AN EFFECTIVE HYBRID FIREFLY ALGORITHM WITH THE CUCKOO SEARCH FOR ENGINEERING OPTIMIZATION PROBLEMS

MOHAMED A. TAWHID
Department of Mathematics and Statistics, Faculty of Science
Thompson Rivers University, Kamloops, BC, Canada V2C 0C8

AHMED F. ALI*
Department of Computer Science, Faculty of Computers & Informatics
Suez Canal University, Ismailia, Egypt
Postdoctoral Fellow
Department of Mathematics and Statistics, Faculty of Science
Thompson Rivers University, Kamloops, BC, Canada V2C 0C8

(Communicated by Zhipeng Cai)

Abstract. Firefly and cuckoo search algorithms are two of the most widely used nature-inspired algorithms due to their simplicity and inexpensive computational cost when they applied to solve a wide range of problems. In this article, a new hybrid algorithm is suggested by combining the firefly algorithm and the cuckoo search algorithm to solve constrained optimization problems (COPs) and real-world engineering optimization problems. The proposed algorithm is called Hybrid FireFly Algorithm and Cuckoo Search (HFFACS) algorithm. In the HFFACS algorithm, a balance between the exploration and the exploitation processes is considered. The main drawback of the firefly algorithm is it is easy to fall into stagnation when the new solution is not better than its previous best solution for several generations. In order to avoid this problem, the cuckoo search with Lévy flight is invoked to force the firefly algorithm to escape from stagnation and to avoid premature convergence. The proposed algorithm is applied to six benchmark constrained optimization problems and five engineering optimization problems and compared against four algorithms to investigate its performance. The numerical experimental results show the proposed algorithm is a promising algorithm and can obtain the optimal or near optimal solution within a reasonable time.

1. Introduction. Meta-heuristics are global search methods that cover all heuristics methods, which give evidence of getting good solutions for the problem of interest within a reasonable time. Meta-heuristics structures are mainly based on simulating nature and artificial intelligence tools. The main two components of meta-heuristic methods are exploration (diversification) process and exploitation process (intensification). In exploring process, the search space is explored to avoid

2010 Mathematics Subject Classification. Primary: 68T01, 68T20, 68U20, 90C90, 90C30.
Key words and phrases. Firefly algorithm, cuckoo search algorithm, global optimization problems, constrained optimization problems, engineering optimization problems.

The research of the 1st author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).*

* Corresponding author: Ahmed F. Ali.
trapping in local minima, while in exploitation process the best-obtained solution is improved. Meta-heuristics can be categorized into two classes; population-based methods and point-to-point based methods. The population-based methods include Ant Colony Optimization (ACO) [7], Genetic Algorithms (GAs) [15], Scatter Search (SS) [12], etc, while the point-to-point based methods include Tabu Search (TS) [11], Simulated Annealing (SA) [18], Variable Neighborhood Search (VNS) [26], [27], Iterated Local Search (ILS) [35]. In the last two decades, new meta-heuristics methods have been inspired from nature. These methods are called natural-inspired methods or swarm intelligence algorithms, such as Artificial Bee Colony (ABC) [16], Particle Swarm Optimization (PSO) [17], Bacterial foraging [31], Bat Algorithm (BA)[46], Wolf search [36], Cat swarm [6], Cuckoo Search [45], Firefly Algorithm (FA) [46], Fish swarm/school [19], etc. These methods suffer from slow convergence, and they can fall into stagnation when no particle discovers a position that is better than its previous best position for several generations.

In order to circumvent the drawbacks of each metaheuristic algorithm, several researchers have tried to combine two metaheuristic algorithms (known as hybridization) or integrate a metaheuristic algorithm with a local search algorithm (known as memetic). Hybridization alludes to the procedure of integrating the best aspects of two or more algorithms to structure a good-attribute algorithm. One of the most vastly employed strategies to suggest hybrid metaheuristics is by integrating metaheuristics with local search algorithms. This integration is called memetic algorithms. We refer to the Interested readers to [29] for a good review on memetic algorithms. Another approach to frame hybrid metaheuristics is by integrating two or more metaheuristics [4]; we follow this approach in this work. Hybrid metaheuristics can outperform their ingredient algorithms due to their integration [30]. This feature is the main motivation for the study of hybrid metaheuristics. Many hybrid metaheuristics have been developed to enhance performance and to get global optima [24], [28]. Although some of these enhancements and developments are important, the new hybrid metaheuristic strategies need more investigations. Recent research directions in hybrid metaheuristics engage various directions. For example, the first direction is the application of hybrid metaheuristics to various kinds of optimization problems, such as unconstrained optimization problems [1], constrained optimization [3], [43], integer programming, and minimax problems [2]. The second direction is the application of hybrid metaheuristics to real-life applications [20], [21]. The third direction is to integrate metaheuristics to a hybrid algorithm [22], [4]. The fourth direction is how to hybridize a given set of metaheuristics into a single algorithm [34], [10].

The objective of this paper is to propose a new hybrid algorithm by combining two of the most promising swarm intelligence algorithms, which are called firefly algorithm and cuckoo search algorithm. The proposed algorithm is called Hybrid FireFly Algorithm and Cuckoo Search (HFFACS) algorithm. In the HFFACS algorithm, the balance between the exploration and the exploitation process is considered to overcome the slow convergence of the standard firefly algorithm. The standard firefly algorithm has a good ability to perform exploration and exploitation search. However, it is easy to fall into stagnation when the new solutions are not better than the current best solution. The cuckoo search algorithm and the Lévy flight with their capability of performing a wide exploration search in the search space are invoked to help the firefly algorithm to jump out of stagnation.
and to avoid the premature convergence. In order to investigate the proposed algorithm, HFFACS is tested on six benchmark constrained optimization problems and five engineering optimization problems and compared against four algorithms. The computational results indicate that the proposed algorithm is a promising algorithm and can get the optimal or near optimal solution within an acceptable time.

The rest of this paper is organized as follows. The definition of the constrained optimization problem is presented in Section 2. The main concepts of the firefly algorithm and cuckoo search algorithm are discussed in Sections 3, 4 respectively. The proposed algorithm is shown in detail in Section 5. In Section 6, the experimental results and all the comparative experiments results between the proposed algorithm and the other algorithms are reported. The description of the engineering optimization problems and the experimental results of the proposed algorithm with these problems are presented in Section 7. Finally, the conclusion and future work make up Section 8.

2. Constrained optimization problem. The constrained optimization problems and constraint handling are one of the most challenging in many applications. A general form of a constrained optimization is defined as follows.

Minimize \( f(x), x = (x_1, x_2, \ldots, x_n)^T \),

Subject to \( g_i(x) \leq 0, \quad i = \{1, \ldots, m\}, \)
\( h_j(x) = 0, \quad j = \{1, \ldots, l\}, \)
\( x_l \leq x_i \leq x_u \)

Where \( f(x) \) is the objective function, \( x \) is the vector of \( n \) variables, \( g_i(x) \) are inequality constraints, \( h_j(x) = 0 \) are equality constraints, \( x_l, x_u \) are variables bounds.

3. Overview of firefly algorithm. In the following subsections, the main concepts and characteristics of the firefly algorithm are highlighted as follows.

3.1. Main concepts. The firefly algorithm (FA) is a nature-inspired population-based algorithm, proposed by Xin-She Yang in late 2007 and 2008 [44], [45]. According to many recent publications, FA is a promising algorithm and outperforms other meta-heuristics algorithms such as the algorithms in [23], [46]. FA has three flashing characteristics and idealized rules, which are inspired by the real fireflies. These rules can be summarized as follows:

1. All fireflies are unisex, and they will move to other fireflies regardless of their sex.
2. The attractiveness of the firefly is proportional to its brightness, and it decreases as the distance from the other firefly increases. The firefly with less bright will move towards the brighter one. The firefly will move randomly if there is no brighter firefly than a particular one.
3. The brightness of a firefly is determined by the value of the objective function.

3.2. Attractiveness and brightness. The attractiveness of a firefly is determined by its brightness, which is associated with the objective function. The firefly with the less bright is attracted to the brighter firefly. The brightness (light intensity) \( I \) of firefly decreases with the distance from its source, and light is absorbed by
the environment. It is known that light intensity $I(r)$ varies following the inverse square law as follows.

$$I(r) = \frac{I_0}{r^2} \quad (2)$$

Where $I_0$ is the light intensity at the source, $r$ is the distance between any two fireflies. With the fixed light absorption coefficient and to avoid the singularity at $r = 0$ in the expression in 2. The combined effect of both the inverse square law and absorption can be approximated as the following Gaussian form

$$I(r) = I_0 e^{(-\gamma r^2)} \quad (3)$$

Since a firefly, attractiveness is proportional to the light intensity seen by adjacent fireflies, the attractiveness function of the firefly can be defined by:

$$B(r) = B_0 e^{(-\gamma r^2)} \quad (4)$$

Where $B_0$ is the initial attractiveness at $r = 0$, $\gamma$ is the media light absorption coefficient.

3.3. The distance between two fireflies. At the position, $x_i$ and $x_j$, the distance between any two fireflies $i$ and $j$ can be defined as Euclidian or Cartesian distance as follows [45], [44], [23].

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \quad (5)$$

Where $x_{i,k}$ is the $k^{th}$ component of spatial coordinates $x_i$ of $i^{th}$ firefly and $d$ is the number of dimensions. For $d = 2$, Equation 2 can be defined as follows.

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (6)$$

3.4. Firefly movement. The firefly $i$ is attracted and moves to the firefly $j$ if the firefly $j$ is brighter than firefly $i$. The movement of the firefly $i$ to firefly $j$ can be defined as follows.

$$x_i = x_i + B_0 \exp(-\gamma r_{ij}^2)(x_j - x_i) + \alpha (rand - 0.5) \quad (7)$$

3.5. Firefly algorithm. The main steps of the firefly algorithm are presented in Algorithm 1. In Algorithm 1, the firefly algorithm starts by setting the initial values of the main parameters in the algorithms such as the randomization parameter $\alpha$, firefly attractiveness $B_0$, media light absorption coefficient $\gamma$, population size $P$ and maximum generation number MGN. The initial population is generated randomly, and the objective function is calculated for each solution in the population. The following steps are repeated until the termination criterion satisfied which is to reach the desired number of iterations MGN. For each solution $x_i$ and $x_j$, $i = 1, \ldots, P$ and $j = 1, \ldots, i$. If the objective function of firefly $j$ is better than the objective function of firefly $i$, the firefly $i$ will move towards the firefly $j$ as shown in Equation 7. The attractiveness varies with distance $r$ via $\exp(-\gamma r^2)$ as shown in Equation 4. Each solution $x_i$ is evaluated in the population, and the corresponding light intensity $I_i$ of each solution is updated. The best-found solution is assigned $x_{best}$ and produced as an optimal solution.
Algorithm 1 Firefly algorithm

1: Set the initial values of the randomization parameter $\alpha$, firefly attractiveness $\beta_0$, media light absorption coefficient $\gamma$, population size $P$ and maximum generation number $MGN$.
2: Generate the initial population $x_i$ randomly, $i = \{1, \ldots, P\}$ \{Initialization\}
3: Evaluate the fitness function $f(x_i)$ of all solutions in the population
4: Light intensity $I_i$ at $x_i$ is determined by $f(x_i)$
5: Set $t = 0$
6: repeat
7: \hspace{1em} for $(i = 1; i < P; i++)$ do
8: \hspace{2em} for $(j = 1; j < i; j++)$ do
9: \hspace{3em} if $I_{i(t+1)} < I_{j(t+1)}$ then
10: \hspace{4em} Move firefly $i$ towards $j$
11: \hspace{2em} end if
12: \hspace{2em} Obtain attractiveness $\beta$, where $\beta(r) = \beta_0 e^{-\gamma r^2}$
13: \hspace{2em} Evaluate the fitness function $f(x_i)$ of all solutions in the population
14: \hspace{2em} Update light intensity $I_i$
15: \hspace{2em} end for
16: \hspace{1em} end for
17: Rank the solutions and keep the best solution $x_{best}$ found so far in the population
18: $t = t + 1$
19: until $t > MGN$
20: Produce the optimal solution.

4. Overview of cuckoo search.

4.1. Main concepts. The cuckoo search (CS) algorithm is a nature-inspired population-based algorithm inspired from the reproduction strategy of the cuckoo birds. The cuckoo birds lay their eggs in communal nests, and they may remove other’s eggs to increase the probability of hatching their own eggs [32].

Some host bird can discover the eggs are not its own and throw these eggs away or abandons its nest and build a new nest in a new place. Some kind cuckoo birds can mimic the color and the pattern of the eggs of a few host birds to reduce the probability of discovering the intruding eggs. The cuckoos laid their eggs in a nest where the host bird just laid its eggs since the cuckoo eggs are hatching earlier than the host bird eggs. Once the eggs are hatching, the cuckoo chick’s starts to propel the host eggs out of the nest to increase its share of food provided by its host bird.

4.2. Lèvy flights. Lèvy flight [5] is a random walk in which the step-lengths are distributed according to a heavy-tailed probability distribution. After a large number of steps, the distance from the origin of the random walk tends to a stable distribution.

4.3. Cuckoo search characteristic. The cuckoo search algorithm is based on the following rules

1. At a time, cuckoo chooses a nest randomly to lay an egg.
2. The best nest is the nest with high-quality eggs, and it will carry over to the next generation.
3. The number of available host nests is fixed. The probability of discovering an intruding egg by the host bird is $P_a \in [0, 1]$. 

4. If the host bird discovers the intruding egg, it throws it away from the nest or abandon the nest and starts to build a new nest elsewhere.

4.4. Cuckoo search algorithm. The main steps of the standard cuckoo search algorithm are presented in Algorithm 2. These steps are summarized as follows.

The standard cuckoo search algorithm starts by setting the initial values of the main parameters such as the initial values of population size $P$, probability $P_a \in [0, 1]$, the maximum number of iterations MGN and the initial iteration counter $t$ is setting. The initial population $P$ is generated randomly, and each solution $x_i$ in the population is evaluated by calculating its fitness function $f(x_i)$. The following steps are repeated until the termination criterion satisfied. A new solution is generated randomly using a Lévy flight as follows.

$$x_i^{(t+1)} = x_i^t + \alpha \odot \text{Levy}(\lambda)$$

Where $\odot$ denotes entry-wise multiplication, $\alpha$ is the step size, and Lévy $(\lambda)$ is the Lévy distribution.

The new solution is replaced with a randomly selected solution if its objective function is better than the objective function of the selected random solution. A fraction $P_a$ is generated randomly to abandon and replace the worst solutions by generating new solutions using local random walks as follows.

$$x_i^{(t+1)} = x_i^t + \gamma (x_j^t - x_k^t)$$

Where $x_j^t$ and $x_k^t$ are two different solutions selected randomly and $\gamma$ is a random number.

The solutions are ranked according to their objective values, and the best solution is assigned and the iteration counter increases. The operation is repeated until the termination criteria are satisfied. Finally, the best-found solution is produced.

5. The proposed HFFACS algorithm. In the section, the proposed algorithm is presented as shown in Algorithm 3. The proposed algorithm is summarized as follows.

• Initialization. The proposed algorithm starts by setting all initial values of its parameters such as the randomization parameter $\alpha$, firefly attractiveness $\beta_0$, media light absorption coefficient $\gamma$, population size $P$, probability $P_a \in [0, 1]$ and maximum generation number MGN.

• Solution evaluation. The initial population is generated randomly, and each solution in the population is evaluated by calculating its objective function $f(x_i)$.

• Solution update. The following steps are repeated until the termination criterion satisfied which is to reach the desired number of iterations MGN. For each $x_i$ and $x_j$, $i = 1, \ldots, P$ and $j = 1, \ldots, i$. If the objective function of solution $j$ is better than the objective function of solution $i$. The solution $i$ will move towards the solution $j$ as shown in Equation 7. The attractiveness varies with distance via $\exp(-\gamma r^2)$ as shown in Equation 4.

• Diversification process. The main drawback of the firefly algorithm is the premature convergence and stagnation after the number of iterations. In order to avoid this drawback, we combine the cuckoo search with a fraction $P_a$ which is generated randomly to abandon and replace the worst solutions by
Algorithm 2 Cuckoo search algorithm

1: Set the initial value of the host nest size $P$, probability $p_a \in [0, 1]$ and maximum
number of iterations $Max_{itr}$.
2: Set $t := 0$. \{Counter initialization\}.
3: for $(i = 1 : i \leq P)$ do
4: Generate initial population of $P$ host $x_i^{(t)}$. \{n is the population size\}.
5: Evaluate the fitness function $f(x_i^{(t)})$.
6: end for
7: repeat
8: Generate a new solution (Cuckoo) $x_i^{(t+1)}$ randomly by Lévy flight.
9: Evaluate the fitness function of a solution $x_i^{(t+1)}$ if $f(x_i^{(t+1)}) > f(x_j^{(t)})$ then
10: Replace the solution $x_j$ with the solution $x_i^{(t+1)}$
11: end if
12: Abandon a fraction $p_a$ of worse nests.
13: Build new nests at new locations using Lévy flight a fraction $p_a$ of worse nests
14: Keep the best solutions (nests with quality solutions)
15: Rank the solutions and find the current best solution
16: Set $t = t + 1$. \{Iteration counter increasing\}.
17: until $(t > MGN)$. \{Termination criteria satisfied\}.
18: Produce the best solution.

generating new solutions using a Lévy flight as shown in Equation 8. The Lévy flight operator has a strong ability of the exploration process and can help the proposed algorithm to find the global minimum of the tested functions in a reasonable time.

- **termination criterion.** The proposed algorithm is terminated when it reaches to the maximum number of iterations MGN. Finally, the best solution is obtained.

5.1. **Constraints handling.** In this paper, a non-stationary penalty function has been used, where the values of the penalty function are dynamically changed during the search process. A general form of the penalty function is defined as follows:

$$ F(x) = f(x) + h(k)H(x), x \in S \subset R^n $$  \hspace{1cm} (10)

Where $f(x)$ is the objective function, $h(k)$ is a nonstationary (dynamically modified) penalty function, $k$ is the current iteration number, and $H(x)$ is a penalty factor, which is calculated as follows.

$$ H(x) = \sum_{i=1}^{m} \theta(q_i(x))q_i(x)\gamma(q_i(x)) $$  \hspace{1cm} (11)

Where $q_i(x) = \max(0, g_i(x))$, $i = 1, \ldots, m$, $g_i$ are the constrains of the problem, $q_i$ is a relative violated function of the constraints, $\theta(q_i(x))$ is the power of the penalty function, the values of the functions $h(\cdot)$, $\theta(\cdot)$ and $\gamma(\cdot)$ are problem
Algorithm 3 The proposed HFFACS algorithm

1: Set the initial values of the randomization parameter $\alpha$, firefly attractiveness $\beta_0$, media light absorption coefficient $\gamma$, population size $P$ and maximum generation number $MGN$.
2: Generate the initial population $x_i$ randomly, $i = \{1, \ldots, P\}$ \{Initialization\}
3: Evaluate the fitness function $f(x_i)$ of all solutions in the population
4: Light intensity $I_i$ at $x_i$ is determined by $f(x_i)$
5: Set $t = 0$
6: repeat
7: for $(i = 1; i < P; i++)$ do
8: for $(j = 1; j < i; j++)$ do
9: if $I_i(t+1) < I_j(t+1)$ then
10: Move solution $i$ towards $j$ as shown in Equation 7
11: end if
12: Calculate the attractiveness parameter $\beta$, where $\beta(r) = \beta_0 e^{-\gamma r^2}$
13: Evaluate the fitness function $f(x_i)$ of all solutions in the population
14: Update all the solutions in the population
15: end for
16: end for
17: Abandon a fraction $p_a$ of worse nests.
18: Build new nests at new locations using Lévy flight a fraction $p_a$ of worse nests as shown in Equation 8
19: Keep the best solutions (nests with quality solutions)
20: Rank the solutions and find the current best solution
21: Set $t = t + 1$. \{Iteration counter increasing\}
22: until $(t > MGN)$. \{Termination criteria satisfied\}
23: Produce the optimal solution.

dependent. The following values are used for the penalty function:

$$
\gamma(q_i(x)) = \begin{cases} 
1 & \text{if } q_i(x) < 1 \\
2 & \text{Otherwise}
\end{cases}
$$

Where the assignment function is

$$
\theta(q_i(x)) = \begin{cases} 
10 & \text{if } q_i(x) < 0.001 \\
20 & \text{if } 0.001 \leq q_i(x) < 0.1 \\
30 & \text{if } 0.1 < q_i(x) < 1 \\
300 & \text{otherwise}
\end{cases}
$$

and the penalty value $h(t) = t \ast \sqrt{t}$

6. **Numerical experiments.** In order to investigate the efficiency of the proposed HFFACS, the general performance of the HFFACS is presented on different benchmark functions and compared the results of the proposed algorithm with variant particle swarm optimization algorithms. HFFACS was programmed in MATLAB, the results of the comparative algorithms are taken from their original papers. In the following subsections, the parameters setting of the proposed algorithm and the optimal values of the applied test functions are reported in Table 2. Also, the
performance analysis of the proposed algorithm is presented with the comparative results between it and the other algorithms.

6.1. **Parameter setting.** The parameters of the HFFACS algorithm are summarized with their assigned values in Table 1. These values are based on the common setting in the literature or determined through our preliminary numerical experiments.

6.2. **Test constrained optimization problems.** The HFFACS algorithm, is tested on 6 constrained optimization problems $f_1 - f_6$, which are reported in Table 2. The optimal values of these benchmark problems are reported in Table 3 and the functions with their definitions are reported as follows.

6.3. **The general performance of the proposed HFFACS.** In order to investigate the general performance of the proposed algorithm with the constrained optimization problems, the best, worst, mean standard deviation (st.d) and the success rate (succ) of function evaluations are reported in Table 4 over 30 runs. The run is considered successes if the algorithm reached the global minimum for each function within 100,000 function evaluation value.

   The general performance of the functions $f_1 - f_6$ is shown in Figure 1 by plotting the number of iterations versus the function values after 100 iterations. Figure 1 shows that the function values rapidly decrease as the number of iterations increases. It can be concluded from Table 4 and Figure 1, that the proposed algorithm is a promising algorithm and can obtain the optimal or near optimal solution within few thousands of function evaluation values.

6.4. **Comparison between the Cuckoo search, Firefly algorithms, and the proposed HFFACS.** In order to investigate the efficiency of the proposed algorithm, we compare it against the standard cuckoo search (CS) and the firefly (FA) algorithms. The combination between the cuckoo search and the firefly algorithms can accelerate the convergence of the proposed HFFACS algorithm. The mean of function evaluation values of the three algorithms is reported in Table 5. The results in Table 5 show that the proposed HFFACS obtain the global minimum of all testes functions faster than the other two standard algorithms. We can conclude from Table 5, that the combination of the standard cuckoo search and the firefly algorithms in the proposed HFFACS can accelerate the convergence of it.

6.5. **HFFACS and other algorithms.** HFFACS is compared with four benchmark algorithms (particle swarm optimization with different variants algorithms) in order to verify the efficiency of it. Before discussing the comparison results of all
Table 2. Constrained test problems

| Problem | \( f(x) \) | Subject to |
|---------|-------------|-----------|
| Problem 1 \[13\] | \( f_1 = (x_1 - 2)^2 + (x_2 - 1)^2 \), | 
| Subject to | \( x_1 = 2x_2 - 1, \quad \frac{x_1}{4} + x_2^2 - 1 \leq 0 \), | with \( x_i \in [-100, 100] \), \( i = 1, 2 \) |
| Problem 2 \[9\] | \( f_2 = (x_1 - 10)^3 + (x_2 - 20)^3 \), | 
| Subject to | \( 100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0 \), | \( (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \), |
| | | \( 13 \leq x_1 \leq 100 \), \( 0 \leq x_2 \leq 100 \) |
| Problem 3 \[14\] | \( f_3 = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^2 + 3(x_4 - 11)^2 + \) | 
| Subject to | \( 10x_1^2 + 7x_2^2 + x_3^2 - 4x_2x_7 - 10x_6 - 8x_7, \) | \( -127 + 2x_1^2 + 3x_2^2 + x_3 + 4x_4^2 + x_5 \leq 0 \), |
| | | \( -282 + 7x_1 + 3x_2 + 10x_4^2 + x_4 - x_5 \leq 0 \), |
| | | \( -196 + 23x_1 + x_3^2 + 6x_4^2 - 8x_7 \leq 0 \), |
| | | \( 4x_1^2 + x_2^2 + 3x_1x_2 + 2x_1 + 5x_6 - 11x_7 \leq 0 \), |
| | | \( -10 \leq x_i \leq 10 \), \( i = 1, \ldots, 7 \). |
| Problem 4 \[14\] | \( f_4 = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \), | 
| Subject to | \( 0 \leq 85.334407 + 0.0056858T_1 + T_2x_1x_4 - 0.0022053x_3x_5 \leq 92 \), | \( 90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0021813x_3^2 \leq 110 \), |
| | | \( 20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25 \), |
| | | \( 78 \leq x_1 \leq 102, \quad 33 \leq x_2 \leq 45, \quad 27 \leq x_i \leq 45, \quad i = 3, 4, 5, \) |
| | | Where \( T_1 = x_2x_5 \) and \( T_5 = 0.0006262 \) |
| Problem 5 \[14\] | \( f_5 = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \), | 
| Subject to | \( 0 \leq 85.334407 + 0.0056858T_1 + T_2x_1x_4 - 0.0022053x_3x_5 \leq 92 \), | \( 90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0021813x_3^2 \leq 110 \), |
| | | \( 20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25 \), |
| | | \( 78 \leq x_1 \leq 102, \quad 33 \leq x_2 \leq 45, \quad 27 \leq x_i \leq 45, \quad i = 3, 4, 5, \) |
| | | Where \( T_1 = x_2x_3 \) and \( T_5 = 0.00026 \) |
| Problem 6 \[25\] | \( f_6 = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 - 10x_6 - 0.5(\sum\limits_{i=1}^{5} x_i^2) \), | 
| Subject to | \( 6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 - 6.5 \leq 0 \), | \( 10x_1 + 10x_3 + x_6 \leq 20 \), |
| | | \( 0 \leq x_i \leq 1, \quad i = 1, \ldots, 5, \) |
| | | \( 70 \leq x_6 \leq 50 \) |

Table 3. Function properties

| Problem | \( f \) | Optimal value |
|---------|--------|--------------|
| Problem 1 | \( f_1 \) | 1.3934651 |
| Problem 2 | \( f_2 \) | -6961.81381 |
| Problem 3 | \( f_3 \) | 680.630057 |
| Problem 4 | \( f_4 \) | -30665.538 |
| Problem 5 | \( f_5 \) | Unknown |
| Problem 6 | \( f_6 \) | -213 |
Table 4. The function evaluation of the proposed HFFACS algorithm for functions $f_1 - f_6$

| $f$ | Best | Worst | Mean  | St.d  | Succ |
|-----|------|-------|-------|-------|------|
| $f_1$ | 9840 | 13,200 | 11,480 | 1681.428 | 30 |
| $f_2$ | 11,520 | 16,200 | 13,800 | 2340.256 | 30 |
| $f_3$ | 40,920 | 43,800 | 42,320 | 1441.66 | 30 |
| $f_4$ | 10,200 | 16,800 | 13,400 | 3304.54 | 30 |
| $f_5$ | 8280  | 10,800 | 9360  | 1297.59 | 30 |
| $f_6$ | 42,000 | 53,280 | 48,360 | 5776.227 | 30 |

Figure 1. The general performance of the HFFACS algorithm with constrained optimization problems

Table 5. The mean function evaluation of the cuckoo search, firefly and the proposed HFFACS algorithms

| $f$ | CS   | FA   | HFFACS |
|-----|------|------|--------|
| $f_1$ | 12,230 | 13,420 | 11,480 |
| $f_2$ | 14,840 | 15,460 | 13,800 |
| $f_3$ | 42,960 | 43,450 | 42,320 |
| $f_4$ | 14,840 | 15,640 | 13,400 |
| $f_5$ | 10,240 | 11,840 | 9360  |
| $f_6$ | 48,840 | 50,450 | 48,360 |
algorithms, a brief description of the comparative four algorithms [33] is given as follows.

- **RWMPSOg.** RWMPSOg is a Random Walk Memetic Particle Swarm Optimization (with global variant), which combines the particle swarm optimization with a random walk with direction exploitation.
- **RWMPSOl.** RWMPSOl is a Random Walk Memetic Particle Swarm Optimization (with local variant), which combines the particle swarm optimization with a random walk with direction exploitation.
- **PSOg.** PSOg is a standard particle swarm optimization with global variant without local search method.
- **PSOl.** PSOl is a standard particle swarm optimization with local variant without local search method.

### 6.6. Comparison between RWMPSOg, RWMPSOl, PSOg, PSOl, and HFFACS

In this subsection, the comparison results between the proposed HFFACS algorithm and the other algorithms is presented in order to verify the efficiency of our proposed algorithm. The five comparative algorithms are tested on six benchmark functions \( f_1 - f_6 \), which are reported in Table 2. The results of the comparative algorithms are reported from their original papers [33]. The average (mean), standard deviation maximum (St.d), and success rate (\% Suc) of the function values are reported over 30 runs in Table 6. The run is considered successes if the algorithm reached the global minimum of the solution within 100,000 function evaluation value. The best results of the comparative algorithms are reported with **boldface text.** The results in Table 6 show that the proposed HFFACS algorithm successes in all runs and obtains the desired objective value of each function faster than the other algorithms.

### 7. Engineering optimization problems

In order to verify the efficiency of the proposed algorithm, we test it on other five real-world engineering optimization problems. These problems are presented in the following subsections as follows.

#### 7.1. Welded beam design problem

The first problem is a welded beam design problem which aims to minimize the cost of the welded beam structure. The problem consists of a beam \( A \) and the required weld to hold it to member \( B \) subject to constraints on shear stress \( \tau \), beam bending stress \( \sigma \), buckling load on the bar \( P_c \), beam end deflection \( \delta \). The four design variables of the problem \( h(x_1), l(x_2), t(x_3) \) and \( b(x_4) \) are shown in Figure 2.

Minimize \( f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \)

Subject to

\[
\begin{align*}
g_1(x) &= \tau(x) - \tau_{\text{max}} \leq 0 \\
g_2(x) &= \sigma(x) - \sigma_{\text{max}} \leq 0 \\
g_3(x) &= x_1 - x_4 \leq 0 \\
g_4(x) &= 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 0.5 \leq 0 \\
g_5(x) &= 0.125 - x_1 \leq 0 \\
g_6(x) &= \delta(x) - \delta_{\text{max}} \leq 0 \\
g_7(x) &= P - P_c(x) \leq 0
\end{align*}
\]
Table 6. Comparison between RWMPSO_g, RWMPSO_l, PSO_g, PSO_l, and HFFACS for functions $f_1 - f_6$

| $f$ | Mean  | Std   | Succ |
|-----|-------|-------|------|
| $f_1$ | RWMPSO_g  | 1.832 | 0.474 | 25   |
|      | RWMPSO_l  | 1.427 | 0.061 | 30   |
|      | PSO_g     | 2.042 | 0.865 | 24   |
|      | PSO_l     | 1.454 | 0.078 | 30   |
|      | HFFACS    | **1.393** | 1.234e-04 | 30 |
| $f_2$ | RWMPSO_g  | -6961.283 | 0.380 | 30   |
|      | RWMPSO_l  | -6960.717 | 1.798 | 30   |
|      | PSO_g     | -6960.668 | 1.043 | 24   |
|      | PSO_l     | -6939.627 | 58.789 | 22 |
|      | HFFACS    | **-6961.813** | 3.818e-05 | 30 |
| $f_3$ | RWMPSO_g  | 680.915 | 0.178 | 30   |
|      | RWMPSO_l  | 680.784 | 0.062 | 30   |
|      | PSO_g     | 681.254 | 0.245 | 30   |
|      | PSO_l     | 680.825 | 0.077 | 30   |
|      | HFFACS    | **680.6301** | 3.464e-08 | 30 |
| $f_4$ | RWMPSO_g  | -30665.550 | 0.000 | 30   |
|      | RWMPSO_l  | -30665.550 | 0.000 | 30   |
|      | PSO_g     | -30665.550 | 0.000 | 30   |
|      | PSO_l     | -30665.550 | 0.000 | 30   |
|      | HFFACS    | **-30665.518** | 2.545e-07 | 30 |
| $f_5$ | RWMPSO_g  | -31021.173 | 11.506 | 30   |
|      | RWMPSO_l  | -31026.435 | 0.000 | 30   |
|      | PSO_g     | -31021.140 | 12.617 | 30   |
|      | PSO_l     | -31026.440 | 0.000 | 30   |
|      | HFFACS    | **-31026.427** | 1.767e-06 | 30 |
| $f_6$ | RWMPSO_g  | -212.616 | 1.043 | 30   |
|      | RWMPSO_l  | -212.047 | 0.002 | 30   |
|      | PSO_g     | -211.833 | 1.840 | 30   |
|      | PSO_l     | -212.933 | 0.365 | 30   |
|      | HFFACS    | **-212.962** | 0.002 | 30   |

Where the other parameters are defined as follows.

$$
\tau(x) = \sqrt{((\tau')^2 + (\tau'')^2 + \frac{2\tau'\tau''x_2}{2R}}}, \quad \tau' = \frac{p}{\sqrt{2x_1x_2}}
$$

$$
\tau'' = \frac{MR}{J}, \quad M = P(L + \frac{x_2}{2}), \quad R = \sqrt{\left(\frac{x_1 + x_3}{2}\right)^2 + \frac{x_2^2}{4}}
$$

$$
J = 2\left\{\frac{x_1x_2}{\sqrt{2}} + \left(\frac{x_1 + x_3}{2}\right)^2\right\}, \quad \sigma(x) = \frac{6PL}{x_4x_5^3}
$$

$$
\delta(x) = \frac{4PL^3}{E x_4x_5^3}, \quad P_e(x) = \frac{4.013\sqrt{EGx_4^2x_5^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\frac{\sqrt{E}}{4G}\right) \quad (12)
$$
Figure 2. Welded beam design problem

Figure 3. Pressure vessel design problem

Where $P = 6000$lb, $L = 14$, $\delta_{\text{max}} = 0.25$ in, $E = 30,106$ psi, $G = 12,106$ psi, $\tau_{\text{max}} = 13,600$psi, $\sigma_{\text{max}} = 30,000$ psi and $0.1 \leq x_i \leq 10.0$ ($i = 1, 2, 3, 4$).

7.2. Pressure vessel design problem. The second problem is a cylindrical vessel. The objective of this problem is minimizing the total cost of material, forming and welding. The problem has four design variables which are shown in Figure 3, $T_s$ ($x_1$, thickness of the shell), $T_h$ ($x_2$, thickness of the head), $R$ ($x_3$, inner radius) and $L$ ($x_4$, length of the cylindrical section of the vessel without the head). $T_s$ and $T_h$ are integer multiples of 0.0625 in, $R$ and $L$ are continuous variables.

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to

$$g_1(x) = -x_1 + 0.0193x_3$$
$$g_2(x) = -x_2 + 0.00954x_3 \leq 0$$
$$g_3(x) = -\pi x_2^2 x_4 - 4/3\pi x_3^3 + 1296000 \leq 0$$
$$g_4(x) = x_4 - 240 \leq 0$$

Where $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10 \leq x_3 \leq 200$ and $10 \leq x_4 \leq 200$.

7.3. Speed reducer design problem. The third problem is a speed reducer problem which aim to minimize a gear box volume and weight subject to several constrains. The speed reducer is shown in Figure 4. The problem has seven design variables $x_1 - x_7$, which describe as follow. $x_1$ is a width of the gear face (cm), $x_2$ teeth module (cm), $x_3$ number of pinion teeth, $x_4$ shaft 1 length between bearings...
(cm), $x_5$ shaft 2 length between bearing (cm), $x_6$ diameter of shaft 1 (cm) and $x_7$ diameter of shaft 2 (cm).

$$f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934)$$
$$-1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Subject to

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$
$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^3} - 1 \leq 0$$
$$g_3(x) = \frac{1.93x_4^2}{x_2x_6^2x_3} - 1 \leq 0$$
$$g_4(x) = \frac{1.93x_5^2}{x_2x_7^2x_3} - 1 \leq 0$$
$$g_5(x) = \frac{[(745x_4/x_2x_3)^2 + 16.9 \times 10^6]^{0.5}}{110.0x_6^3} - 1 \leq 0$$
$$g_6(x) = \frac{[(745x_5/x_2x_3)^2 + 157.5.9 \times 10^6]^{0.5}}{85.0x_7^3} - 1 \leq 0$$
$$g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0$$
$$g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0$$
$$g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0$$
$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_1} - 1 \leq 0$$
$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$ (13)
7.4. **Three-bar truss design problem.** The fourth problem is the three-bar truss design problem. The objective of this problem is to minimize the weight $f(x)$ of three bar truss subject to some constraints as follow.

$$f(x) = (2\sqrt{2x_1 + x_2}) \times l$$

Subject to

$$g_1(x) = \frac{\sqrt{2x_1} + x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0$$

$$g_2(x) = \frac{x_2}{\sqrt{2x_1^2 + 2x_1x_2}} P - \sigma \leq 0$$

$$g_3(x) = \frac{1}{\sqrt{2x_2 + x_1}} P - \sigma \leq 0$$

Where $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$, $l = 100$ cm, $P = 2kN/cm^2$, and $\sigma = 2kN/cm^2$. The minimum weighted structure should be achieved by determining the optimal cross-sectional areas $x_1$ and $x_2$.

7.5. **Spring design problem.** The last problem is a spring design problem. The objective of the problem is to minimize the weight $f(x)$ of it subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variable. The mean coil diameter $D(x_2)$, the wire diameter $d(x_1)$ and the number of active coils $P(x_3)$ are the design variables as shown in Figure 5.

$$f(x) = (x_3 + 2)x_2x_1^2$$

Subject to

$$g_1(x) = 1 - \frac{x_2^2x_3}{71785x_1} \leq 0$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^2 - x_1^4)} + \frac{1}{5108x_1} - 1 \leq 0$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

Where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$ and $2 \leq x_3 \leq 15$.

7.6. **The general performance of the proposed HFFACS algorithm with engineering optimization problems.** The convergence curves of the performance of the proposed algorithm with the five engineering optimization problems
AN EFFECTIVE HYBRID FIREFLY ALGORITHM WITH ...

Figure 6. The general performance of the proposed HFFACS algorithm with engineering optimization problems

Table 7. The general performance (function evaluations) of HFFACS with engineering optimization problems

| Design problem     | Best  | Mean  | Worst | Std   |
|--------------------|-------|-------|-------|-------|
| Welded beam design | 21,000| 22,253.4| 23,760| 1397.31|
| Pressure vessel design | 39,360 | 40,120 | 40,800 | 723.32|
| Speed reducer design | 87,840 | 90,200 | 92,640 | 2401 |
| Three-bar truss design | 6120 | 7440 | 8400 | 2401 |
| Spring design       | 36,120 | 38,560 | 40,080 | 2134.291|

Table 8. The general performance (function values) of HFFACS with engineering optimization problems

| Design problem     | Best      | Mean      | Worst     | Std   |
|--------------------|-----------|-----------|-----------|-------|
| Welded beam design | 2.380957 | 2.380957 | 2.380957 | 1.82e-8|
| Pressure vessel design | 6059.71433 | 6059.71433 | 6059.71433 | 4.54e-12|
| Speed reducer design | 2.994471 | 2.994471 | 2.994471 | 0.00 |
| Three-bar truss design | 263.895843 | 263.895843 | 263.895843 | 3.54e-11|
| Spring design      | 0.012665 | 0.012665 | 0.012665 | 5.77e-12|

are shown in Figure 6. In Figure 6, we plot the number of iterations versus the number of function values. The best mean worst and standard deviation of the function evaluations and function values are reported over 30 runs in Tables 7 and 8 respectively. We can conclude from Figure 6, Tables 7 and 8 that the proposed algorithm can reach to the global minimum of the tested functions within a few number of iterations.
8. Conclusion and future work. A new hybrid nature-inspired algorithm is presented in this paper to solve constrained and real-world engineering optimization problems. The proposed algorithm is called Hybrid Firefly Algorithm and Cuckoo Search (HFFACS) algorithm. Invoking cuckoo search algorithm with a Lévy flight improves the performance of the firefly algorithm and helps it to avoid premature convergence and to trap in local minima. In order to examine the general performance of the HFFACS algorithm, it tested on six benchmarks constrained and five real-world engineering problems and compared against four algorithms. The experimental results show the HFFACS performance is faster than the other four algorithms and can obtain the optimal or near optimal solution within a reasonable time. Also, we would like to apply our proposed algorithm on solving unconstrained optimization problems [1], large scale problems and molecular potential energy function [42], [41], team formation problem [8], and minimax and integer programming problems [2], [39], [40]. Furthermore, we would like to use binary version to solve feature selection problems [37], [38].

REFERENCES

[1] A. F. Ali and M. A. Tawhid, Hybrid particle swarm optimization with a modified arithmetical crossover for solving unconstrained optimization problems, INFOR: Information Systems and Operational Research, 53 (2015), 125–141.
[2] A. F. Ali and M. A. Tawhid, Hybrid simulated annealing and pattern search algorithm for solving integer programming and minimax problems, Pacific Journal of Optimization, 12 (2016), 151–184.
[3] A. F. Ali and M. A. Tawhid, A hybrid PSO and DE algorithm for solving engineering optimization problems, Applied Mathematics & Information Sciences, 10 (2016), 431–449.
[4] C. Blum, J. Puchinger, G. R. Raidl and A. Roli, Hybrid metaheuristics in combinatorial optimization: A survey, Applied Soft Computing, 11 (2011), 4135–4151.
[5] C. Brown, L. S. Liebovitch and R. Glendon, Lévy lights in dobe juhoansi foraging patterns, Human Ecol., 35 (2007), 129–138.
[6] S. A. Claus, P. W. Tsai and J. S. Pan, Cat swarm optimization, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4099 (2006), 854–858.
[7] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.
[8] W. H. El-Ashmawi, A. F. Ali and M. A. Tawhid, An improved particle swarm optimization with a new swap operator for team formation problem, Journal of Industrial Engineering International, 2018, 1–19.
[9] C. A. Floudas and P. M. Pardalos, A Collection of Test Problems for Constrained Global Optimization Algorithms, In P. M. Floudas, Lecture notes in computer science, Vol.455. Berlin, 1990.
[10] M. Gen and L. Lin, Multiobjective evolutionary algorithm for manufacturing scheduling problems, state of the art survey, Journal of Intelligent Manufacturing, 25 (2014), 849–866.
[11] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research, 13 (1986), 533–549.
[12] F. Glover, A template for scatter search and path relinking, Lecture Notes on Computer Science, 1997, 1354–1363.
[13] D. M. Himmelblau, Applied Nonlinear Programming, New York: McGraw-Hill, 1972.
[14] W. Hock and K. Schittkowski, Test Examples for Nonlinear Programming Codes, In Lecture notes in economics and mathematical systems (Vol.187). Berlin, Springer, 1981.
[15] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[16] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony algorithm, Journal of Global Optimization, 39 (2007), 459–471.
[17] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks, 4 (1995), 1942–1948.
[18] S. Kirkpatrick, C. Gelatt and M. Vecchi, Optimization by simulated annealing, Science, 220 (1983), 671–680.

[19] X. L. Li, Z. J. Shao and J. X. Qian, Optimizing method based on autonomous animates Fish-swarm algorithm, Xitong Gongcheng Lilun yu Shijian-System Engineering Theory and Practice, 22 (2002), 32.

[20] C. Liang, Y. Huang and Y. Yang, A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning, Comput. Ind. Eng., 56 (2009), 1021–1028.

[21] L. Lin, M. Gen and X. Wang, Integrated multistage logistics network design by using hybrid evolutionary algorithm, Comput. Ind. Eng., 56 (2009), 854–873.

[22] M. Lozano and C. Garcia-Martinez, Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification, Overview and Progress Report. Comput. Oper. Res., 37 (2010), 481–497.

[23] S. Lukasik and S. Zak, Firefly Algorithm for Continuous Constrained Optimization Tasks, in Proceedings of the International Conference on Computer and Computational Intelligence (ICCCI 09), N.T. Nguyen, R. Kowalczyk, and S.-M. Chen, Eds., vol. 5796 of LNAI, 97–106, Springer, Wroclaw, Poland, October 2009.

[24] O. Makeyev, E. Sazonov, M. Moklyachuk and P. Logez-Meye, Hybrid evolutionary algorithm for microscrew thread parameter estimation, Eng. Appl. Artif. Intell., 23 (2010), 446–452.

[25] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1994.

[26] N. Mladenovic, A variable Neighborhood Algorithm a New Meta-Heuristic for Combinatorial Optimization, Abstracts of Papers Presented at Optimization Days, Montréal, Canada, p. 112, 1995.

[27] M. Mladenovic and P. Hansen, Variable neighborhood search, Computers and Operations Research, 24 (1997), 1097–1100.

[28] D. Mongus, B. Repnik, M. Mernik and B. Zalik, A hybrid evolutionary algorithm for tuning a cloth-simulation model, Appl. Soft Comput., 12 (2012), 266–273.

[29] F. Neri and C. Cotta, Memetic algorithms and memetic computing optimization, A literature review. Swarm and Evolutionary Computation, 2 (2012), 1–14.

[30] T. Niknam and E. A. Farsani, A hybrid self-adaptive particle swarm optimization and modified shuffled frog leaping algorithm for distribution feeder reconfiguration, Eng. Appl. Artif. Intell., 23 (2010), 1340–1349.

[31] M. K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, Control Systems, IEEE, 22 (2002), 52–67.

[32] R. B. Payne, M. D. Sorenson and K. Klitz, The Cuckoos, Oxford University Press, 2005.

[33] Y. G. Petalas, K. E. Parsopoulos and M. N. Vrahatis, Memetic particle swarm optimization, Ann oper Res, 156 (2007), 99–127.

[34] C. Prodhon, A hybrid evolutionary algorithm for the periodic location-routing problem, Eur. J. Oper. Res., 210 (2011), 204–212.

[35] T. Stützle, Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications, Ph.D. Thesis, Darmstadt University of Technology, 1998.

[36] R. Tang, S. Fong, X. S. Yang and S. Deb, Wolf search algorithm with ephemeral memory, In Digital Information Management (ICDIM), 2012 Seventh International Conference on Digital Information Management, (2012), 165–172.

[37] M. A. Tawhid and K. B. Dsouza, Hybrid Binary Bat Enhanced Particle Swarm Optimization Algorithm for solving feature selection problems, Applied Computing and Informatics, 2018.

[38] M. A. Tawhid and K. B. Dsouza, Hybrid binary dragonfly enhanced particle swarm optimization algorithm for solving feature selection problems, Mathematical Foundations of Computing, 1 (2018), 181–200.

[39] M. A. Tawhid and A. F. Ali, Direct search firefly algorithm for solving global optimization problems, Applied Mathematics & Information Sciences, 10 (2016), 841–860.

[40] M. A. Tawhid and A. F. Ali, A simplex grey wolf optimizer for solving integer programming and minimax problems, Numerical Algebra, Control & Optimization, 7 (2017), 301–323.

[41] M. A. Tawhid and A. F. Ali, A hybrid social spider optimization and genetic algorithm for minimizing molecular potential energy function, Soft Computing, 21 (2017), 6499–6514.

[42] M. A. Tawhid and A. F. Ali, A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function, Memetic Computing, 9 (2017), 347–359.
[43] Y. Wang, Z. Cai, Y. Zhou and Z. Fan, Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique, *Struct. Multidiscip. Optimiz.*, **37** (2009), 395–413.

[44] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK, 2008.

[45] X. S. Yang and S. Deb, Cuckoo search via Lévy flights, In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009*, World Congress on, IEEE, 2009, 210–214.

[46] X. S. Yang, Firefly algorithm, stochastic test functions and design optimization, *International Journal of Bio-Inspired Computation*, **2** (2010), 78–84.

Received August 2018; revised October 2018.

*E-mail address: mtawhid@tru.ca*

*E-mail address: ahmed.fouad@sci.suez.edu.eg*