problems. These problems are highly nonlinear with dimensions from 2 to 20. The formulation of these problems are:

i) R1: Optimal capacity of gas production facilities [55]

$$\begin{aligned} Min:\ f(x) = {} & 61.8 + 5.72x_1 + 0.2623[(40-x_1)ln(\frac{x_2}{200})]^{-0.85} \\ & + 0.087(40-x_1)ln(\frac{x_2}{200}) + 700.23x_2^{-0.75} \end{aligned}$$

$$subject\ to\ x_1 \geq 17.5, \quad x_2 \geq 200$$

$$Bounds:\ 17.5 \leq x_1 \leq 40, \quad 300 \leq x_2 \leq 600$$

The best-known optimal value is $f_{\min} = 169.844$.

TABLE 1. Standard benchmark functions used for validation.

No.	Function name	Equation
f1	Ackley's function	$f_1(x) = -20exp\left[-\frac{1}{5}\sqrt{(\frac{1}{d}\sum_{i=1}^{d} x_i^2)}\right] - exp\sum_{i=1}^{d} cos(2\pi x_i) + 20 + e, -15 \le x_i \le 30$ Global minimum = 0 at x^* = (0, 0, …, 0). This function is multimodal.
f2	Sphere function	$f_2(x) = \sum_{i=1}^{d} x_i^2, -5.12 \le x_i \le 5.12$ Global minimum = 0 at x^* = (0, 0, …, 0). This function is unimodal.
f3	Griewank's function	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1, -600 \le x_i \le 600$ Global minimum = 0 at x^* = (0, 0, …, 0). This function is multimodal.
f4	Rastrigin's function	$f_4(x) = 10n + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)], -5.12 \le x_i \le 5.12$ Global minimum = 0 at x^* = (0, 0, …, 0). This function is multimodal.
f5	Rosenbrock's function	$f_5(x) = \sum_{i=1}^{d} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, -5 \le x_i \le 5$ Global minimum = 0 at x^* = (1, 1, …, 1). This function is multimodal.
f6	Zakharov's function	$f_6(x) = \sum_{i=1}^{d} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^4, -5 \le x_i \le 10$ Global minimum = 0 at x^* = (0, 0, …, 0). This function is unimodal.
f7	Michalewicz's function	$f_7(x) = -\sum_{i=1}^{d} sin(x_i)sin^{2m}\left(\frac{ix_i^2}{\pi}\right), m = 10, 0 \le x_i \le \pi$ Global minimum = –1.8013 at x^* = (0, 0, …, 0). This function is multimodal.
f8	Dixon and Price's function	$f_8(x) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$ Global minimum = 0 at $x^* = 2^{-\frac{2^i - 2}{2^i}}$, i = 1, 2, …, d. This function is unimodal.
f9	Levy's function	$f_9(x) = sin^2(\pi y_i) + \sum_{i=1}^{d-1}(y_i - 1)^2[1 + 10sin^2(\pi y_i + 1)] + (y_i - 1)^2[1 + sin^2(2\pi y_i)],$ $y_i = 1 + \frac{x_i - 1}{4}, i = 1, 2, ..., d, -10 \le x_i \le 10$ Global minimum = 0 at x^* = (1, …, 1). This function is multimodal.
f10	Cross-in-Tray function	$f_{10}(x) = -0.0001(\left\|sin(x_1)sin(x_2)exp(\left\|100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right\|)\right\| + 1)^{0.1}, -10 \le x_i \le 10$ Global minimum = –2.06261 at x^* = (1.3491, –1.3491), (1.3491, 1.3491), (–1.3491, 1.3491) and (–1.3491, –1.3491). This function is multimodal.
f11	Drop-Wave function	$f_{11}(x) = -\frac{1 + cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}, -5.12 \le x_i \le 5.12$ Global minimum = –1 at x^* = (0, 0). This function is multimodal.
f12	Eggholder function	$f_{12}(x) = -(x_2 + 47)\sin\left(\sqrt{\left\|x_2 + \frac{x_1}{2} + 47\right\|}\right) - x_1 \sin\left(\sqrt{\left\|x_1 - (x_2 + 47)\right\|}\right), -5.12 \le x_i \le 5.12$ Global minimum = –959.6407 at x^* = (512, 404.2319). This function is multimodal.

TABLE 1. ***(Continued.)*** **Standard benchmark functions used for validation.**

f13	Holder table function	$f_{13}(x) = -\left\lvert sin(x_1)\cos(x_2)exp\left(\left\lvert 1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert, -10 \le x_i \le 10$ Global minimum = −19.2085 at x^* = (8.05502, 9.66459), (8.05502, −9.66459), (−8.05502, 9.66459) and (−8.05502, −9.66459). This function is multimodal.
f14	Schaffer function N. 2	$f_{14}(x) = 0.5 + \frac{\sin^2\left(x_1^2 - x_2^2\right) - 0.5}{\left[1+0.001(x_1^2+x_2^2)\right]^2}, -100 \le x_i \le 100$ Global minimum = 0 at x^* = (0, 0). This function is multimodal.
f15	Shubert function	$f_{15}(x) = (\sum_{i=1}^{5} icos((i+1)x_1+i))(\sum_{i=1}^{5} icos((i+1)x_2+i)), -5.12 \le x_i \le 5.12$ Global minimum = −186.7309. This function is multimodal.
f16	Schaffer function N. 4	$f_{16}(x) = 0.5 + \frac{\cos\left(\sin\left(\left\lvert x_1^2 - x_2^2\right\rvert\right)\right) - 0.5}{\left[1+0.001(x_1^2+x_2^2)\right]^2}, -100 \le x_i \le 100$ Global minimum = 0.292579 at x^* = (0, 1.25313). This function is multimodal.
f17	Beale function	$f_{17}(x) = \left(1.5 - x_1 + x_1x_2\right)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2, -4.5 \le x_i \le 4.5$ Global minimum = 0 at x^* = (3, 0.5). This function is multimodal.
f18	Rotated Hyper-Ellipsoid function	$f_{18}(x) = \sum_{i=1}^{d}\sum_{j=1}^{i} x_j^2, -65.536 \le x_i \le 65.536$ Global minimum = 0 at x^* = (0, 0, …, 0). This function is unimodal.
f19	Matyas function	$f_{19}(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2, 10 \le x_i \le 10$ Global minimum = 0 at x^* = (0, 0). This function is unimodal.
f20	Styblinski–Tang function	$f_{20}(x) = \frac{1}{2}\sum_{i=1}^{d}\left(x_i^4 - 16x_i^2 + 5x_i\right), -5 \le x_i \le 5, i = 1,2,...,d$ Global minimum = −39.16599*d* at x^* = (−2.903534, …, −2.903534). This function is multimodal.
f21	De Jong function N. 5	$f_{21}(x) = \left(0.002 + \sum_{i=1}^{25}\frac{1}{i+\left(x_1-a_{1i}\right)^6+\left(x_2-a_{2i}\right)^6}\right)^{-1}$ $a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & ... & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & ... & 32 & 32 & 32 \end{pmatrix}$ Global minimum = 0.99804 at x^* = (-32, -32). This function is multimodal.

ii) R2: Gas transmission compressor design [55]

$$\begin{aligned} Min : f(x) = {} & 8.61 \times 10^5 \times x_1^{(1/2)} x_2 x_3^{(-2/3)} (x_2^2 - 1)^{(-1/2)} \\ & + 3.69 \times 10^4 \times x_3 + 7.72 \times 10^8 \times x_1^{-1} x_2^{0.219} \\ & - 765.43 \times 10^6 \times x_1^{-1} \end{aligned}$$

$$Bounds : 10 \le x_1 \le 55, \quad 1.1 \le x_2 \le 2, \quad 10 \le x_3 \le 40$$

The best-known optimal value is $f_{\min} = 2.96438E + 06$.

iii) R3: Optimal thermohydraulic performance of an artificial roughened air heater [55]

$$Max : L = 2.51 ln e^+ + 5.5 - 0.1R_M - G_H$$

where

$$R_M = 0.95x_2^{0.53}, \quad G_H = 4.5(e^+)^{0.28}(0.7)^{0.57},$$

$$e^+ = x_1x_3(\frac{\bar{f}}{2})^{(1/2)}, \quad \bar{f} = \frac{f_s + f_r}{2},$$

$$f_s = 0.079x_3^{-0.25},$$

$$f_r = 2(0.95x_3^{0.53} + 2.5ln(1/2x_1)^2 - 3.75)^{-2}$$

$$Bounds : 0.02 \le x_1 \le 0.8, \quad 10 \le x_2 \le 40,$$

$$3000 \le x_3 \le 20000$$

The best-known optimal value is $f_{\max} = 4.21421$.

iv) R4: Spread spectrum radar poly-phase code design problem [55]

$$Min : f(x) = \max\{f_1(X), f_2(X), \ldots, f_{2m}(X)\}$$

where

$$X = (x_1, x_2, \ldots, x_n) \in R^n | 0 \le x_j \le 2\pi,$$

$$j = 1, 2, \ldots, n \quad and\ m = 2n - 1,$$

$$f_{2i-1}(x) = \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right), \quad i = 1, 2, \ldots, n$$

Algorithm 3 Pseudocode of the HA–STFA Algorithm

```
1: Objective function: min or max f(x);
2: Randomly create an initial population of N hydrozoans;
3: t = 1;
4: while (t < MaxGeneration)
5:     Evaluate the fitness of all N hydrozoans;
6:     Calculate the growth factor for each hydrozoan and determine the median of those growth factors;
7:     Determine the number of buds of each hydrozoan using (6);
8:     Create medusae by cloning each hydrozoan at the same number of buds of that hydrozoan;
9:     Mutate all newly created medusae;
10:    Evaluate the fitness of each newly created medusa and find the one with the highest fitness value;
11:    for each newly created medusa
12:        Calculate the velocity of the ocean current at the position of the medusa using (13);
13:        Calculate the DMS odor strength from each food source sensed by each medusa using (19);
14:        Determine the food source with the highest odor strength;
15:        Update position of the medusa using (20);
16:    end for
17:    Select pairs of parents from the pool of medusae by using the roulette wheel selection;
18:    Crossover the selected parent pairs;
19:    Mutate the offspring;
20:    Evaluate the fitness of all offspring;
21:    Combine the group of offspring with the group of hydrozoans;
22:    Sort the combined list according to fitness;
23:    Select the top N individuals;
24:    t = t + 1;
25: end while
26: Output the best individual;
```

$$f_{2i}(x) = 0.5 + \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j|+1}^{j} x_k \right), \quad i = 1, 2, \ldots, n-1$$

$$f_{m+i}(x) = -f_i(x), \quad i = 1, 2, \ldots, m$$

The best-known optimal value is $f_{\min} = 0$.

v) R5: Chemical equilibrium application [56]

$$\begin{aligned} x_1x_2 + x_1 - 3x_5 &= 0, \\ 2x_1x_2 + x_1 + x_2x_3^2 + R_8x_2 - Rx_5 + 2R_{10}x_2^2 + R_7x_2x_3 & \\ + R_9x_2x_4 &= 0, \\ 2x_2x_3^2 + 2R_5x_3^2 - 8x_5 + R_6x_3 + R_7x_2x_3 &= 0, \\ R_9x_2x_4 + 2x_4^2 - 4Rx_5 &= 0, \\ x_1(x_2+1) + R_{10}x_2^2 + x_2x_3^2 + R_8x_2 + R_5x_3^2 + x_4^2 - 1 + R_6x_3 & \\ + R_7x_2x_3 + R_9x_2x_4 &= 0 \end{aligned}$$

where

$$R = 10, \quad R_5 = 0.193, \; R_6 = \frac{0.002597}{\sqrt{40}}, \; R_7 = \frac{0.003448}{\sqrt{40}},$$

$$R_8 = \frac{0.00001799}{40}, \quad R_9 = \frac{0.0002155}{\sqrt{40}},$$

$$R_{10} = \frac{0.00003846}{40}$$

The best-known optimal value is $f_{\min} = 0$.

The performance of HA-STFA was compared to the Invasive weed optimization (IWO), Differential evolution with probabilistic parent centric crossover (Pro. DEPCX), and Quadratic approximation invasive weed optimization (QAIWO). The results of these three algorithms were obtained from Naidu and Ojha [21] and Ali *et al.* [55].

A. RESULTS FOR STANDARD BENCHMARK FUNCTIONS

For the standard benchmark functions, each algorithm was run five times on each test function. Each run started with a different initial population and stopped when the maximum number of function evaluations (maxNFE) was reached. The parameters of HA-STFA, HA and STFA are given in Table 2.

Table 3 compares HA-STFA, HA and STFA. Table 4 compares HA-STFA with twelve state-of-the-art algorithms. These tables report minimum, maximum, mean and standard deviation from 5 runs. Performance was measured in terms of the ability to locate the global minimum. The best results (both lowest mean and standard deviation) obtained among all algorithms are shown in bold.

Table 3 shows that the hybrid HA-STFA performed better than HA and STFA individually. When any individual algorithms obtained the optimal solution, HA-STFA also obtained it. When any individual algorithms did not obtain the optimal solution, HA-STFA always obtained a better solution. Besides achieving better results, HA-STFA also converged faster than both HA and STFA on most functions. As an example, figures 1 – 5 compare the convergence curves of

TABLE 2. Parameters of HA-STFA, HA and STFA.

Parameters	Values		
	HA-STFA	HA	STFA
N	50	50	50
α	0.9	0.9	-
β	1.1	1.1	-
Pc	0.9, 1.0	0.9, 1.0	-
Pm	-	0.1, 0.2, 0.3, 0.4	-
Pm1	0.1, 0.2, 0.3, 0.4	-	-
M	3	-	3
T	3	-	3
λ	0.05	-	0.05
γ	0.5	-	0.5
σ_0	1	-	1
η	0.005	-	0.005
maxNFE	1,500,000	1,500,000	1,500,000

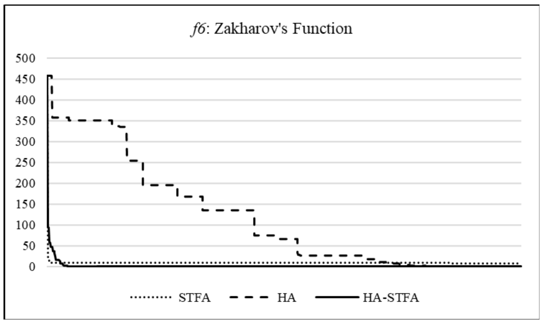

FIGURE 1. Convergence curves for *f6*.

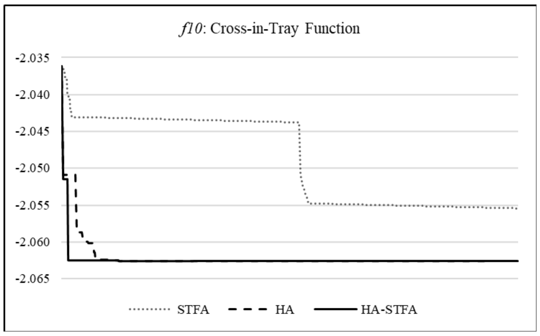

FIGURE 2. Convergence curves for *f10*.

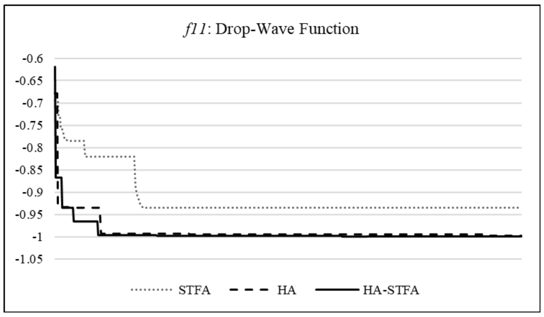

FIGURE 3. Convergence curves for *f11*.

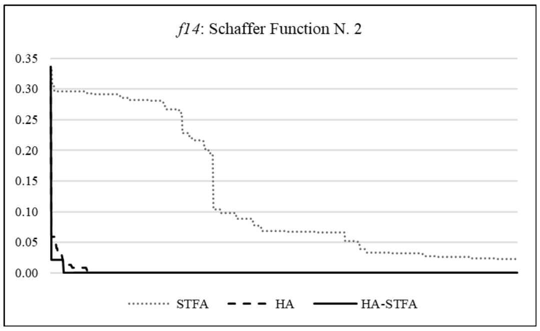

FIGURE 4. Convergence curves for *f14*.

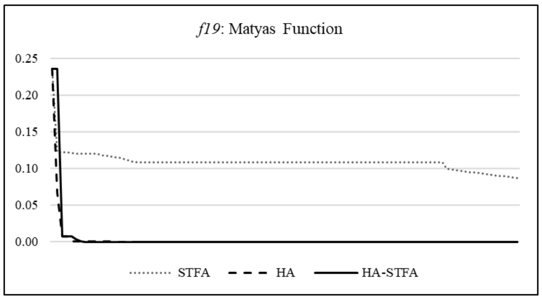

FIGURE 5. Convergence curves for *f19*.

STFA, HA and HA-STFA for *f6*, *f10*, *f11*, *f14* and *f19*. In these figures, all three algorithms obtained either optimal or near-optimal solutions. For functions *f10*, *f11* and *f14*, HA-STFA converged slightly faster than HA, and converged much faster than STFA. HA-STFA converged slightly slower than STFA for function *f6* and converged slightly slower than HA for function *f19*. However, it converged to better solutions than any of the other algorithms.

From the results in Table 4, the following paragraphs summarize the results of HA-STFA according to function categories – unimodal and multimodal.

For all but one of the 30-dimensional unimodal functions, the hybrid HA–STFA algorithm ranked fourth, after NBA, GWO and BSA. HA-STFA performance was very close to that of the top three algorithms. These unimodal functions were *f*2, *f*6, *f*18 and *f19*. For the remaining unimodal function, *f*8, HA-STFA came a very close second to MFPA.

For the 2-dimensional multimodal functions, the hybrid HA–STFA algorithm was able to find the global minimum on the following 6 out of 9 functions: *f7*, *f10*, *f11*, *f12*, *f13* and *f21*. HA-STFA found the global minimum of the above six functions on all 5 runs, regardless of initial population states. For *f14* and *f16*, HA-STFA came in second place. For *f17*, only MFPA was able to find the global minimum, whereas all the other algorithms obtained near-optimal solutions.

For the 5-dimensional multimodal function, *f15*, all algorithms except GA converged to the global minimum.

TABLE 3. Comparative results of HA-STFA, HA and STFA on standard benchmark functions.

Functions	Algorithms	Min	Max	Mean	SD
f1	STFA	2.58132	3.952205	3.49918	0.531568
	HA	1.80529E-05	2.98427E-05	2.37581E-05	4.90702E-06
	HA-STFA	**8.88178E-16**	**8.88178E-16**	**8.88178E-16**	**0**
f2	STFA	1.216160	1.960434	1.46E+00	0.303928
	HA	6.8696E-08	9.63429E-08	8.5703E-08	1.04615E-08
	HA-STFA	**3.96571E-103**	**6.21013E-97**	**1.33285E-97**	**4.39E-97**
f3	STFA	1.349633	6.9159190	4.559081	2.93457
	HA	1.00147E-09	1.46163E-09	1.3136E-09	1.97037E-10
	HA-STFA	**1.11022E-16**	**1.11022E-16**	**1.11022E-16**	**0**
f4	STFA	44.07935	60.84896	54.25363	7.7363842
	HA	3.99316E-06	5.30598E-06	4.68302E-06	4.99189E-07
	HA-STFA	**5.68434E-14**	**5.68434E-14**	**5.68434E-14**	**0**
f5	STFA	84.48532	587.8396	332.83149	231.1101
	HA	25.33098	26.39151	25.754278	0.413551
	HA-STFA	**6.69960E-09**	**1.02460E-07**	**5.36299E-08**	**1.59E-08**
f6	STFA	8.60E-01	1.19E+00	8.60E-01	3.30E-01
	HA	3.01163E-08	5.61857E-08	4.61914E-08	9.98885E-09
	HA-STFA	**9.94E-101**	**5.28E-109**	**7.32E-58**	**1.37E-90**
f7	STFA	-1.8013	-1.8013	-1.8013	1.0063E-10
	HA	-1.8013	-1.8013	-1.8013	7.41269E-07
	HA-STFA	**-1.8013**	**-1.8013**	**-1.8013**	**0**
f8	STFA	8.90E-01	4.23E+00	2.03E+00	1.48E+00
	HA	6.67E-01	6.67E-01	6.67E-01	5.21E-08
	HA-STFA	**3.39E-16**	**1.81E-15**	**1.91E-15**	**2.46E-15**
f9	STFA	1.711051	2.845357	2.28507	0.522168
	HA	1.47E-01	1.97E-01	1.77E-01	2.30E-02
	HA-STFA	**1.64E-04**	**8.95E-02**	**1.36E-02**	**4.87E-02**
f10	STFA	-2.06261	-2.06261	-2.06261	9.04719E-14
	HA	-2.06261	-2.06261	-2.06261	1.7196E-10
	HA-STFA	**-2.06261**	**-2.06261**	**-2.06261**	**0**
f11	STFA	-1	-1	-1	6.09556E-11
	HA	**-1**	**-1**	**-1**	**0**
	HA-STFA	**-1**	**-1**	**-1**	**0**
f12	STFA	-959.641	-959.641	-959.641	4.28115E-06
	HA	1.75804E-09	7.95971E-09	2.84617E-09	2.95539E-09
	HA-STFA	**-959.641**	**-959.641**	**-959.641**	**0**
f13	STFA	-19.2085	-19.2085	-19.2085	3.05178E-10
	HA	**-19.2085**	**-19.2085**	**-19.2085**	**0**
	HA-STFA	**-19.2085**	**-19.2085**	**-19.2085**	**0**
f14	STFA	1.33227E-15	4.44089E-15	2.13163E-15	1.30987E-15
	HA	2.03698E-11	6.83204E-11	4.21598E-11	2.03241E-11
	HA-STFA	**2.22E-16**	**2.22E-16**	**2.22E-16**	**0**
f15	STFA	-186.731	-186.731	-186.731	4.56145E-06
	HA	1.31797E-15	8.49484E-15	4.67101E-15	3.44741E-15
	HA-STFA	**-186.731**	**-186.731**	**-186.731**	**5.72E-08**
f16	STFA	0.500091	0.500091	0.500091	3.57149E-08
	HA	0.500091	0.500091	0.500091	3.67184E-08
	HA-STFA	**0.500091**	**0.500091**	**0.500091**	**0**
f17	STFA	**2.55017E-16**	**1.06342E-14**	**6.67711E-15**	**4.15987E-15**
	HA	3.96046E-09	7.6215E-09	5.26673E-09	1.63231E-09
	HA-STFA	1.54E-14	1.88E-13	5.64E-14	7.41E-14
f18	STFA	1.0624E-05	1.24144	0.605006	0.593407
	HA	2.53679E-07	3.81501E-07	3.32137E-07	5.08372E-08
	HA-STFA	**7.30E-137**	**6.40E-132**	**1.30E-132**	**2.90E-132**
f19	STFA	9.57681E-16	2.76325E-15	1.6929E-15	1.07139E-15
	HA	6.555E-11	2.27276E-10	1.4251E-10	6.74145E-11
	HA-STFA	**3.87E-308**	**9.33E-101**	**1.86E-101**	**4.17E-101**
f20	STFA	-78.3323	-78.3323	-78.3323	1.45928E-10
	HA	2.00222E-08	5.38167E-08	2.79472E-08	1.61828E-08
	HA-STFA	**-78.3323**	**-78.3323**	**-78.3323**	**0**
f21	STFA	0.040750	7.531460	4.299456	2.692422
	HA	0.998004	0.998004	0.998004	4.98205E-09
	HA-STFA	**0.998004**	**0.998004**	**0.998004**	**9.59E-15**

TABLE 4. Comparative results of HA-STFA with twelve state-of-the-art algorithms on standard benchmark functions.

Functions	Algorithms	Min	Max	Mean	SD
f1	FPA	3.64153E-14	7.94831E-12	2.07259E-12	2.47204E-12
	MFPA	**8.88178E-16**	**8.88178E-16**	**8.88178E-16**	**0**
	BAT	9.96441E-06	7.18095	2.36293	2.02684
	FF	9.27835E-06	0.000133534	8.17798E-05	3.15732E-05
	GA	2.22971E-07	5.44992E-06	1.24595E-06	1.2319E-06
	SA	1.06E-06	2.57994	0.086127	0.471006
	BSA	**8.88178E-16**	**8.88178E-16**	**8.88178E-16**	**0**
	CSO	8.88178E-16	4.44089E-15	1.59872E-15	1.58882E-15
	GWO	4.44089E-15	4.44089E-15	4.44089E-15	0
	NBA	8.88178E-16	4.44089E-15	3.73035E-15	1.58882E-15
	MFO	4.44089E-15	4.44089E-15	4.44089E-15	0
	WCA	4.44089E-15	4.44089E-15	4.44089E-15	0
	HA-STFA	**8.88178E-16**	**8.88178E-16**	**8.88178E-16**	**0**
f2	FPA	9.52325E-33	1.53645E-25	6.19243E-27	2.81558E-26
	MFPA	2.98961E-70	6.01114E-60	2.00588E-61	1.09744E-60
	BAT	7.14891E-13	3.59489E-10	5.21099E-11	7.47075E-11
	FF	1.10737E-12	1.60971E-10	4.72459E-11	3.72695E-11
	GA	7.33358E-16	7.51111E-13	1.53934E-13	1.6844E-13
	SA	7.06E-07	0.000201	2.65E-05	4.20E-05
	BSA	2.4282E-308	3.661E-304	7.3323E-305	0
	CSO	4.1498E-109	8.8685E-101	3.1144E-101	3.645E-101
	GWO	2.227E-308	2.6068E-308	2.4036E-308	0
	NBA	**2.2445E-308**	**2.4177E-308**	**2.2951E-308**	**0**
	MFO	1.14251E-93	7.52127E-91	1.52803E-91	2.997E-91
	WCA	3.92795E-37	1.71526E-36	1.24855E-36	6.353E-37
	HA-STFA	3.96571E-103	6.21013E-97	1.33285E-97	4.39E-97
f3	FPA	2.65431E-08	7.6365E-05	5.58634E-06	1.40067E-05
	MFPA	**0**	**0**	**0**	**0**
	BAT	1.8846E-12	0.976356	0.307127	0.305973
	FF	3.57752E-08	0.00739636	0.000619655	0.00188366
	GA	2.22045E-16	6.19282E-13	1.16063E-13	1.7023E-13
	SA	0.013878	0.446633	0.182588	0.134813
	BSA	1.11022E-16	6.32827E-15	1.44329E-15	2.73304E-15
	CSO	1.11022E-16	2.22045E-16	1.33227E-16	4.96507E-17
	GWO	1.11022E-16	2.22045E-16	1.33227E-16	4.96507E-17
	NBA	1.11022E-16	1.36557E-14	2.81997E-15	6.05738E-15
	MFO	0.00739604	0.024573294	0.015747599	0.006399021
	WCA	6.66134E-16	0.078703541	0.025091993	0.031028975
	HA-STFA	1.11022E-16	1.11022E-16	1.11022E-16	0
f4	FPA	0	3.19389E-12	1.6982E-13	5.89897E-13
	MFPA	**0**	**0**	**0**	**0**
	BAT	5.98135E-10	9.94956	2.02308	2.22454
	FF	3.6469E-10	3.30763E-08	9.21223E-09	7.45866E-09
	GA	4.01457E-13	0.994959	0.0663306	0.252429
	SA	3.82E-08	1.98992	0.657481	0.749738
	BSA	5.68434E-14	2.27374E-13	1.13687E-13	6.96187E-14
	CSO	5.68434E-14	5.68434E-14	5.68434E-14	0
	GWO	5.68434E-14	5.68434E-14	5.68434E-14	0
	NBA	31.83864463	151.2329161	83.17829109	47.65707806
	MFO	81.58641548	182.0763428	129.3865892	36.69628959
	WCA	38.803373	70.64196697	57.70751532	14.53794382
	HA-STFA	5.68434E-14	5.68434E-14	5.68434E-14	0

TABLE 4. ***(Continued.)*** **Comparative results of HA-STFA with twelve state-of-the-art algorithms on standard benchmark functions.**

f5	FPA	4.59735E-10	1.05915E-05	7.70467E-07	2.0289E-06
	MFPA	**0**	**9.66355E-30**	**3.30336E-31**	**1.76315E-30**
	BAT	5.72513E-08	3.54771	0.260185	0.809771
	FF	1.24204E-09	7.89547E-07	1.93476E-07	1.75331E-07
	GA	1.58342E-06	0.00356429	0.00240795	0.0007972
	SA	0.006559	7.42722	1.53167	2.3835
	BSA	28.91165882	28.97582324	28.95856492	0.026796485
	CSO	26.33294233	28.62872444	27.57499184	1.015615147
	GWO	24.21807255	27.11477835	25.8015223	1.435921599
	NBA	0.414435352	4.070944621	3.056660964	1.521388875
	MFO	16.44296772	67.35578576	37.09060267	27.62971644
	WCA	3.316136919	6.136901696	4.484072048	1.238870605
	HA-STFA	6.69960E-09	1.02460E-07	5.36299E-08	1.59E-08
f6	FPA	3.50262E-31	2.091E-25	1.52229E-26	4.79979E-26
	MFPA	9.98E-71	1.35E-42	4.49E-44	2.46E-43
	BAT	3.67E-12	3.10E-10	9.83E-11	7.86E-11
	FF	6.41E-13	4.87E-10	1.39E-10	1.35E-10
	GA	4.80E-16	2.33E-12	5.50E-13	7.62E-13
	SA	8.10E-09	0.001371	5.89E-05	0.000249
	BSA	**3.1153E-308**	**2.6379E-305**	**7.4013E-306**	**0**
	CSO	1.19789E-77	5.68989E-63	1.13853E-63	2.54429E-63
	GWO	4.9583E-234	1.9626E-228	3.9301E-229	0
	NBA	2.2376E-308	6.9439E-304	1.389E-304	0
	MFO	119.9725788	383.0563821	274.1935263	97.15193325
	WCA	8.92E-27	8.99E-24	2.34E-24	3.83451E-24
	HA-STFA	9.94E-101	5.28E-109	7.32E-58	1.37E-90
f7	FPA	-1.8013	-1.8013	-1.8013	9.03E-16
	MFPA	-1.8013	-1.8013	-1.8013	9.03E-16
	BAT	-1.98795	-1.68801	-1.81228	0.0519564
	FF	-1.8013	-1.8013	-1.8013	7.41E-11
	GA	-1.8013	-1.8013	-1.8013	1.57E-11
	SA	-1.80128	-1.39938	-1.78277	0.075241
	BSA	-1.8013	-1.8013	-1.8013	2.645E-06
	CSO	**-1.8013**	**-1.8013**	**-1.8013**	**0**
	GWO	-1.8013	-1.8013	-1.8013	9.997E-09
	NBA	**-1.8013**	**-1.8013**	**-1.8013**	**0**
	MFO	-1.8013	-1.8013	-1.8013	2.7733E-15
	WCA	-1.8013	-1.8013	-1.8013	1.5700E-16
	HA-STFA	**-1.8013**	**-1.8013**	**-1.8013**	**0**
f8	FPA	7.27E-15	2.67E-10	2.00E-11	6.00E-11
	MFPA	**3.70E-32**	**3.70E-32**	**3.70E-32**	**0**
	BAT	1.41E-11	1.51E-09	2.88E-10	2.85E-10
	FF	2.64E-11	2.61E-09	6.09E-10	5.43E-10
	GA	2.35E-14	2.10E-10	3.52E-11	5.50E-11
	SA	2.84E-08	0.001708	0.000181	0.000379
	BSA	0.666666	0.996789	0.928948	0.146626
	CSO	0.666666	0.666965	0.666726	0.000133
	GWO	0.66666667	0.666666675	0.666666673	1.7725E-09
	NBA	0.666666667	0.666666667	0.666666667	3.65439E-12
	MFO	0.666666667	0.666667014	0.666666736	1.55556E-07
	WCA	0.666666667	0.666666667	0.666666667	2.21697E-15
	HA-STFA	3.39E-16	1.81E-15	1.91E-15	2.46E-15

TABLE 4. ***(Continued.)*** **Comparative results of HA-STFA with twelve state-of-the-art algorithms on standard benchmark functions.**

f9	FPA	4.01E-30	1.24E-24	1.01E-25	2.71E-25
	MFPA	**1.50E-32**	**1.50E-32**	**1 .49976E-32**	**1.11E-47**
	BAT	1.51E-14	5.14E-11	1.34E-11	1.19E-11
	FF	2.25E-12	1.61E-10	5.04E-11	5.03E-11
	GA	4.54E-16	1.38E-12	2.18E-13	3.93E-13
	SA	1.10E-05	0.009845	0.001214	0.002051
	BSA	1.839089	2.644047	2.380187	0.359279
	CSO	1.286495958	2.376673188	1.778589254	0.427509844
	GWO	0.815350217	1.448592622	1.069875804	0.24271658
	NBA	0.447641252	33.58522095	15.65608033	15.13540811
	MFO	10.81871505	45.00432867	23.4191773	13.22469995
	WCA	1.146E-31	1.817296686	3.63E-01	0.812719785
	HA-STFA	1.64E-04	8.95E-02	1.36E-02	4.87E-02
f10	FPA	-2.06261	-2.06261	-2.06261	8.58E-11
	MFPA	-2.06261	-2.06261	-2.06261	9.03E-16
	BAT	-2.06261	-2.06261	-2.06261	1.23E-11
	FF	-2.06261	-2.06261	-2.06261	1.40E-11
	GA	-2.05408	-2.03373	-2.0376	5.67E-03
	SA	-2.06261	-2.05137	-2.06102	2.39E-03
	BSA	-2.06261	-2.06261	-2.06261	3.73552E-12
	CSO	**-2.06261**	**-2.06261**	**-2.06261**	**0**
	GWO	-2.06261	-2.06261	-2.06261	4.54436E-11
	NBA	**-2.06261**	**-2.06261**	**-2.06261**	**0**
	MFO	**-2.06261**	**-2.06261**	**-2.06261**	**0**
	WCA	-2.06261	-2.06261	-2.06261	3.14018E-16
	HA-STFA	**-2.06261**	**-2.06261**	**-2.06261**	**0**
f11	FPA	-1	-1	-1	4.95E-10
	MFPA	**-1**	**-1**	**-1**	**0**
	BAT	-1	-0.78575	-0.93335	0.030208
	FF	-1	-1	-1	1.61E-09
	GA	-0.99996	-0.93624	-0.95722	0.030176
	SA	-0.99998	-0.78574	-0.92983	0.034068
	BSA	**-1**	**-1**	**-1**	**0**
	CSO	-1	-1	-1	1.5016E-06
	GWO	-1	-1	-1	1.12322E-06
	NBA	-1	-1	-1	1.28166E-06
	MFO	-1	-0.936245	-0.98725	0.028510611
	WCA	-1	-1	-1	1.89689E-06
	HA-STFA	**-1**	**-1**	**-1**	**0**
f12	FPA	-959.641	-959.641	-959.641	5.78E-13
	MFPA	-959.641	-959.641	-959.641	5.78E-13
	BAT	-959.641	-633.842	-831.837	105.893
	FF	-959.641	-559.787	-816.391	105.934
	GA	-32.806	-30.7438	-30.9245	0.474535
	SA	-956.366	-293.295	-542.529	166.754
	BSA	**-959.641**	**-959.641**	**-959.641**	**0**
	CSO	**-959.641**	**-959.641**	**-959.641**	**0**
	GWO	-959.641	-959.641	-959.641	1.659E-08
	NBA	**-959.641**	**-959.641**	**-959.641**	**0**
	MFO	**-959.641**	**-959.641**	**-959.641**	**0**
	WCA	**-959.641**	**-959.641**	**-959.641**	**0**
	HA-STFA	**-959.641**	**-959.641**	**-959.641**	**0**

TABLE 4. ***(Continued.)*** **Comparative results of HA-STFA with twelve state-of-the-art algorithms on standard benchmark functions.**

*f*13	FPA	-19.2085	-19.2085	-19.2085	3.11E-07
	MFPA	-19.2085	-19.2085	-19.2085	7.81E-15
	BAT	-19.2085	-16.2678	-18.7184	1.11466
	FF	-19.2085	-19.2085	-19.2085	1.44E-09
	GA	-1.72536	-1.66051	-1.66777	0.014743
	SA	-19.2085	-19.2084	-19.2085	4.40E-05
	BSA	-19.2085	-19.2085	-19.2085	8.550E-07
	CSO	**-19.2085**	**-19.2085**	**-19.2085**	**0**
	GWO	-19.2085	-19.2085	-19.2085	3.952E-08
	NBA	**-19.2085**	**-19.2085**	**-19.2085**	**0**
	MFO	-19.2085	-19.2085	-19.2085	1.250E-07
	WCA	**-19.2085**	**-19.2085**	**-19.2085**	**0**
	HA-STFA	**-19.2085**	**-19.2085**	**-19.2085**	**0**
*f*14	FPA	**0**	**0**	**0**	**0**
	MFPA	**0**	**0**	**0**	**0**
	BAT	1.82E-14	0.148546	0.034009	0.037992
	FF	1.10E-12	5.44E-11	1.71E-11	1.49E-11
	GA	1.89E-11	0.00015	1.14E-05	3.13E-05
	SA	9.61E-03	0.248544	0.082308	0.060543
	BSA	2.22045E-16	6.66134E-16	3.55271E-16	1.98603E-16
	CSO	2.22045E-16	5.19584E-14	1.06581E-14	2.30878E-14
	GWO	2.22045E-16	6.66134E-16	3.10862E-16	1.98603E-16
	NBA	2.22045E-16	4.44089E-16	3.10862E-16	1.21619E-16
	MFO	2.22045E-16	4.44089E-16	3.55271E-16	1.21619E-16
	WCA	2.22045E-16	2.88658E-15	1.15463E-15	1.09232E-15
	HA-STFA	2.22E-16	2.22E-16	2.22E-16	0
*f*15	FPA	-186.731	-186.731	-186.731	7.34E-06
	MFPA	-186.731	-186.731	-186.731	5.28E-15
	BAT	-186.731	-79.4109	-173.261	31.9251
	FF	-186.731	-186.731	-186.731	8.55E-08
	GA	-69.9381	-16.6114	-24.3187	13.1135
	SA	-186.731	-186.731	-186.731	9.25E-07
	BSA	-186.731	-186.731	-186.731	2.74592E-06
	CSO	-186.731	-186.731	-186.731	7.36118E-08
	GWO	-186.731	-186.731	-186.731	2.57055E-07
	NBA	-186.731	-186.731	-186.731	2.00972E-14
	MFO	**-186.731**	**-186.731**	**-186.731**	**0**
	WCA	**-186.731**	**-186.731**	**-186.731**	**0**
	HA-STFA	-186.731	-186.731	-186.731	5.72E-08
*f*16	FPA	0.500091	0.500091	0.500091	3.66E-09
	MFPA	0.500091	0.500091	0.500091	1.94E-08
	BAT	**0.5**	**0.5**	**0.5**	**0**
	FF	0.500091	0.500091	0.500091	2.09E-09
	GA	0.539118	0.540138	0.539745	0.000243
	SA	0.500096	0.500133	0.500110	1.21E-05
	BSA	0.500091	0.500091	0.500091	0
	CSO	0.500091	0.500091	0.500091	0
	GWO	0.500091	0.500091	0.500091	1.56877E-08
	NBA	0.500091	0.500091	0.500091	0
	MFO	0.500091	0.500091	0.500091	0
	WCA	0.500091	0.500091	0.500091	1.11022E-16
	HA-STFA	0.500091	0.500091	0.500091	0

TABLE 4. *(Continued.)* **Comparative results of HA-STFA with twelve state-of-the-art algorithms on standard benchmark functions.**

*f*17	FPA	7.11E-27	2.25E-20	1.98E-21	5.85E-21
	MFPA	**0**	**0**	**0**	**0**
	BAT	3.28E-12	0.658064	0.117083	0.239276
	FF	4.50E-13	2.47E-10	6.15E-11	5.94E-11
	GA	5.23E-14	8.14E-11	1.60E-11	2.00E-11
	SA	3.11E-06	0.047376	0.003245	0.008865
	BSA	2.77E-32	6.16E-32	4.12919E-32	1.85658E-32
	CSO	6.16298E-32	4.39411E-27	8.81098E-28	1.96384E-27
	GWO	6.01311E-12	7.90539E-10	3.49105E-10	3.63548E-10
	NBA	2.77334E-32	6.16298E-32	4.80712E-32	1.85658E-32
	MFO	2.77334E-32	3.48208E-31	1.1525E-31	1.3461E-31
	WCA	2.77334E-32	3.23248E-30	7.01963E-31	1.41505E-30
	HA-STFA	1.54E-14	1.88E-13	5.64E-14	7.41E-14
*f*18	FPA	2.43E-29	1.39E-24	9.74E-26	2.79E-25
	MFPA	2.69E-67	5.62E-61	3.31E-62	1.07E-61
	BAT	6.37E-13	4.90E-10	7.86E-11	9.80E-11
	FF	2.15E-10	5.55E-08	1.18E-08	1.18E-08
	GA	4.41E-15	1.66E-12	2.78E-13	4.12E-13
	SA	1.34E-09	4.87E-05	5.49E-06	1.05E-05
	BSA	2.5055E-308	3.397E-304	7.1507E-305	0
	CSO	9.3766E-101	1.52085E-89	3.04617E-90	6.79897E-90
	GWO	2.3788E-308	2.9326E-308	2.5448E-308	0
	NBA	**2.2301E-308**	**2.4768E-308**	**2.3336E-308**	**0**
	MFO	2.41437E-90	1.17286E-89	6.14007E-90	5.10162E-90
	WCA	4.74943E-35	4.08387E-34	1.57651E-34	1.55456E-34
	HA-STFA	7.30E-137	6.40E-132	1.30E-132	2.90E-132
*f*19	FPA	5.06E-34	1.82E-26	6.92E-28	3.32E-27
	MFPA	1.14E-68	5.56E-50	2.08E-51	1.02E-50
	BAT	3.63E-13	2.28E-11	7.44E-12	6.70E-12
	FF	7.50E-13	5.75E-11	1.56E-11	1.35E-11
	GA	1.63E-14	9.16E-12	1.53E-12	2.37E-12
	SA	1.61E-07	8.77E-03	1.84E-03	2.48E-03
	BSA	2.6039E-308	1.2547E-307	5.2332E-308	0
	CSO	1.4942E-108	1.7859E-99	3.6201E-100	7.9606E-100
	GWO	2.9069E-308	4.463E-306	9.3458E-307	0
	NBA	**2.2596E-308**	**4.9887E-308**	**3.627E-308**	**0**
	MFO	2.4924E-308	7.4666E-308	4.3903E-308	0
	WCA	1.58413E-44	9.45753E-40	2.02183E-40	4.164E-40
	HA-STFA	3.87E-308	9.33E-101	1.86E-101	4.17E-101
*f*20	FPA	-78.3323	-78.3323	-78.3323	1.45E-14
	MFPA	-78.3323	-78.3323	-78.3323	1.45E-14
	BAT	-78.3323	-64.1956	-75.505	5.75136
	FF	-78.3323	-78.0204	-78.3219	0.056953
	GA	-78.3323	-50.0589	-58.5409	9.53754
	SA	-78.3323	-75.5983	-78.2412	0.499163
	BSA	-78.3323	-78.3309	-78.3319	0.00061313
	CSO	**-78.3323**	**-78.3323**	**-78.3323**	**0**
	GWO	-78.3323	-78.3323	-78.3323	3.37015E-08
	NBA	**-78.3323**	**-78.3323**	**-78.3323**	**0**
	MFO	**-78.3323**	**-78.3323**	**-78.3323**	**0**
	WCA	**-78.3323**	**-78.3323**	**-78.3323**	**0**
	HA-STFA	**-78.3323**	**-78.3323**	**-78.3323**	**0**
*f*21	FPA	0.998004	0.998004	0.998004	1.30E-14
	MFPA	0.998004	0.998004	0.998004	1.80E-16
	BAT	0.998004	22.9006	7.5505	5.72583
	FF	0.998004	2.98216	1.48794	0.609329
	GA	12.6705	12.6705	12.6705	1.44E-13
	SA	0.998004	22.9006	5.67102	5.5792
	BSA	0.998004	0.998004	0.998004	1.28405E-08
	CSO	0.998004	0.998004	0.998004	6.28037E-16
	GWO	0.998004	0.998004	0.998004	5.17488E-09
	NBA	0.998004	0.998004	0.998004	5.3175E-08
	MFO	**0.998004**	**0.998004**	**0.998004**	**1.11022E-16**
	WCA	0.998004	0.998004	0.998004	2.22045E-16
	HA-STFA	0.998004	0.998004	0.998004	9.59E-15

TABLE 5. The number of functions for which each algorithm obtained the best results.

Algorithms	Number of functions	Functions
FPA	1	*f14*
MFPA	9	*f1, f3, f4, f5, f8, f9, f11, f14, f17*
BAT	1	*f16*
FF	0	-
GA	0	-
SA	0	-
BSA	4	*f1, f6, f11, f12*
CSO	5	*f7, f10, f12, f13, f20*
GWO	0	-
NBA	8	*f2, f7, f10, f12, f13, f18, f19, f20*
MFO	5	*f10, f12, f15, f20, f21*
WCA	4	*f12, f13, f15, f20*
HA-STFA	7	*f1, f7, f10, f11, f12, f13, f20*

TABLE 6. Results of wilcoxon signed rank test between HA-STFA and other twelve algorithms.

Algorithms	Results
FPA	H_0
MFPA	H_0
BAT	H_1 (0.05)
FF	H_1 (0.05)
GA	H_1 (0.05)
SA	H_1 (0.05)
BSA	H_1 (0.05)
CSO	H_1 (0.1)
GWO	H_1 (0.05)
NBA	H_1 (0.1)
MFO	H_1 (0.05)
WCA	H_1 (0.05)

For the 30-dimensional multimodal functions, the hybrid HA–STFA algorithm achieved very good results on 5 out of 6 benchmark functions: *f1*, *f3*, *f4*, *f5* and *f20*. It tied for the first place with MFPA and BSA for *f1*. For *f3*, *f4* and *f5*, HA-STFA ranked second only to MFPA. HA-STFA, CSO, NBA, MFO and WCA successfully found the global minimum for *f20*. Our HA–STFA only struggled in finding the global minimum with *f9*; it came in eighth, losing by small margins to the other seven algorithms.

In summary, HA-STFA algorithm was able to obtain very good results on both unimodal and multimodal functions; it successfully located the global or near-global minima for all tested functions. Moreover, HA-STFA was also very robust—the deviation between runs was very low—indicated by low standard deviations shown in Table 4. The success was due to the combination of the advantages from HA and STFA— the good exploration capability of HA and the good exploitation capability of STFA. In addition, cloning also retained good solutions for the next iteration.

Further analyzing the results in Table 4, Table 5 shows the number of functions, for which each algorithm obtained the best result. These are the results, ranked from best to worst: MFPA (9 functions); NBA (8 functions); HA-STFA (7 functions); CSO and MFO (5 functions); BSA and WCA (4 functions); FPA and BAT (1 function); FF, GA, SA and GWO (0 function).

Finally, the Wilcoxon signed-rank test was determined whether HA-STFA performed significantly better than the other twelve algorithms. The Wilcoxon test was used, rather than t-test, because we cannot guarantee that the results will be normally distributed [57]. We performed a one-tailed hypothesis test, using 0.05 and 0.1 significance levels:

$$H_0 : M_{HA-STFA} \geq M_{Compared\ Method}$$
$$H_1 : M_{HA-STFA} < M_{Compared\ Method}$$

The significance tests are shown in Table 6. In Table 6, with the exception of FPA and MFPA, it can be seen that the null hypothesis was rejected. In other words, HA-STFA performed significantly better than the others. More precisely, HA-STFA outperformed BAT, FF, GA, SA, BSA, GWO, MFO and WCA with statistical significance of 0.05, and outperformed CSO and NBA with statistical significance of 0.10. For FPA and MFPA, however, the null hypothesis was

TABLE 7. Performances of four algorithms on real-life engineering problems.

Problems	Dimension	Algorithms	Mean	SD
R1	2	IWO	**169.844**	**0**
		Pro.DEPCX	169.844	2.8E−14
		QAIWO	**169.844**	**0**
		HA-STFA	**169.844**	**0**
R2	3	IWO	2.96439E+06	1.7E−9
		Pro.DEPCX	2.96447E+06	4.6E−10
		QAIWO	**2.96437E+06**	**3.2E−10**
		HA-STFA	2.96438E+06	1.20
R3	3	IWO	**4.21421**	**0**
		Pro.DEPCX	4.21421	8.88178E−16
		QAIWO	**4.21421**	**0**
		HA-STFA	4.21421	3.48E-02
R4	20	IWO	1.294402	0.088571
		Pro.DEPCX	**1.08687**	**0.109388**
		QAIWO	1.135443	0.08852
		HA-STFA	1.705059	0.122185
R5	5	IWO	3.5E−02	5.1E−02
		Pro.DEPCX	-	-
		QAIWO	3.4E−03	6.2E−04
		HA-STFA	**1.59E-07**	**1.67E-07**

accepted, indicating that HA-STFA did not perform better than FPA and MFPA. Comparing HA-STFA with FPA and MFPA (see Table 3), we can observe, in 21 functions:

- HA-STFA performed better than FPA on 9 functions, worse on 3 functions, and equally well on 9 functions;
- HA-STFA performed better than MFPA on 4 functions, worse on 7 functions, and equally well on 10 functions.

B. RESULTS FOR REAL-LIFE ENGINEERING PROBLEMS

To further demonstrate the capability of the hybrid HA–STFA, a set of five real-life engineering design problems were used as benchmarks. In Table 7, HA-STFA was compared with Invasive weed optimization (IWO), Quadratic approximation invasive weed optimization (QAIWO) and Differential Evolution with probabilistic Parent Centric Crossover (Pro.DEPCX) algorithms. Table 7 reports means and standard deviations (SD) from 30 independent runs, when the termination condition was met. The mean and SD reveal the effectiveness and consistency of the algorithm over 30 runs. A small SD signified that the algorithm is more consistent. Table 7 shows that the mean and SD for all algorithms were similar.

V. CONCLUSION

We designed a new hybrid algorithm for solving optimization problems. It combines the search features of two nature-inspired optimization algorithms: Hydrozoan Algorithm (HA) and Sea Turtle Foraging Algorithm (STFA), to achieve better exploration and exploitation capabilities than the individual algorithms. The performance of our HA-STFA was examined with twenty-one benchmark functions and compared to twelve other widely used algorithms: FPA, MFPA, BAT, FF, GA, SA, BSA, CSO, GWO, NBA, MFO and WCA. The results from the Wilcoxon signed-rank test indicated that HA-STFA, FPA and MFPA performed equally well, whereas HA-STFA performed significantly better than BAT, FF, GA, SA, BSA, CSO, GWO, NBA, MFO and WCA. For future work, additional crossover operators as well as a self-adaptive strategy for selecting crossover operators will be included to further enhance the search capability.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

DATA AVAILABILITY

The details of the functions used to support the findings of this study are included within the article.

REFERENCES

[1] M. Thakur, "A new genetic algorithm for global optimization of multimodal continuous functions," *J. Comput. Sci.*, vol. 5, no. 2, pp. 298–311, Mar. 2014.

[2] H. Shan, T. Yasuda, and K. Ohkura, "A self adaptive hybrid enhanced artificial bee colony algorithm for continuous optimization problems," *Biosystems*, vols. 132–133, pp. 43–53, Jun. 2015.

[3] D. A. Wood, "Hybrid bat flight optimization algorithm applied to complex wellbore trajectories highlights the relative contributions of Metaheuristic components," *J. Natural Gas Sci. Eng.*, vol. 32, pp. 211–221, May 2016.

[4] B. Farnad, A. Jafarian, and D. Baleanu, "A new hybrid algorithm for continuous optimization problem," *Appl. Math. Model.*, vol. 55, pp. 652–673, Mar. 2018.

[5] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and metaheuristic algorithms and their relevance to the real world: A survey," *Int. J. Comput. Eng. Res. Trends*, vol. 2, no. 5, pp. 296–304, 2015.

[6] B. A. Sawyerr, A. O. Adewumi, and M. M. Ali, "Real-coded genetic algorithm with uniform random local search," *Appl. Math. Comput.*, vol. 228, pp. 589–597, Feb. 2014.

[7] O. Hegazy, O. S. Soliman, and M. A. Salam, "Comparative study between FPA, BA, MCS, ABC, and PSO algorithms in training and optimizing of LS-SVM for stock market prediction," *Int. J. Adv. Comput. Res.*, vol. 5, no. 18, pp. 35–45, Mar. 2015.

[8] G. Khademi, H. Mohammadi, and D. Simon, "Hybrid invasive weed/biogeography-based optimization," *Eng. Appl. Artif. Intell.*, vol. 64, pp. 213–231, Sep. 2017.

[9] T. O. Ting, X. S. Yang, S. Cheng, and K. Huang, "Hybrid metaheuristic algorithms: Past, present, and future," in *Recent Advances in Swarm Intelligence and Evolutionary Computation*. Cham, Switzerland: Springer, 2015, pp. 71–83.

[10] Y.-C. Chuang, C.-T. Chen, and C. Hwang, "A real-coded genetic algorithm with a direction-based crossover operator," *Inf. Sci.*, vol. 305, pp. 320–348, Jun. 2015.

[11] M. J. Mahmoodabadi and A. R. Nemati, "A novel adaptive genetic algorithm for global optimization of mathematical test functions and real-world problems," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 4, pp. 2002–2021, Dec. 2016.

[12] Q. Zhang, R. M. Ogren, and S.-C. Kong, "A comparative study of biodiesel engine performance optimization using enhanced hybrid PSO–GA and basic GA," *Appl. Energy*, vol. 165, pp. 676–684, Mar. 2016.

[13] F. Jiang, H. Xia, Q. A. Tran, Q. M. Ha, N. Q. Tran, and J. Hu, "A new binary hybrid particle swarm optimization with wavelet mutation," *Knowl.-Based Syst.*, vol. 130, pp. 90–101, Aug. 2017.

[14] H. Haklı and H. Uğuz, "A novel particle swarm optimization algorithm with levy flight," *Appl. Soft Comput.*, vol. 23, pp. 333–345, Oct. 2014.

[15] F. Javidrad and M. Nazari, "A new hybrid particle swarm and simulated annealing stochastic optimization method," *Appl. Soft Comput.*, vol. 60, pp. 634–654, Nov. 2017.

[16] H. Garg, "A hybrid PSO-GA algorithm for constrained optimization problems," *Appl. Math. Comput.*, vol. 274, pp. 292–305, Feb. 2016.

[17] X. S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation* (Lecture Notes in Computer Science), vol. 7445. Berlin, Germany: Springer, 2012, pp. 240–249.

[18] E. Nabil, "A modified flower pollination algorithm for global optimization," *Expert Syst. Appl.*, vol. 57, pp. 192–203, Sep. 2016.

[19] Y. Zhou, R. Wang, and Q. Luo, "Elite opposition-based flower pollination algorithm," *Neurocomputing*, vol. 188, pp. 294–310, May 2016.

[20] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecol. Informat.*, vol. 1, no. 4, pp. 355–366, Dec. 2006.

[21] Y. R. Naidu and A. K. Ojha, "A hybrid version of invasive weed optimization with quadratic approximation," *Soft Comput.*, vol. 19, no. 12, pp. 3581–3598, Dec. 2015.

[22] X. Cai, Z. Hu, and Z. Fan, "A novel memetic algorithm based on invasive weed optimization and differential evolution for constrained optimization," *Soft Comput.*, vol. 17, no. 10, pp. 1893–1910, Oct. 2013.

[23] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.

[24] X. Chen, H. Tianfield, W. Du, and G. Liu, "Biogeography-based optimization with covariance matrix based migration," *Appl. Soft Comput.*, vol. 45, pp. 71–85, Aug. 2016.

[25] W. Gong, Z. Cai, C. X. Ling, and H. Li, "A real-coded biogeography-based optimization with mutation," *Appl. Math. Comput.*, vol. 216, no. 9, pp. 2749–2758, Jul. 2010.

[26] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization* (Studies in Computational Intelligence), vol. 284. Berlin, Germany: Springer, 2010, pp. 65–74.

[27] Q. Liu, L. Wu, W. Xiao, F. Wang, and L. Zhang, "A novel hybrid bat algorithm for solving continuous optimization problems," *Appl. Soft Comput.*, vol. 73, pp. 67–82, Dec. 2018.

[28] G. Yildizdan and Ö. K. Baykan, "A novel modified bat algorithm hybridizing by differential evolution algorithm," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112949, doi: 10.1016/j.eswa.2019.112949.

[29] M. R. Ramli, Z. A. Abas, M. I. Desa, Z. Z. Abidin, and M. B. Alazzam, "Enhanced convergence of bat algorithm based on dimensional and inertia weight factor," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 31, no. 4, pp. 452–458, Oct. 2019.

[30] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications* (Lecture Notes in Computer Science), vol. 5792. Berlin, Germany: Springer, 2009, pp. 169–178.

[31] R. M. Rizk-Allah, E. M. Zaki, and A. A. El-Sawy, "Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems," *Appl. Math. Comput.*, vol. 224, pp. 473–483, Nov. 2013.

[32] A. Ritthipakdee, A. Thammano, N. Premasathian, and D. Jitkongchuen, "Firefly mating algorithm for continuous optimization problems," *Comput. Intell. Neurosci.*, vol. 2017, pp. 1–10, 2017.

[33] S. Khalilpourazari and S. Khalilpourazary, "Optimization of time, cost and surface roughness in grinding process using a robust multi-objective dragonfly algorithm," *Neural Comput. Appl.*, to be published, doi: 10.1007/s00521-018-3872-8.

[34] S. H. R. Pasandideh and S. Khalilpourazari, "Sine cosine crow search algorithm: A powerful hybrid meta heuristic for global optimization," 2018, *arXiv:1801.08485*. [Online]. Available: http://arxiv.org/abs/1801.08485

[35] S. Khalilpourazari and S. Khalilpourazary, "SCWOA: An efficient hybrid algorithm for parameter optimization of multi-pass milling process," *J. Ind. Prod. Eng.*, vol. 35, no. 3, pp. 135–147, Apr. 2018.

[36] S. Khalilpourazari and S. Khalilpourazary, "An efficient hybrid algorithm based on water cycle and moth-flame optimization algorithms for solving numerical and constrained engineering optimization problems," *Soft Comput.*, vol. 23, no. 5, pp. 1699–1722, Mar. 2019.

[37] S. Khalilpourazari and S. H. R. Pasandideh, "Modeling and optimization of multi-item multi-constrained EOQ model for growing items," *Knowl.-Based Syst.*, vol. 164, pp. 150–162, Jan. 2019.

[38] D. Tansui and A. Thammano, "Nature-inspired optimization method: Hydrozoan algorithm for solving continuous problems," in *Proc. 18th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Kanazawa, Japan, Jun. 2017, pp. 23–28.

[39] D. Tansui and A. Thammano, "Sea turtle foraging algorithm for continuous optimization problems," in *Proc. WCSE*, Tokyo, Japan, 2016, pp. 678–681.

[40] L. Leclère, R. R. Copley, T. Momose, and E. Houliston, "Hydrozoan insights in animal development and evolution," *Current Opinion Genet. Develop.*, vol. 39, pp. 157–167, Aug. 2016.

[41] P. Cartwright and A. M. Nawrocki, "Character evolution in hydrozoa (phylum Cnidaria)," *Integrative Comparative Biol.*, vol. 50, no. 3, pp. 456–472, Sep. 2010.

[42] S. Berking, "A model for budding in hydra: Pattern formation in concentric rings," *J. Theor. Biol.*, vol. 222, no. 1, pp. 37–52, May 2003.

[43] S. Berking, "Principles of branch formation and branch patterning in hydrozoa," *Int. J. Develop. Biol.*, vol. 50, nos. 2–3, pp. 123–134, 2006.

[44] S. Roy, S. Biswas, and S. S. Chaudhuri, "Nature-inspired swarm intelligence and its applications," *Int. J. Modern Edu. Comput. Sci.*, vol. 12, pp. 55–65, 2014.

[45] S. C. Chu, H. C. Huang, J. F. Roddick, and J. S. Pan, "Overview of algorithms for swarm intelligence," in *Computational Collective Intelligence. Technologies and Applications*. Berlin, Germany: Springer, 2011, pp. 28–41.

[46] J. Okuyama, O. Abe, H. Nishizawa, M. Kobayashi, K. Yoseda, and N. Arai, "Ontogeny of the dispersal migration of green turtle (Chelonia mydas) hatchlings," *J. Experim. Mar. Biol. Ecology*, vol. 379, nos. 1–2, pp. 43–50, Oct. 2009.

[47] G. A. Nevitt, "Sensory ecology on the high seas: The odor world of the procellariiform seabirds," *J. Experim. Biol.*, vol. 211, no. 11, pp. 1706–1713, Jun. 2008.

[48] C. S. Endres and K. J. Lohmann, "Perception of dimethyl sulfide (DMS) by loggerhead sea turtles: A possible mechanism for locating high-productivity oceanic regions for foraging," *J. Experim. Biol.*, vol. 215, no. 20, pp. 3535–3538, Oct. 2012.

[49] X. B. Meng. (2015). *Bird Swarm Algorithm*. MathWorks File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/51256-bird-swarm-algorithm-bsa

[50] X. B. Meng. (2015). *CSO*. MathWorks File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/48204-cso

[51] S. Mirjalili. (2018). *Grey Wolf Optimizer*. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/44974-grey-wolf-optimizer-gwo,MathWorks File Exchange

[52] X. B. Meng. (2015). *Novel Bat Algorithm*. MathWorks File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/51258-novel-bat-algorithm-nba

[53] S. Mirjalili. (2018). *Moth-Flame Optimization Algorithm*. MathWorks File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52269-moth-flame-optimization-mfo-algorithm

[54] A. Sadollah. (2016). *Water Cycle Algorithm*. MathWorks File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/56339-water-cycle-algorithm-wca

[55] M. Ali, M. Pant, and V. P. Singh, "Two modified differential evolution algorithms and their applications to engineering design problems," *World J. Model. Simul.*, vol. 6, no. 1, pp. 72–80, 2010.

[56] Y. Zhou, Q. Luo, and H. Chen, "A novel differential evolution invasive weed optimization algorithm for solving nonlinear equations systems," *J. Appl. Math.*, vol. 2013, pp. 1–18, 2013.

[57] D. S. Moore and G. P. McCabe, *Introduction to the Practice of Statistics*, New York, NY, USA: W. H. Freeman, 1989.

DARANAT TANSUI is currently pursuing the Ph.D. degree with the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. Her current research interests include swarm intelligence, memetic algorithms, and optimization techniques.

ARIT THAMMANO is currently an Associate Professor with the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. His current research interests include evolutionary algorithms, memetic algorithms, swarm intelligence, and optimization techniques.