Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100304

Direct Search Firefly Algorithm for Solving Global Optimization Problems

Mohamed A. Tawhid [1,2,*] *and Ahmed F. Ali*[3,4]

[1] Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC, V2C 0C8, Canada
[2] Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Moharam Bey 21511, Alexandria, Egypt
[3] Department of Computer Science, Faculty of Computers & Informatics, Suez Canal University, Ismailia, Egypt
[4] Department of Mathematics and Statistics, Faculty of Science, Thompson Rivers University, Kamloops, BC, V2C 0C8, Canada

Received: 5 Oct. 2015, Revised: 5 Jan. 2016, Accepted: 6 Jan. 2016
Published online: 1 May 2016

Abstract: In this paper, we propose a new hybrid algorithm for solving global optimization problems, namely, integer programming and minimax problems. The main idea of the proposed algorithm, Direct Search Firefly Algorithm (DSFFA), is to combine the firefly algorithm with direct search methods such as pattern search and Nelder-Mead methods. In the proposed algorithm, we try to balance between the global exploration process and the local exploitation process. The firefly algorithm has a good ability to make a wide exploration process while the pattern search can increase the exploitation capability of the proposed algorithm. In the final stage of the proposed algorithm, we apply a final intensification process by applying the Nelder-Mead method on the best solution found so far, in order to accelerate the search instead of letting the algorithm running with more iterations without any improvement of the results. Moreover, we investigate the general performance of the DSFFA algorithm on 7 integer programming problems and 10 minimax problems, and compare it against 5 benchmark algorithms for solving integer programming problems and 4 benchmark algorithms for solving minimax problems. Furthermore, the experimental results indicate that DSFFA is a promising algorithm and outperforms the other algorithms in most cases.

Keywords: Firefly algorithm, Direct search methods, pattern search method, Nelder-Mead method, integer programming problems, Minimax problems

1 Introduction

Our goal of this paper is to solve minimax and integer programming problems via a metaheuristic algorithm.

Metaheuristic algorithms have been applied to solve many NP-hard optimization problems. Recently, there are new metaheuristic algorithms which are inspired from the behaviour of a group of social organisms. These algorithms are called nature inspired algorithm or swarm intelligence algorithms, such as Ant Colony Optimization (ACO) [13], Artificial Bee Colony (ABC) [25], Particle Swarm Optimization (PSO) [26], Bacterial foraging [38], Bat algorithm (BA) [54], Bee Colony Optimization (BCO) [46], Wolf search [45], Cat swarm [11], Cuckoo search [53], Firefly algorithm (FA) [51], [53], Fish swarm/school [29], etc.

Firefly algorithm (FA) is one of the most promising swarm intelligence algorithm inspired by the flashing behaviour of fireflies [51]. Due to the powerful of firefly algorithm, many researchers have applied it to solve various applications, for example, Horng et al. [19], [20] applied FA for digital image compression and demonstrated that FA used least computation time. In [5], Banati and Bajaj used FA for feature selection and showed that firefly algorithm produced consistent and better performance in terms of time and optimality than other algorithms. In [15] and Azad [2], the authors used FA to solve engineering design problems. Basu and Mahanti [7] as well as Chatterjee et al. [10] applied FA for antenna design optimization. Sayadi et al. [43] developed a discrete version of FA which can efficiently solve NP-hard scheduling problems, also in [1], [53], [55], the authors used FA efficiently to solve

* Corresponding author e-mail: Mtawhid@tru.ca

multi-objective load dispatch problems. Furthermore, in [37], [43], [56], FA have been applied for scheduling and traveling salesman problem in a promising way.

The above-mentioned algorithms have been widely used to solve unconstrained and constrained problems and their applications. However these algorithms have been applied in a few works to solve minimax and integer programming problems, although the variety of many real life applications for these two problems such as warehouse location problem, VLSI (very large scale integration) circuits design problems, robot path planning problems, scheduling problem, game theory, engineering design problems, [12], [35], [57].

An integer programming problem is a mathematical optimization problem in which all of the variables are restricted to be integers. The unconstrained integer programming problem can be defined as follow.

$$min f(x), \;\; x \in S \subseteq \mathbb{Z}^n, \tag{1}$$

where $\mathbb{Z}$ is the set of integer variables, S is a not necessarily bounded set.

One of the most famous exact integer programming algorithms is Branch and Bound (BB). However it suffers from high complexity, since it explores a hundred of nodes in a big tree structure when we solve a large scale problems. Recently, there are some efforts to apply some of swarm intelligence algorithms to solve integer programming problems such as ant colony algorithm [23], [24], artificial bee colony algorithm [3], [47], particle swarm optimization algorithm [39], cuckoo search algorithm [48] and firefly algorithm[4].

We consider another optimization problem in this paper, namely, minimax problem. The general form of the minimax problem [50] can be defined as

$$\min \; F(x) \tag{2}$$

where

$$F(x) = \max \;\; f_i(x), \quad i = 1, \ldots, m \tag{3}$$

with $f_i(x) : S \subset \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$.
The nonlinear programming problems, with inequality constraints, of the form

$$\begin{aligned} &\min \;\; F(x), \\ &g_i(x) \geq 0, \quad i = 2, \ldots, m, \end{aligned}$$

can be transformed into the following minimax problem

$$\min \; \max \;\; f_i(x), \quad i = 1, \ldots, m \tag{4}$$

where

$$\begin{aligned} &f_1(x) = F(x), \\ &f_i(x) = F(x) - \alpha_i g_i(x), \\ &\alpha_i > 0, \; ; \quad i = 2, \ldots, m. \end{aligned} \tag{5}$$

It has been proved that for sufficiently large α_i, the optimum point of the minimax problem, coincides with the optimum point of the nonlinear programming problem [6].

One of the common gradient based approaches for solving minimax problems is Sequential Quadratic Programming (SQP). Starting from an initial approximation of the solution, a Quadratic Programming (QP) problem is solved at each iteration of the SQP method, yielding a direction in the search space.

There are other algorithms based on a smooth techniques have been applied for solving minimax problems. These techniques are solving a sequence of smooth problems, which approximate the minimax problems in the limit [30], [40], [50]. The algorithms based in theses techniques aim to generate a sequence of approximations, which converges to Kuhn-Tucker point of the minimax problem, for a decreasing sequence of positive smoothing parameters. However, the drawback of theses algorithms is these parameters are small too fast and the smooth problems become significantly ill-conditioned.

Some swarm intelligence algorithms have been applied to solve minimax problems such as PSO [39]. The main drawback of applying swarm intelligence algorithms for solving minimax and integer programming problems is that they are a population based methods which are computationally expensive.

The main objective of this paper is to produce a new hybrid swarm intelligence algorithm by combining the direct search methods with the firefly algorithm in order to solve minimax and integer programming problems [18]. In the proposed algorithm, we try to overcome the expensive computation time of applying other swarm intelligence algorithms. Invoking the pattern search method can accelerate the search, while applying the Nelder-Mead method can avoid running the algorithm more iterations around the optimal solution without any improvements.

Moreover, we investigate the general performance of the proposed FA on well-known benchmark functions and compare its results against different algorithms. We call the proposed algorithm, Direct Search Firefly Algorithm (DSFFA). In this algorithm, we try to combine the firefly algorithm, with its good capability of exploring the search space, and two of the most promising direct search methods, pattern search and Nelder-Mead methods as local search methods.

We investigate the general performance of the DSFFA algorithm on 7 integer programming problems and 10 minimax problems and compare it against 5 benchmark algorithms for solving integer programming problems and 4 benchmark algorithms for solving minimax problems. The experimental results indicate that DSFFA is a promising algorithm and outperforms the other algorithms in most cases.

The rest of this paper is organized as follows. In Section 2, we highlight the applied direct search methods. In Section 3, we present the standard firefly algorithm and its main components. We describe the proposed algorithm and its main structure in Section 4. In Section 5, we

Table 1: The parameters of the pattern search algorithm.

parameter	definition
Δ_0	Initial mesh size
d	Variable dimension
σ	Reduction factor of mesh size
m	Pattern search repetition number
ε	Tolerance

Algorithm 1 Exploratory search

INPUT: Get the values of x^0, k, Δ_0, d
OUTPUT: New base point x^k

Set $i = 1$
Set $k = 1$
repeat
  Set $x_i^k = x_i^{k-1} + \Delta_{k-1} x_i^{k-1}$
  if $f(x_i^k) < f(x_i^{k-1})$ **then**
    Set $x_i^{k+1} = x_i^k$
  end if
  Set $i = i + 1$
  Set $k = k + 1$
until $i \leq d$

Algorithm 2 The basic pattern search algorithm

INPUT: Get the values of x
OUTPUT: best solution x^*

Set the values of the initial values of the mesh size Δ_0, reduction factor of mesh size σ and termination parameter ε
Set $k = 1$ {**Parameter setting**}
Set the starting base point x^{k-1} {**Initial solution**}
repeat
  Perform exploratory search as shown in Algorithm 1
  if exploratory move success **then**
    Go to 16
  else
    if $\|\Delta_k\| < \varepsilon$, **then**
      Stop the search and the current point is x^*
    else
      Set $\Delta_k = \sigma \Delta_{k-1}$ {**Incremental change reduction**}
      Go to 5
    end if
  end if
  Perform pattern move, where $x_p^{k+1} = x^k + (x^k - x^{k-1})$
  Perform exploratory move with x_p as the base point
  Set x^{k+1} equal to the output result exploratory move
  if $f(x_p^{k+1}) < f(x^k)$ **then**
    Set $x^{k-1} = x^k$
    Set $x^k = x^{k+1}$ {**New base point**}
    Go to 16
  else
    Go to 9 {**The pattern move fails**}
  end if
  Set $k = k + 1$
until $k \leq m$

present the numerical experimental results. Finally, we give the conclusion of the paper in Section 6.

2 Definition of the problems and an overview of the applied algorithms

In this section and its subsections, we give an overview the pattern search method and the Nelder-Mead method.

2.1 Pattern search method

Direct search method is a method for solving optimization problem that dose not require any information about the gradient of the objective function. Pattern search method is one of the most applied direct search methodS to solve global optimization problems. The pattern search method (PS) was proposed by Hook and Jeeves (HJ) [21]. In PS method, there are two type of moves, the exploratory moves and the pattern moves. In the exploratory moves a coordinate search is applied around a selected solution with a step length of Δ as shown in Algorithm 1. If the function value of the new solution is better than the current solution, the exploratory move is successful. Otherwise, the step length is reduced as in (6). If the exploratory move is successful, then the pattern search is applied in order to generate the iterate solution. If the iterate solution is better than the current solution, the exploratory move is applied on the iterate solution and the iterate solution is accepted as a new solution. Otherwise, if the exploratory move is unsuccessful, the pattern move is rejected and the step length Δ is reduced. The operation is repeated until termination criteria are satisfied. The algorithm of HJ pattern search and the main steps of it are presented in Algorithm 2. The parameters in Algorithms 1 and 2 are reported in Table 1.

We can summarize the pattern search algorithm in the following steps.

- **Step 1.** The algorithm starts by setting the initial values of the mesh size Δ_0, reduction factor of mesh size σ and termination parameter ε.
- **Step 2.** Apply exploratory search as shown in algorithm 1 by calculating $f(x^k)$ in order to obtain a new base point
- **Step 3.** If the exploratory move is successful, perform pattern search move, otherwise check the value of the mesh size Δ, if $\Delta < \varepsilon$, where ε is a very small value, stop the search and produces the current solution.
- **Step 4.** If the exploratory move fails and Δ is not less than ε, reduce the mesh size as shown in the following equation

$$\Delta_k = \sigma \Delta_{k-1} \tag{6}$$

- **Step 5.** Apply pattern move by calculating x_p, where $x_p^{k+1} = x^k + (x^k - x^{k-1})$.
- **Step 6.** Set x_p as a new base point and apply exploratory move on it.
- **Step 7.** If the pattern move is successful, repeat the pattern search move on the new point, otherwise the pattern search fails and reduces the mesh size as in (6).
- **Step 8.** The steps are repeated until the termination criteria are satisfied (number of iterations).

2.2 Nelder Mead method

Nelder and Mead in 1965 [34] proposed the Nelder-Mead algorithm (NM). NM algorithm is one of the most popular derivative-free nonlinear optimization algorithms. It starts with $n+1$ points (vertices) $x_1, x_2, \ldots, x_{n+1}$. The vertices are evaluated, ordered and re-labeled in order to assign the best point and the worst point. In minimization problems, the x_1 is considered as the best vertex or point if it has the minimum value of the objective function, while the worst point x_{n+1} with the maximum value of the objective function. At each iteration, new points are computed, along with their function values, to form a new simplex. Four scalar parameters must be specified to define a complete Nelder-Mead algorithm: coefficients of reflection ρ, expansion χ, contraction τ, and shrinkage ϕ. These parameters are chosen to satisfy $\rho > 0$, $\chi > 1$, $0 < \tau < 1$, and $0 < \phi < 1$. The main steps of the Nelder-Mead algorithm are presented as shown below in Algorithm 3. The Nelder-Mead algorithm starts with $n+1$ vertices x_i, $i = 1, \ldots, n+1$, which are evaluated by calculation their fitness function values. The vertices are ordered according to their fitness functions. The reflection process starts by computing the reflected point $x_r = \bar{x} + \rho(\bar{x} - x_{(n+1)})$, where $\bar{x}$ is the average of all points except the worst. If the reflected point x_r is lower than the nth point $f(x_n)$ and greater than the best point $f(x_1)$, then the reflected point is accepted and the iteration is terminated. If the reflected point is better than the best point, then the algorithm starts the expansion process by calculating the expanded point $x_e = \bar{x} + \chi(x_r - \bar{x})$. If x_e is better than the reflected point nth, the expanded point is accepted, Otherwise the reflected point is accepted and the iteration is terminated. If the reflected point x_r is greater than the nth point x_n the algorithm starts a contraction process by applying an outside x_{oc} or inside contraction x_{ic} depending on the comparison between the values of the reflected point x_r and the nth point x_n. If the contracted point x_{oc} or x_{ic} is greater than the reflected point x_r, the shrink process is starting. In the shrink process, the points are evaluated and the new vertices of simplex at the next iteration will be $x'_2, \ldots, x'_{n+1}$, where $x' = x_1 + \phi(x_i - x_1), i = 2, \ldots, n+1$.

In Figure 1, we present an example in order to explain the main steps of the Nelder-Mead algorithm in two dimensions.

- **Step 1.** Given the current solution x, two neighbourhood trial points y_1 and y_2 are generated in a neighbourhood of x as shown in Figure 1 (a).
- **Step 2.** A simplex is constructed in order to find a local trial point as shown in Figure 1 (b).
- **Step 3.** If y_2 is a worst point, we apply the Nelder-Mead algorithm to find a better movement, as shown in Figure 1 (c). If we find a better movement, we refer to this point as a local trial point.

3 Overview of the firefly algorithm

In the following subsection, we will give an overview of the main concepts and structure of the firefly algorithm as follows.

3.1 Main concepts

The firefly algorithm (FA) is a population based metaheuristic algorithm. FA was proposed by Xin-She Yang in late 2007 and 2008 [52], [53]. FA has been inspired from the behaviour of the swarm such as bird folks, insects, fish schooling in nature. According to many recent publications, FA is a promising algorithm and outperforms other metaheuristic algorithms such as genetic algorithm [32], [51], [52], [53]. FA has three flashing characteristics and idealized rules, which are inspired from the real fireflies. We can summarize these rules 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 less brighter firefly 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

In the firefly algorithm, 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

$$I(r) = \frac{I_0}{r^2}, \tag{7}$$

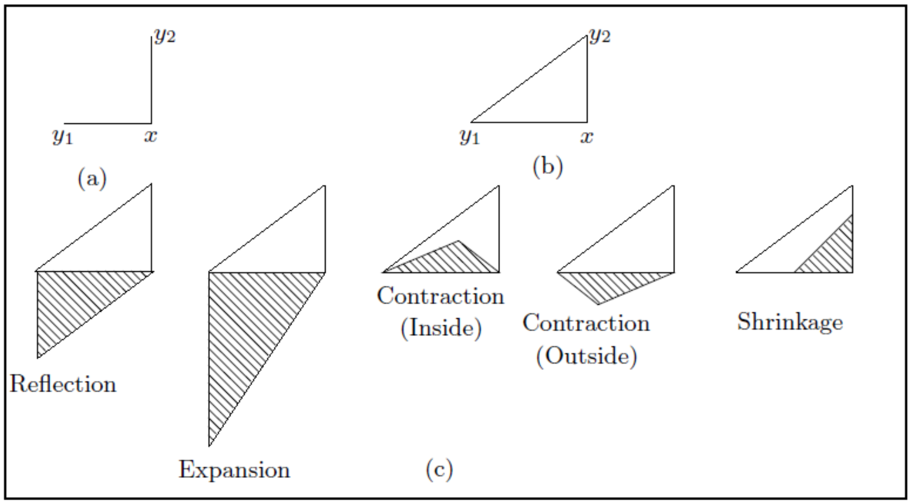

Fig. 1: Nelder-Mead search strategy in two dimensions.

where I_0 is the light intensity at the source, r is the distance between any two fireflies. With the fixed light absorption coefficient γ and in order to avoid singularity at $r = 0$ in the expression in (7). The combined effect of both the inverse square low and absorption can be approximated to Gaussian form, i.e.,

$$I(r) = I_0 e^{-\gamma r^2}, \tag{8}$$

Since a firefly attractiveness is proportional to the light the intensity, the attractiveness function of the firefly can be defined as

$$B(r) = B_0 e^{-\gamma r^2}, \tag{9}$$

where B_0 is the initial attractiveness at $r = 0$

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 in [32], [52], [53], i.e.,

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}, \tag{10}$$

where $x_{i,k}$ is the kth component of spatial coordinates x_i of ith firefly and d is the number of dimensions. For $d = 2$, (10) can be written as

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{11}$$

3.4 Firefly movement

The firefly i is attracted and moved 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

$$x_i = x_i + \beta_0 \exp(-\gamma r_{ij}^2)(x_j - x_i) + \alpha(rand - 0.5). \tag{12}$$

In (12), the first term is the current position of a firefly, the second term is the attractiveness of the firefly to light intensity seen by neighbour fireflies and the third term is the random movement of firefly when there are no brighter firefly. The coefficient α is a randomization parameter, where $\alpha \in [0,1]$, while rand is a random number, $rand \in [0,1]$.

3.5 Special cases

The firefly algorithm has two special cases based on the absorption coefficient γ. The first case when $\gamma = \infty$, in this case, the attractiveness to light intensity is almost zero and the fireflies cannot see each other. Therefore, the firefly algorithm behaves like a random walk method.

The second case, when $\gamma = 0$, the light intensity does not decreases as the distance r between two fireflies increases and the attractive coefficient is constant $\beta = \beta_0$. In this case the firefly algorithm corresponds to the standard particle swarm optimization algorithm (PSO).

3.6 Firefly algorithm

In this subsection, we highlight the main steps of the standard Firefly algorithm (FFA)as shown in Algorithm 4 as follows.

– **Step 1.** The algorithm starts with the initial values of the most important parameters such as the randomization parameter α, firefly attractiveness β_0,

Algorithm 3 The Nelder-Mead Algorithm

1. Let x_i denote the list of vertices in the current simplex, $i = 1,\ldots,n+1$.
2. Order. Order and re-label the $n+1$ vertices from lowest function value $f(x_1)$ to highest function value $f(x_{n+1})$ so that $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$.
3. Reflection. Compute the reflected point x_r by
$x_r = \bar{x} + \rho(\bar{x} - x_{(n+1)})$, where $\bar{x}$ is the centroid of the n best points,
$\bar{x} = \sum(x_i/n), i = 1,\ldots,n$.
if $f(x_1) \leq f(x_r) < f(x_n)$ **then**
  replace x_{n+1} with the reflected point x_r and go to Step 7.
end if
4. Expansion.
if $f(x_r) < f(x_1)$ **then**
  Compute the expanded point x_e by $x_e = \bar{x} + \chi(x_r - \bar{x})$.
end if
if $f(x_e) < f(x_r)$ **then**
  Replace x_{n+1} with x_e and go to Step 7.
else
  Replace x_{n+1} with x_r and go to Step 7.
end if
5. Contraction.
if $f(x_r) \geq f(x_n)$ **then**
  Perform a contraction between $\bar{x}$ and the best among x_{n+1} and x_r.
end if
if $f(x_n) \leq f(x_r) < f(x_{n+1})$ **then**
  Calculate $x_{oc} = \bar{x} + \tau(x_r - \bar{x})$ {*Outside contract.*}
end if
if $f(x_{oc}) \leq f(x_r)$ **then**
  Replace x_{n+1} with x_{oc} and go to Step 7.
else
  Go to Step 6.
end if
if $f(x_r) \geq f(x_{(n+1)})$ **then**
  Calculate $x_{ic} = \bar{x} + \tau(x_{n+1} - \bar{x})$. {*Inside contract*}
end if
if $f(x_{ic}) \geq f(x_{(n+1)})$ **then**
  replace x_{n+1} with x_{ic} and go to Step 7.
else
  go to Step 6.
end if
6. Shrink. Evaluate the n new vertices
$x' = x_1 + \phi(x_i - x_1), i = 2,\ldots,n+1$.
Replace the vertices $x_2,\ldots,x_{n+1}$ with the new vertices $x'_2,\ldots,x'_{n+1}$.
7. Stopping Condition. Order and re-label the vertices of the new simplex as $x_1, x_2, \ldots, x_{n+1}$ such that $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$
if $f(x_{n+1}) - f(x_1) < \varepsilon$ **then**
  Stop, where $\varepsilon > 0$ is a small predetermined tolerance.
else
  Go to Step 3.
end if

media light absorption coefficient γ, population size P and finally the maximum generation number MGN which is the standard termination criterion in the algorithm.

–Step 2. The initial population x_i, $i = \{1,\ldots,P\}$ is randomly generated and the fitness function of each solution $f(x_i)$ in the population is evaluated by calculating its corresponding objective function.

–Step 3. The following steps are repeated until the termination criterion satisfied which is to reach the desired number of iterations MGN

Step 3.1. For each 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, then firefly i will move towards the firefly j as in (12).

Step 3.2. Obtain attractive varies with distance r via $\exp(-\gamma r^2)$ as in (9).

Step 3.3. Evaluate each solution x_i in the population and update the corresponding light intensity I_i of each solution.

Step 3.4. Rank the fireflies and find the current best solution x_{best}.

–Step 4. Produce the best found solution so far.

Algorithm 4 Firefly algorithm

Set the initial values of the randomization parameter α, firefly attractiveness β_0, media light absorption coefficient γ, population size P and maximum generation number MGN.
Generate the initial population x_i randomly, $i = \{1,\ldots,P\}$ {**Initialization**}
Evaluate the fitness function $f(x_i)$ of all solutions in the population
Light intensity I_i at x_i is determined by $f(x_i)$
Set $t = 0$
repeat
  for $(i = 1; i < P; i{+}{+})$ **do**
    for $(j = 1; j < i; j{+}{+})$ **do**
      if $I_i^{(t+1)} < I_j^{(t+1)}$ **then**
        Move firefly i towards j
      end if
      Obtain attractiveness β, where $\beta(r) = \beta_0 e^{-\gamma r^2}$
      Evaluate the fitness function $f(x_i)$ of all solutions in the population
      Update light intensity I_i
    end for
  end for
  Rank the solutions and keep the best solution x_{best} found so far in the population
  $t = t + 1$
until $t < MGN$
Produce the optimal solution.

4 The proposed DSFFA algorithm

In this section, we present the proposed DSFFA algorithm in details and report all DSFFA parameters and their best values in Table 2.

Algorithm 5 DSFFA algorithm

Set the initial values of the randomization parameter α, firefly attractiveness β_0, media light absorption coefficient γ, population size P and maximum generation number MGN.
Generate the initial population x_i randomly, $i = \{1,\ldots,P\}$ {**Initialization**}
Evaluate the fitness function $f(x_i)$ of all solutions in the population
Light intensity I_i at x_i is determined by $f(x_i)$
Set $t = 0$
repeat
  for $(i = 1; i < P; i{+}{+})$ **do**
    for $(j = 1; j < i; j{+}{+})$ **do**
      if $I_i^{(t+1)}) < I_j^{(t+1)})$ **then**
        Move firefly i towards j
      end if
      Obtain attractiveness β, where $\beta(r) = \beta_0 e^{-\gamma r^2}$
      Evaluate the fitness function $f(x_i)$ of all solutions in the population
      Update light intensity I_i
    end for
  end for
  Rank the solutions and keep the best solution x_{best} found so far in the population
  Apply pattern search method on the best solution x_{best} as shown in Algorithm 2 {**Pattern search algorithm**}
  $t = t + 1$
until $t < MGN$
Apply Nelder-Mead method on the N_{elite} best solutions as shown in Algorithm 3 {**Final intensification**}

We present the main steps of the proposed DSFFA algorithm in Algorithm 5 and list it as follows.

- **Step 1.** The algorithm starts with the initial values of the randomization parameter α, firefly attractiveness β_0, media light absorption coefficient γ, population size P and maximum generation number MGN.
- **Step 2.** The initial population is randomly generated x_i, $i = \{1,\ldots,P\}$.
- **Step 3.** Each solution in the population is evaluated by calculating its corresponding fitness value $f(x_i)$.
- **Step 4.** The light intensity I_i of each solution x_i in the population is determined by its corresponding fitness value $f(x_i)$.
- **Step 5.** The following steps are repeated until the termination criterion is satisfied which is to reach the desired number of iterations MGN.

  Step 5.1. For each 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 in (12).

  Step 5.2. Obtain attractive varies with distance r via $\exp(-\gamma r^2)$ as in (9).

  Step 5.3. Evaluate each solution x_i in the population and update the corresponding light intensity I_i of each solution.

  Step 5.4. Rank the fireflies and find the current best solution x_{best}.

  Step 5.5. Apply the pattern search method as shown in Algorithm 2 on the best found solution so far. The PS method is used in order to increase the exploitation capability of the proposed algorithm.
- **Step 6.** In order to accelerate the search and avoid running the algorithm with more iterations without any improvement in the results, we apply the Nelder-Mead method on the best found solution in the previous stage as a final intensification process.

5 Numerical experiments

In order to investigate the efficiency of the DSFFA , we present the general performance of it with different benchmark functions and compare the results of the proposed algorithm against variant algorithms. We program DSFFA via MATLAB and take the results of the comparative algorithms from their original papers. In the following subsections, we report the parameter setting of the proposed algorithm with more details and the properties of the applied test functions. Also, we present the performance analysis of the proposed algorithm with the comparative results between it and the other algorithms.

5.1 Parameter setting

In Table 2, we summarize the parameters of the DSFFA algorithm and their assigned values. These values are based on the common setting in the literature and determined through our preliminary numerical experiments.

Table 2: Parameter setting.

Parameters	Definitions	Values
P	Population size	20
α	Randomization parameter	0.5
β_0	Firefly attractiveness	0.2
γ	Light absorption coefficient	1
ε	Step size for checking descent directions	10^{-3}
m	Local PS repetition number	5
Δ_0	Initial mesh size	$(U_i - L_i)/3$
σ	Reduction factor of mesh size	0.01
MGN	Maximum generation number	2d
N_{elite}	No. of best solution for final intensification	1

- **Population size** P. The experimental tests show that the best population size is $P = 20$, increasing this number will increase the evaluation function values without any improvement in the obtained results.

–**Randomization parameter** α. The randomization parameter α is one of the most important parameters in the firefly algorithm. In our proposed algorithm, we find that the quality of the solution is related to the value of α parameter and obtain the best solution when we reduce the parameter α with a geometric progression reduction as the cooling schedule of simulated annealing. In this paper, the experimental tests show that the best initial value of α is $\alpha_0 = 0.5$ and the value of α is updated as the following.

$$delta = 1 - \left(\frac{10^{(-4)}}{0.9}\right)^{(1/MGN)},$$
$$\alpha = \quad (1 - delta)\alpha. \tag{13}$$

–**Firefly attractiveness** β_0. Firefly movement is based on the value of the attractiveness parameter β and updated as in (9). We set the initial value of attractiveness parameters $\beta_0 = 0.2$.

–**Media light absorption coefficient** γ. The firefly algorithm is very sensitive to media light absorption coefficient parameter γ. It turns out the best initial value of γ is 1.

–**Pattern search parameters**. DSFFA uses PS as a local search algorithm in order to refine the obtained solution from the firefly algorithm at each iteration. In PS the mesh size is initialized as Δ_0, in our experiments we set $\Delta_0 = (U_i - L_i)/3$ and when no improvement achieved in the exploration search process, the mesh size is deducted by using shrinkage factor σ. The experimental results show that the best value of σ is 0.1. The PS steps are repeated m times, in order to increase the exploitation process of the algorithm. In our experiment, we set $m = 3$ as a pattern search iteration number.

–**Stopping condition parameters**. DSFFA terminates the search when the number of iterations reaches to the desired maximum number of iterations or any other terminations depending on the comparison with other algorithms. In our experiment, we set the value of the maximum iteration number $MGN = 2d$, where d is the dimension of the problems.

–**Final intensification**. The best obtained solutions from the firefly algorithm and the pattern search method are collected in list in order to apply the Nelder-Mead method on them, the number of the solutions in this list is called N_{elit}. We set $N_{elit} = 1$ in order to avoid increasing in the function evaluation value, .

5.2 Integer programming optimization test problems

We test the efficiency of the DSFFA algorithm on 7 benchmark integer programming problems ($FI_1 - FI_7$). In Table 3, we list the properties of the benchmark functions (function number, dimension of the problem, problem bound and the global optimal of each problem) and report the functions with their definitions.

Table 3: The properties of the Integer programming test functions.

Function	Dimension (d)	Bound	Optimal
FI_1	5	[-100 100]	0
FI_2	5	[-100 100]	0
FI_3	5	[-100 100]	-737
FI_4	2	[-100 100]	0
FI_5	4	[-100 100]	0
FI_6	2	[-100 100]	-6
FI_7	2	[-100 100]	-3833.12

Test problem 1 [42]. This problem is defined by

$$FI_1(x) = \|x\|_1 = |x_1| + \ldots + |x_n|.$$

Test problem 2 [42]. This problem is defined by

$$FI_2 = x^T x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

Test problem 3 [17]. This problem is defined by

$$FI_3 = \begin{bmatrix} 15 & 27 & 36 & 18 & 12 \end{bmatrix} x + x^T \begin{bmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{bmatrix} x.$$

Test problem 4 [17]. This problem is defined by

$$FI_4(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2$$

Test problem 5 [17]. This problem is defined by

$$FI_5(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

Test problem 6 [41]. This problem is defined by

$$FI_6(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2.$$

Test problem 7 [17]. This problem is defined by

$$FI_7(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2.$$

5.3 The efficiency of the proposed DSFFA algorithm with integer programming problems

We verify the powerful of the proposed DSFFA with integer programming problems and compare the standard firefly algorithm with the proposed DSFFA algorithm without applying the final intensification process

(Nelder-Mead method). We set the same parameter values for both algorithms in order to make a fair comparison.

We show the efficiency of the proposed algorithm by selecting the functions FI_1, FI_2 and FI_3 and plotting the values of function values versus the number of iterations as shown in Figure 2. In Figure 2, the solid line refers to the proposed DSFFA results, while the dotted line refers to the standard firefly results after 100 iterations. Figure 2 shows that the function values rapidly decrease as the number of iterations increases for DSFFA results than those of the standard firefly algorithm. We can conclude from Figure 2 that the combination between the standard firefly algorithm with pattern search method can improve the performance of the standard firefly algorithm and accelerate the convergence of the proposed algorithm.

5.4 The general performance of the DSFFA algorithm with integer programming problems

In this subsection, we investigate the general performance of the proposed algorithm on the integer programming problems by plotting the values of function values versus the number of iterations as shown in Figure 3 for four test functions FI_4, FI_5, FI_6 and FI_7. Figure 3 depicts the results of the proposed algorithm without applying the Nelder-Mead method in the final stage of the algorithm after 100 iterations. We can conclude from Figure 3 that the function values of the proposed DSFFA rapidly decrease as the number of iterations increases and the hybridization between the firefly algorithm and the pattern search method can accelerate the search and help the algorithm to obtain the optimal or near optimal solution in reasonable time.

5.5 The efficiency of applying the Nelder-Mead method in the proposed DSFFA algorithm with integer programming problems

We apply Nelder-Mead method in the final stage of the proposed DSFFA algorithm in order to accelerate the convergence of the proposed algorithm and avoid running the algorithm with more iterations without any improvement or slow convergence in the obtained results. The results in Table 4 show the mean evaluation function values of the proposed DSFFA without and with applying Nelder-Mead method, respectively. We report the best results in **boldface** text. The results in Table 4 show that invoking the Nelder-Mead method in the final stage can accelerate the search and help the algorithm to reach to the optimal or near optimal solution faster than the proposed algorithm without applying the Nelder-Mead method.

5.6 DSFFA and other algorithms

We compare DSFFA with four benchmark algorithms (particle swarm optimization with different variants algorithms) in order to verify of the efficiency of the proposed algorithm. Before we discuss the comparison results of all algorithms, we present a brief description about the comparative four algorithms [39].

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

5.6.1 Comparison between RWMPSOg, RWMPSOl, PSOg, PSOl and DSFFA for integer programming problems.

In this subsection, we present the comparison results between our DSFFA algorithm and the other algorithms in order to verify of the efficiency of our proposed algorithm. We test the five comparative algorithms on 7 benchmark functions and report the results in Subsection 5.2. We take the results of the comparative algorithms from their original paper [39]. In Table 5, we report the minimum (min), maximum (max), average (Mean), standard deviation (St.D) and Success rate (%Suc) of the evaluation function values over 50 runs. We consider the run succeeds if the algorithm reaches to the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. We report the best results between the comparative algorithms in **boldface** text. The results in Table 5 show that the proposed DSFFA algorithm succeeds in all runs and obtains the desired objective value of each function faster than the other algorithms.

5.7 DSFFA and the branch and bound method

In order to verify of the powerful of the proposed algorithm, we apply another investigation on the integer programming problems by comparing the DSFFA algorithm against the branch and bound (BB) method [8], [9], [28], [33]. Before we discuss the comparative results between the proposed algorithm and the BB method, we present the BB method and the main steps of its algorithm for the sake of completeness.

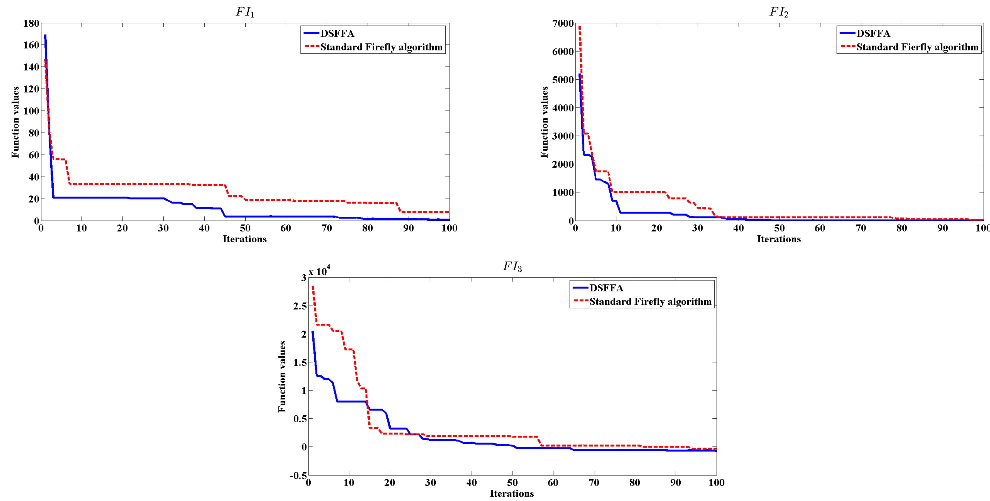

Fig. 2: The efficiency of the proposed DSFFA algorithm with integer programming problems

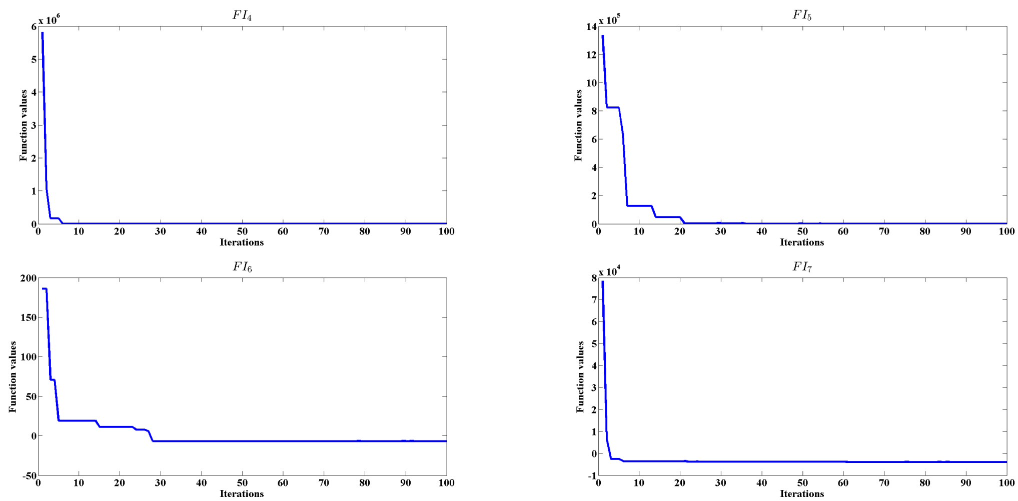

Fig. 3: The general performance of DSFFA algorithm with integer programming problems

Table 4: The efficiency of invoking the Nelder-Mead method in the final stage of DSFFA for $FI_1 - FI_7$ integer programming problems

Function	DSFFA without NM	DSFFA with NM
FI_1	1716.23	**533.64**
FI_2	924.58	**126.8**
FI_3	1315.24	**629.12**
FI_4	378.15	**157.34**
FI_5	1105.63	**801.52**
FI_6	245.67	**96.45**
FI_7	615.47	**154.84**

Table 5: Experimental results (min, max, mean, standard deviation and rate of success) of function evaluation for $FI_1 - FI_7$ test problems

Function	Algorithm	Min	Max	Mean	St.D	Suc
FI_1	RWMPSOg	17,160	74,699	27,176.3	8657	50
	RWMPSOl	24,870	35,265	30,923.9	2405	50
	PSOg	14,000	261,100	29,435.3	42,039	34
	PSOl	27,400	35,800	31,252	1818	50
	DSFFA	512	540	**533.64**	**5.59**	50
FI_2	RWMPSOg	252	912	578.5	136.5	50
	RWMPSOl	369	1931	773.9	285.5	50
	PSOg	400	1000	606.4	119	50
	PSOl	450	1470	830.2	206	50
	DSFFA	122	132	**126.8**	**2.42**	50
FI_3	RWMPSOg	361	41,593	6490.6	6913	50
	RWMPSOl	5003	15,833	9292.6	2444	50
	PSOg	2150	187,000	12,681	35,067	50
	PSOl	4650	22,650	11,320	3803	50
	DSFFA	553	699	**629.12**	**33.87**	50
FI_4	RWMPSOg	76	468	215	97.9	50
	RWMPSOl	73	620	218.7	115.3	50
	PSOg	100	620	369.6	113.2	50
	PSOl	120	920	390	134.6	50
	DSFFA	141	210	**157.34**	**15.36**	50
FI_5	RWMPSOg	687	2439	1521.8	360.7	50
	RWMPSOl	675	3863	2102.9	689.5	50
	PSOg	680	3440	1499	513.1	43
	PSOl	800	3880	2472.4	637.5	50
	DSFFA	624	1024	**801.52**	**96.82**	50
FI_6	RWMPSOg	40	238	110.9	48.6	50
	RWMPSOl	40	235	112	48.7	50
	PSOg	80	350	204.8	62	50
	PSOl	70	520	256	107.5	50
	DSFFA	90	110	**96.45**	**6.11**	
FI_7	RWMPSOg	72	620	242.7	132.2	50
	RWMPSOl	70	573	248.9	134.4	50
	PSOg	100	660	421.2	130.4	50
	PSOl	100	820	466	165	50
	DSFFA	128	234	**154.84**	**16.60**	50

5.7.1 Branch and bound method

The branch and bound method (BB) is one of the most widely used method for solving optimization problems. The main idea of BB method is the feasible region of the problem is partitioned subsequently into several sub regions, this operation is called branching. The lower and upper bounds values of the function can be determined over these partitions, this operation is called bounding. We report the main steps of BB method in Algorithm 6, and summarize the BB algorithm in the following steps.

- **Step 1.** The algorithm starts with a relaxed feasible region $M_0 \supset S$, where S is the feasible region of the problem. This feasible region M_0 is partitioned into finitely many subsets M_i.
- **Step 2.** For each subset M_i, the lower bound β and the upper bound α will be determined, where

Algorithm 6 The branch and bound algorithm

```
Set the feasible region M_0, M_0 ⊃ S
Set i = 0
repeat
    Set i = i + 1
    Partition the feasible region M_0 into many subsets M_i
    For each subset M_i, determine lower bound β, where β = min β(M_i)
    For each subset M_i, determine upper bound α, where α = min α(M_i)
    if (α = β)||(α − β ≤ ε) then
        Stop
    else
        Select some of the subset M_i and partition them
    end if
    Determine new bound on the new partition elements
until (i ≤ m)
```

$\beta(M_i) \leq \inf f(M_i \bigcap S) \leq \alpha(M_i)$, f is the objective function.

–**Step 3.** The algorithm is terminated, if the bounds are equal or very close, i.e $\alpha = \beta$ (or $\alpha - \beta \leq \varepsilon$), ε is a predefined positive constant.

–**Step 4.** Otherwise, if the bounds are not equal or very close, some of the subsets M_i are selected and partitioned in order to obtain a more refined partition of M_0.

–**Step 5.** The procedure is repeated until termination criteria are satisfied.

5.7.2 Comparison between the BB method and DSFFA for integer programming problems

In Table 6, we give the comparison results between the BB method and the proposed DSFFA. We take the results of the BB method from its original paper [27]. In [27], the BB algorithm transforms the initial integer problem programming problem to a continuous problem. For the bounding, the BB uses the sequential quadratic programming method to solve the generated sub problems. While for branching, BB uses depth first traversal with backtracking. We report the average (Mean), standard deviation (St.D) and rate of success (Suc) over 30 runs. We report the best mean evaluation values between the two algorithms in **boldface** text. The results in Table 6 show that the proposed algorithm results are better than the results of the BB method in all tested functions. The overall results in Table 6 show that the proposed algorithm is faster and more efficient than the BB method.

After applying the proposed algorithm on the integer programming problems and comparing it with different 5 algorithms, we conclude that the proposed DSFFA algorithm is a promising algorithm and can obtain the optimal or near optimal solution of the integer programming functions faster than the other comparative algorithms.

5.8 Minimax optimization test problems

In order to investigate the efficiency of the proposed algorithm, we consider another type of optimization problem, namely, minimax problem. We apply DSFFA algorithm on 10 benchmark minimax functions and report their properties in Table 7. We list the form of each function as follows.

Test problem 1 [50]. This problem is defined by

$$\begin{aligned} &\min\ FM_1(x),\\ &FM_1(x) = \max\ f_i(x),\ i=1,2,3,\\ &f_1(x) = x_1^2+x_2^4,\\ &f_2(x) = (2-x1)^2+(2-x_2)^2,\\ &f_3(x) = 2exp(-x_1+x_2). \end{aligned}$$

Test problem 2 [50]. This problem is defined by

$$\begin{aligned} &\min\ FM_2(x),\\ &FM_2(x) = \max\ f_i(x),\ i=1,2,3,\\ &f_1(x) = x_1^4+x_2^2,\\ &f_2(x) = (2-x1)^2+(2-x_2)^2,\\ &f_3(x) = 2exp(-x_1+x_2). \end{aligned}$$

Test problem 3 [50]. This problem is a nonlinear programming problem and transformed to minimax problem according to (4) and (5). It is defined by

$$\begin{aligned} FM_3(x) &= x_1^2+x_2^2+2x_3^2+x_4^2-5x_1-5x_2-21x_3+7x_4,\\ g_2(x) &= -x_1^2-x_2^2-x_3^3-x_4^2-x_1+x_2-x_3+x_4+8,\\ g_3(x) &= -x_1^2-2x_2^2-x_3^2-2x_4+x_1+x_4+10,\\ g_4(x) &= -x_1^2-x_2^2-x_3^2-2x_1+x_2+x_4+5. \end{aligned}$$

Test problem 4 [50]. This problem is a nonlinear programming problem and it is defined by

$$\begin{aligned} &\min\ FM_4(x),\\ &FM_4(x) = \max f_i(x)\ \ i=1,\ldots,5\\ &f_1(x) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+\\ &\quad +10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7,\\ &f_2(x) = f_1(x)+10(2x_1^2+3x_2^4+x_3+4x_4^2+5x_5-127),\\ &f_3(x) = f_1(x)+10(7x_1+3x_2+10x_3^2+x_4-x_5-282),\\ &f_4(x) = f_1(x)+10(23x_1+x_2^2+6x_6^2-8x_7-196),\\ &f_5(x) = f_1(x)+10(4x_1^2+x_2^2-3x_1x_2+2x_3^2+5x_6-11x_7. \end{aligned}$$

Test problem 5 [44]. This problem is defined by

$$\begin{aligned} &\min\ FM_5(x),\\ &FM_5(x) = \max\ f_i(x),\ i=1,2,\\ &f_1(x) = |x_1+2x_2-7|,\\ &f_2(x) = |2x_1+x_2-5|. \end{aligned}$$

Test problem 6 [44]. This problem is defined by

$$\begin{aligned} &\min\ FM_6(x),\\ &FM_6(x) = \max\ f_i(x),\\ &f_i(x) = |x_i|,\ i=1,\ldots,10. \end{aligned}$$

Test problem 7 [31]. This problem is defined by

$$\begin{aligned} &\min\ FM_7(x),\\ &FM_7(x) = \max\ f_i(x),\ i=1,2,\\ &f_1(x) = (x_1-\sqrt{(x_1^2+x_2^2)}cos\sqrt{x_1^2+x_2^2})^2+0.005(x_1^2+x_2^2)^2,\\ &f_2(x) = (x_2-\sqrt{(x_1^2+x_2^2)}sin\sqrt{x_1^2+x_2^2})^2+0.005(x_1^2+x_2^2)^2. \end{aligned}$$

Table 6: Experimental results (mean, standard deviation and rate of success) of function evaluation between BB and DSFFA for $FI_1 - FI_7$ test problems

Function	Algorithm	Mean	St.D	Suc
FI_1	BB	1167.83	659.8	30
	DSFFA	**534.86**	**3.77**	30
FI_2	BB	139.7	102.6	30
	DSFFA	**127.16**	**3.31**	30
FI_3	BB	4185.5	32.8	30
	DSFFA	**648.3**	**25.93**	30
FI_4	BB	316.9	125.4	30
	DSFFA	**157.46**	**17.01**	30
FI_5	BB	2754	1030.1	30
	DSFFA	**667.67**	**103.21**	30
FI_6	BB	211	15	30
	DSFFA	**140.53**	**11.36**	30
FI_7	BB	358.6	14.7	30
	DSFFA	**156.36**	**13.99**	30

Test problem 8 [31]. This problem is defined by

$$\begin{aligned}
&\min\ FM_8(x),\\
&FM_8(x) = \max\ f_i(x),\ \ i = 1,\dots,4,\\
f_1(x) = &\left(x_1-(x_4+1)^4\right)^2+\left(x_2-\left(x_1-(x_4+1)^4\right)^4\right)^2+\\
&2x_3^2+x_4^2-5\left(x_1-(x_4+1)^4\right)-5\left(x_2-(x1-\right.\\
&\left.(x_4+1)^4)^4\right)-21x_3+7x_4,\\
f_2(x) = &f_1(x)+10\left[\left(x_1-(x_4+1)^4\right)^2+\left(x_2-(x_1-\right.\right.\\
&\left.(x_4+1)^4)^4\right)^2+x_3^2+x_4^2+\left(x_1-(x_4+1)^4\right)-\\
&\left.\left(x_2-\left(x_1-(x_4+1)^4\right)^4\right)+x_3-x_4-8\right],\\
f_3(x) = &f_1(x)+10\left[\left(x_1-(x_4+1)^4\right)^2+2\left(x_2-(x_1-\right.\right.\\
&\left.(x_4+1)^4)^4\right)^2+x_3^2+2x_4^2-\left(x_1-(x_4+1)^4\right)-\\
&\left.x_4-10\right],\\
f_4(x) = &f_1(x)+10\left[\left(x_1-(x_4+1)^4\right)^2+\left(x_2-(x_1-\right.\right.\\
&\left.(x_4+1)^4)^4\right)^2+x_3^2+2\left(x_1-(x_4+1)^4\right)-\\
&\left.\left(x_2-\left(x_1-(x_4+1)^4\right)^4\right)-x_4-5\right].
\end{aligned} \tag{14}$$

Test problem 9 [31]. This problem is a nonlinear programming problem and transformed to minimax problem according to (4) and (5). It is defined by

$$\begin{aligned}
&\min\ FM_9(x),\\
&FM_9(x) = \max\ f_i(x),\ \ i = 1,\dots,5,\\
f_1(x) = &\ (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2\\
&+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7,\\
f_2(x) = &\ -2x_1^2-2x_3^4-x_3-4x_4^2-5x_5+127,\\
f_3(x) = &\ -7x_1-3x_2-10x_3^2-x_4+x_5+282,\\
f_4(x) = &\ -23x_1-x_2^2-6x_6^2+8x_7+196,\\
f_5(x) = &\ -4x_1^2-x_2^2+3x_1x_2-2x_3^2-5x_6+11x_7.
\end{aligned}$$

Table 7: Minimax test functions properties.

Function	Dimension (d)	Desired error goal
FM_1	2	1.95222245
FM_2	2	2
FM_3	4	-40.1
FM_4	7	247
FM_5	2	10^{-4}
FM_6	10	10^{-4}
FM_7	2	10^{-4}
FM_8	4	-44
FM_9	7	680
FM_{10}	4	0.1

Test problem 10 [31]. This problem is defined by

$$\begin{aligned}
&\min\ FM_{10}(x),\\
&FM_{10}(x) = \max|f_i(x)|,\quad i = 1,\dots,21,\\
&f_i(x) = \ x_1 exp(x_3t_i)+x_2 exp(x_4t_i)-\frac{1}{1+t_i},\\
&t_i = \ -0.5+\frac{i-1}{20}.
\end{aligned}$$

5.9 The efficiency of the proposed DSFFA algorithm with minimax problems

After verifying from the efficiency of the proposed algorithm with the integer programming problems, we

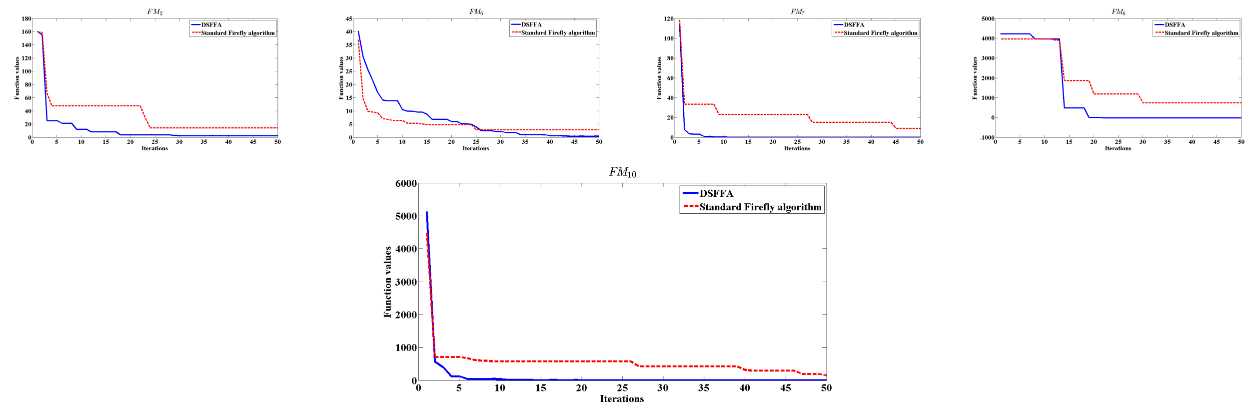

Fig. 4: The efficiency of the proposed DSFFA algorithm with minimax problems

investigate the efficiency of combining the firefly algorithm with the pattern search method to solve minimax problems. This test is applied without invoking the final intensification process (Nelder-Mead method). The parameter setting values for both algorithms are the same for both algorithms in order to make a fair comparison. We select the functions FM_2, FM_6, FM_7, FM_8 and FM_{10} to show the efficiency of the proposed algorithm and plot the values of function values versus the number of iterations as shown in Figure 4. In Figure 4, the solid line refers to the proposed DSFFA results, while the dotted line refers to the standard firefly results after 50 iterations. Figure 4 shows that the function values rapidly decrease as the number of iterations increases for DSFFA results than those of the standard firefly algorithm. The results in Figure 4 show that the combination between the standard firefly algorithm and the pattern search method can improve the performance of the standard firefly algorithm and accelerate the convergence of the proposed algorithm.

5.10 The general performance of the DSFFA algorithm with minimax problems

We verify of the general performance of the proposed DSFFA on the minimax problems by plotting the values of function values versus the number of iterations as shown in Figure 5 for five test functions FM_1, FM_3, FM_4, FM_5 and FM_9.

Figure 5 depicts the results of the proposed algorithm without applying the Nelder-Mead method in the final stage of the algorithm after 50 iterations. The results in Figure 5 show that the function values of the proposed DSFFA rapidly decrease as the number of iterations increases. We conclude that the hybridization between the firefly algorithm and the pattern search method can accelerate the search and help the algorithm to obtain the optimal or near optimal solution in a few iterations.

5.11 The efficiency of applying the Nelder-Mead method in the proposed DSFFA algorithm with minimax problems

We investigate the general performance of the proposed algorithm in order to verify the importance of invoking the Nelder-Mead method in the final stage as a final intensification process. The results in Table 8 show the mean evaluation function values of the proposed DSFFA without and with applying Nelder-Mead method, respectively. We report the best results in **boldface** text and show that invoking the Nelder-Mead method in the final stage enhance the general performance of the proposed algorithm and can accelerate the search to reach to the optimal or near optimal solution faster than the proposed algorithm without applying the Nelder-Mead method.

5.12 DSFFA and other algorithms

We compare DSFFA with three benchmark algorithms in order to verify of the efficiency of the proposed algorithm on minimax problems. Before we discuss the comparison results of all algorithms, we present a brief description about the comparative three algorithms.

– HPS2 [22]. HPS2 is Heuristic Pattern Search algorithm. In [22], the authors applied it for solving bound constrained minimax problems by combining the Hook and Jeeves (HJ) pattern and exploratory moves with a randomly generated approximate descent direction.
– UPSOm [36]. UPSOm is a Unified Particle Swarm Optimization algorithm, which combines the global and local variants of the standard PSO and incorporates a stochastic parameter to imitate mutation in evolutionary algorithms.

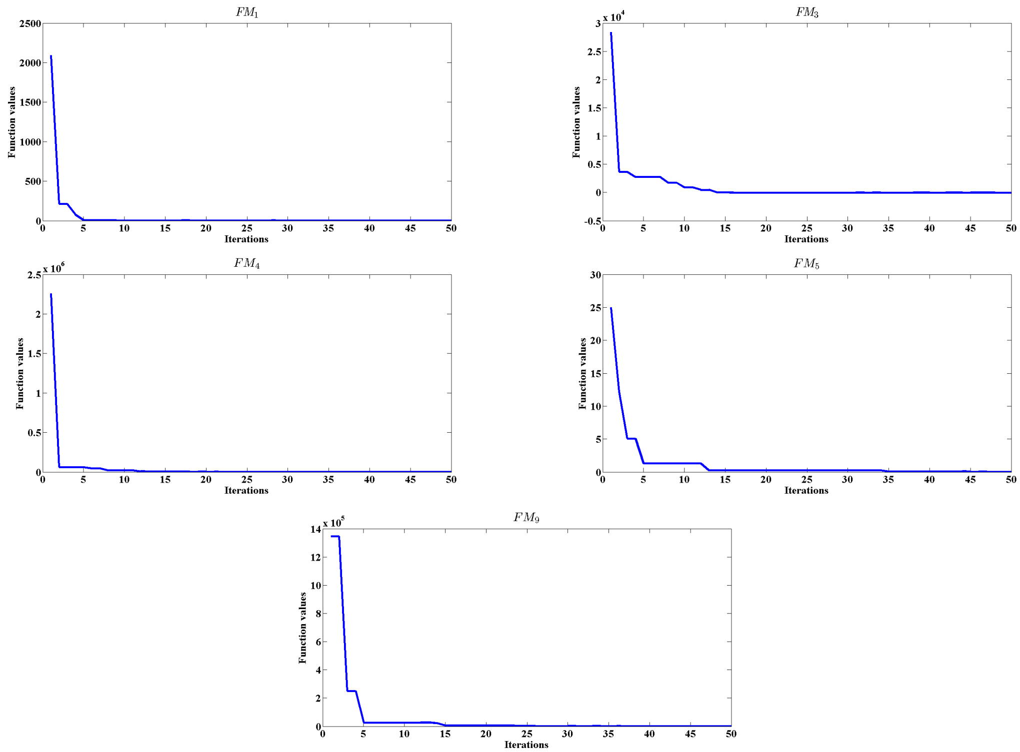

Fig. 5: The general performance of DSFFA algorithm with minimax problems

Table 8: The efficiency of invoking the Nelder-Mead method in the final stage of DSFFA for $FM_1 - FM_{10}$ minimax problems

Function	DSFFA without NM	DSFFA with NM
FM_1	1115.26	**334.61**
FM_2	690.45	**369.39**
FM_3	687.56	**444.31**
FM_4	1587.63	**611.45**
FM_5	438.59	**169.08**
FM_6	13,589	**8558.89**
FM_7	2568.25	**708.13**
FM_8	7569.15	**4388.71**
FM_9	7148.47	**5976.34**
FM_{10}	614.89	**294.22**

– RWMPSOg [39]. RWMPSOg is Random Walk Memetic Particle Swarm Optimization (with global variant), which combines the particle swarm optimization with random walk with direction exploitation.

5.12.1 Comparison between HPS2, UPSOm, RWMPSOg and DSFFA for minimax problems

In this subsection, we give the comparison results between our DSFFA algorithm and the other algorithms in order to verify of the efficiency of the proposed algorithm. We test the four comparative algorithms on 10 benchmark functions and report the results in Subsection 5.8. We take the results of the comparative algorithms from their original paper [22]. In Table 9, we report the average (Avg), standard deviation (SD) and Success rate (%Suc) over 100 runs. The mark (-) for FM_8 in HPS2 algorithm and FM_2, FM_8 and FM_9 in RWMPSOg algorithm in Table 9 means that the results of these algorithms for these functions are not reported in their original paper. The run succeeds if the algorithm reaches the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. The results in Table 9, show that the proposed DSFFA

Table 9: Evaluation function for the minimax problems $FM_1 - FM_{10}$

Algorithm	Problem	Avg	SD	%Suc
HPS2	FM_1	1848.7	2619.4	99
	FM_2	635.8	114.3	94
	FM_3	**141.2**	**28.4**	37
	FM_4	8948.4	5365.4	7
	FM_5	772.0	60.8	100
	FM_6	**1809.1**	2750.3	94
	FM_7	4114.7	1150.2	100
	FM_8	-	-	-
	FM_9	**283.0**	**123.9**	64
	FM_{10}	324.1	173.1	100
UPSOm	FM_1	1993.8	853.7	100
	FM_2	1775.6	241.9	100
	FM_3	1670.4	530.6	100
	FM_4	12,801.5	5072.1	100
	FM_5	1701.6	184.9	100
	FM_6	18,294.5	2389.4	100
	FM_7	3435.5	1487.6	100
	FM_8	6618.50	2597.54	100
	FM_9	2128.5	597.4	100
	FM_{10}	3332.5	1775.4	100
RWMPSOg	FM_1	2415.3	1244.2	100
	FM_2	-	-	-
	FM_3	3991.3	2545.2	100
	FM_4	7021.3	1241.4	100
	FM_5	2947.8	257.0	100
	FM_6	18,520.1	776.9	100
	FM_7	1308.8	505.5	100
	FM_8	-	-	-
	FM_9	-	-	-
	FM_{10}	4404.0	3308.9	100
DSFFA	FM_1	**334.61**	**28.30**	100
	FM_2	**369.39**	**37.98**	100
	FM_3	444.31	130.37	100
	FM_4	**611.45**	**191.82**	80
	FM_5	**169.08**	**31.43**	100
	FM_6	8558.89	**113.133**	100
	FM_7	**708.13**	121.88	100
	FM_8	**4388.71**	212.09	50
	FM_9	5976.34	**146.7**	50
	FM_{10}	**294.22**	**98.38**	90

algorithm succeeds in all runs and obtains the objective value of each function faster than the other algorithms, except for functions FM_3, FM_6 and FM_9, Although the rate of success is 37%, 94% and 64% for FM_3, FM_6 and FM_9 in HPS2 algorithm, respectively, the proposed DSFFA algorithm can obtain its results with these function with 100% rate of success.

5.13 DSFFA and SQP method

The last test for our proposed algorithm is to compare the DSFFA with another known method which is called sequential quadratic programming method (SQP). In the following subsection, we highlight the main steps of the SQP method and how it works.

5.13.1 Sequential quadratic programming (SQP)

In 1963 [49], Wilson proposed the first sequential quadratic programming (SQP) method for the solution of constrained nonlinear optimization problems. Since then, SQP methods have evolved into a powerful and effective class of methods for a wide range of optimization problems. SQP is one of the most effective methods for nonlinearly constrained optimization problems. SQP generates steps by solving quadratic subproblems; it can be used both in trust-region and line search approaches.

Table 10: Experimental results (mean, standard deviation and rate of success) of function evaluation between SQP and DSFFA for $FM_1 - FM_{10}$ test problems

Function	Algorithm	Mean	St.D	Suc
FM_1	SQP	4044.5	8116.6	24
	DSFFA	**341.72**	**24.25**	30
FM_2	SQP	8035.7	9939.9	18
	DSFFA	**373.62**	**45.76**	30
FM_3	SQP	**135.5**	**21.1**	30
	DSFFA	469.12	130.75	30
FM_4	SQP	20,000	0.0	0.0
	DSFFA	**6089.02**	**186.52**	30
FM_5	SQP	**140.6**	**38.5**	30
	DSFFA	177.84	44.49	30
FM_6	SQP	**611.6**	200.6	30
	DSFFA	8523.82	**108.60**	30
FM_7	SQP	15,684.0	7302.0	10
	DSFFA	**691.48**	**104.17**	30
FM_8	SQP	20,000	0.0	0.0
	DSFFA	**4374.68**	**221.69**	20
FM_9	SQP	20,000	0.0	0.0
	DSFFA	**6018.46**	**160.23**	15
FM_{10}	SQP	4886.5	8488.4	22
	DSFFA	**326.48**	**96.81**	30

SQP is suitable for small and large problems and it is appropriate to solving problems with significant nonlinearities.

The SQP method can be viewed as a generalization of Newton's method for unconstrained optimization in that it finds a step away from the current point by minimizing a quadratic model of the problem.

We summarize the main steps of the SPQ method.

– **Step 1.** The SQP algorithm starts with an initial solution x_0 and the initialized Hessian matrix of the objective function.
– **Step 2.** At each iteration, the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method has been used to calculate a positive definite quasi-Newton approximation of the Hessian matrix, where the Hessian update is calculated as the following

$$H_{n+1} = H_n + \frac{q_n q_n^T}{q_n^T s_n} - \frac{H_n^T H_n}{s_n^T H_n s_n}, \quad (15)$$

where $s_n = x_{n+1} - x_n$ and $q_n = \nabla f(x_{n+1})$
– **Step 3.** Solve the QP problem in z as the following

$$min\ q(z) = 1/2z^T Hz + c^T z. \quad (16)$$

– **Step 4.** Use the solution z_n to calculate the new potential solution

$$x_{n+1} = x_n + \alpha_n z_n \quad (17)$$

where α_n is a step length and determined through line search.

For more information about SQP algorithm, we refer the interested reader to [14] and [16].

In Subsection 5.8, we report the results of the two comparative algorithms on 10 benchmark functions. We take the results of the SQP algorithm from paper [27]. In Table 10, we report the average (Avg), standard deviation (SD) and Success rate (%Suc) over 30 runs. The run succeeds if the algorithm reaches the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. The results in Table 10, show that the proposed DSFFA algorithm outperforms the SQP algorithm in 7 of 10 functions, while the results of SQP algorithm are better than our proposed algorithm for functions FM_3, FM_5 and $FM-6$. We can conclude from this comparison that the proposed DSFFA outperforms the SQP algorithm in most of tested minimax problems.

6 Conclusion

In this paper, we propose a new hybrid algorithm, direct search Firefly algorithm (DSFFA), by combining the Firefly algorithm with the pattern search and the Nelder Mead methods in order to solve integer programming and minimax problems. In the proposed algorithm, we try to balance between the exploration and exploitation process in the proposed algorithm. The Firefly algorithm has a good capability of making the exploration and exploitation process, however we increase the capability of the exploitation process in the Firefly algorithm by applying the pattern search algorithm as a local search method and the Nelder Mead method in the final stage of the algorithm in order to refine the best obtained solution instead of running the algorithm with more iterations without any improvement in the results. Also, we intensely test DSFFA algorithm on 17 benchmark functions 7 integer programming problems and 10 minimax problems. Moreover, we compare the proposed algorithm against other 5 algorithms to investigate its performance for solving integer programming problems and 4 algorithms to test its performance for solving minimax problems. Furthermore, the numerical results indicate that the proposed DSFFA algorithm is a promising algorithm and suitable to find a global optimal solution or near optimal solution of the tested functions with their different properties in reasonable time.

Acknowledgments

The research of the 1st author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The postdoctoral fellowship of the 2nd author is supported by NSERC.

References

[1] T. Apostolopoulos and A. Vlachos, Application of the firefly algorithm for solving the economic emissions load dispatch problem, International Journal of Combinatorics, Volume 2011, 2011.
[2] S. K. Azad and S. K. Azad, Optimum design of structures using an improved firefly algorithm, International Journal of Optimization in Civil Engineering, Vol. 1, No. 2, pp. 327–340, 2011.
[3] N. Bacanin and M. Tuba, Artificial Bee Colony (ABC) algorithm for constrained optimization improved with genetic operators, Studies in Informatics and Control, Vol. 21, Issue 2, pp. 137–146, 2012.
[4] N. Bacanin, I. Brajevic and M. Tuba, Firefly algorithm applied to integer programming problems, Recent Advances in Mathematics, 2013.
[5] H. Banati and M. Bajaj, Firefly based feature selection approach, Int. J. Computer Science Issues, Vol. 8, No. 2, pp. 473–480, 2011.
[6] J.W. Bandler, and C. Charalambous, Nonlinear programming using minimax techniques. Journal of Optimization Theory and Applications, Vol. 13, pp. 607–619, 1974.
[7] B. Basu and G. K. Mahanti, Firefly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna, Progress in Electromagnetic Research B., Vol. 32, pp.169–190, 2011.
[8] B. Borchers and J.E. Mitchell, Using an interior point method in a branch and bound algorithm For integer programming, Technical Report, Rensselaer Polytechnic Institute, July 1992.
[9] B. Borchers and J.E. Mitchell, A computational comparison of branch and bound and outer approximation methods for 0-1 mixed integer nonlinear programs, Computers and Operations Research, Vol. 24, No. 8, pp. 699–701, 1997.
[10] A. Chatterjee, G. K. Mahanti, and A. Chatterjee, Design of a fully digital controlled reconfigurable switched beam conconcentric ring array antenna using firefly and particle swarm optimization algorithm, Progress in Electromagnetic Research B., Vol. 36, pp. 13–131, 2012.
[11] S. A. Chu, 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), Vol. 4099, LNAI, pp. 854–858, 2006.
[12] D. Z. Du and P.M. Pardalose, Minimax and applications, Kluwer, 1995.
[13] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.
[14] R. Fletcher, Practical method of optimization, Vol.1 & 2, John Wiley and Sons, 1980.
[15] A. H. Gandomi, X. S. Yang, and A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with Computers, Vol. 27, article DOI 10.1007/s00366-011-0241-y, 2011.
[16] P. E. Gill, W. Murray and M.H. Wright, Practical Optimization, Academic Press, London, 1981.
[17] A. Glankwahmdee, J.S. Liebman and G.L. Hogg, Unconstrained discrete nonlinear programming. Engineering Optimization, Vol. 4, pp. 95–107, 1979.
[18] F.S. Hillier and G. J. Lieberman, Introduction to operations research, MCGraw-Hill, 1995.
[19] M.H. Horng, Y.X. Lee, M.C. Lee and R.J. Liou, Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis, in: Theory and New Applications of Swarm Intelligence (Edited by R. Parpinelli and H. S. Lopes), pp. 115–132, 2012.
[20] M. H. Horng, Vector quantization using the firefly algorithm for image compression, Expert Systems with Applications, Vol. 39, pp. 1078–1091, 2012.
[21] R. Hooke and T. A. Jeeves, Direct search , Solution of numerical and statistical problems, J. Assoc. Comput. Mach., pp. 212–229, 1961.
[22] A. C. P. Isabel, E. Santo and E. Fernandes, Heuristics pattern search for bound constrained minimax problems, computational science and its applications- Vol. 6784, pp. 174–184, ICCSA 2011.
[23] R. Jovanovic and M. Tuba, An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem, Applied Soft Computing, Vol. 11, Issue 8, pp. 5360–5366, 2011.
[24] R. Jovanovic and M. Tuba, Ant colony optimization algorithm with pheromone correction strategy for minimum connected dominating set problem, Computer Science and Information Systems (ComSIS), Vol. 9, Issue 4, Dec 2012.
[25] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization, Vol. 39, No. 3, pp. 459–471, 2007.
[26] J. Kennedy and R. C. Eberhart, Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks, Vol. 4, pp. 1942–1948, 1995.
[27] E.C. Laskari, K.E. Parsopoulos and M.N. Vrahatis, Particle swarm optimization for integer programming, Proceedings of the IEEE 2002 Congress on Evolutionary Computation, Honolulu (HI), pp. 1582–1587, 2002.

[28] E. L. Lawler and D. W. Wood, Branch and bound methods: A Survey, Operations Research, Vol. 14, pp. 699–719, 1966.

[29] X. L. Li, Z.J. Shao, and J.X. Qian, Optimizing method based on autonomous animats: Fish-swarm algorithm. Xitong Gongcheng Lilun yu Shijian/System Engineer- ing Theory and Practice, Vol. 22, No. 11, pp. 32, 2002.

[30] G. Liuzzi, S. Lucidi and M. Sciandrone, A derivative-free algorithm for linearly constrained finite minimax problems. SIAM Journal on Optimization, Vol. 16, pp. 1054–1075, 2006.

[31] L. Lukan and J. Vlcek, Test problems for nonsmooth unconstrained and linearly constrained optimization, Technical report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 2000.

[32] 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, pp. 97–106, Springer, Wroclaw, Poland, October 2009.

[33] V.M. Manquinho, J.P. Marques Silva, A.L. Oliveira and K.A. Sakallah, Branch and bound algorithms for highly constrained integer programs, Technical Report, Cadence European Laboratories, Portugal, 1997.

[34] J. A. Nelder and R. Mead, A simplex method for function minimization, Computer journal, Vol. 7, pp. 308–313, 1965.

[35] G. L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, editor. Handbooks in OR & MS, volume 1, Elsevier, 1989.

[36] K. E. Parsopoulos and M.N. Vrahatis, Unified particle swarm optimization for tackling operations research problems. in Proceeding of IEEE 2005 swarm Intelligence Symposium, Pasadena, USA, pp. 53–59, 2005.

[37] S. Palit, S. Sinha, M. Molla, A. Khanra and M. Kule, A cryptanalytic attack on the knapsack cryptosystem using binary Firefly algorithm, in: 2nd Int. Conference on Computer and Communication Technology (ICCCT), 15-17 Sept 2011, India, pp. 428–432, 2011.

[38] M. K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, Control Systems, IEEE, Vol. 22, No. 3, pp. 52–67, 2002.

[39] Y. G. Petalas, K.E. Parsopoulos and M. N. Vrahatis, Memetic particle swarm optimization, Ann Oper Res, Vol. 156, pp. 99–127, 2007.

[40] E. Polak, J.O. Royset and R.S. Womersley, Algorithms with adaptive smoothing for finite minimax problems, Journal of Optimization Theory and Applications, Vol. 119, pp. 459–484, 2003.

[41] S. S. Rao, Engineering optimization-theory and practice. Wiley, New Delhi, 1994.

[42] G. Rudolph, An evolutionary algorithm for integer programming. In: Davidor Y, Schwefel H-P, Mnner R (eds), pp. 139–148. Parallel Problem Solving from Nature Vol. 3, 1994.

[43] M.K. Sayadi M. K., R. Ramezanian and N. N. Ghaffari, A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems, Int. J. of Industrial Engineering Computations, Vol. 1, pp. 1–10, 2010.

[44] H. P. Schwefel, Evolution and optimum seeking, New York, Wiley, 1995.

[45] 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, pp. 165–172, 2012.

[46] D. Teodorovic and M. DellOrco, Bee colony optimization cooperative learning approach to complex transportation problems. In Advanced OR and AI Methods in Transportation: Proceedings of 16th MiniEURO Conference and 10th Meeting of EWGT (13-16 September 2005).Poznan: Publishing House of the Polish Operational and System Research, pp. 51–60, 2005.

[47] M. Tuba, N. Bacanin and N. Stanarevic, Adjusted artificial bee colony (ABC) algorithm for engineering problems, WSEAS Transaction on Computers, Vol. 11, Issue 4, pp. 111–120, 2012.

[48] M. Tuba, M. Subotic and N. Stanarevic, Performance of a modified cuckoo search algorithm for unconstrained optimization problems, WSEAS Transactions on Systems, Vol. 11, Issue 2, pp. 62–74, 2012.

[49] B. Wilson, A simplicial Algorithm for Concave Programming, PhD thesis, Harvard University, 1963.

[50] S. Xu, Smoothing method for minimax problems, Computational Optimization and Applications, Vol. 20, pp. 267–279, 2001.

[51] X. S. Yang, Firefly algorithm, stochastic test functions and design optimization, International Journal of Bio-Inspired Computation, Vol. 2, No. 2, pp. 78–84, 2010.

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

[53] X. S. Yang, Firefly algorithms for multimodal optimisation, Proc. 5th Symposium on Stochastic Algorithms, Foundations and Applications, (Eds. O. Watanabe and T. Zeugmann), Lecture Notes in Computer Science, Vol. 5792: pp. 169–178, 2009.

[54] X. S. Yang. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pages 6574, 2010.

[55] X. S. Yang, Swarm-based metaheuristic algorithms and no-free-lunch theorems, in: Theory and New Applications of Swarm Intelligence (Eds. R. Parpinelli and H. S. Lopes), Intech Open Science, pp. 1–16, 2012.

[56] A. Yousif, A. H. Abdullah, S. M. Nor, and A. A. Abdelaziz, Scheduling jobs on grid computing using firefly algorithm, J. Theoretical and Applied Information Technology, Vol. 33, No. 2, pp. 155–164, 2011.

[57] S. Zuhe, A. Neumaier and M.C. Eiermann, Solving minimax problems by Interval Methods, BIT, Vol. 30, pp. 742–751, 1990.

Mohamed A. Tawhid got his PhD in Applied Mathematics from the University of Maryland Baltimore County, Maryland, USA. From 2000 to 2002, he was a Postdoctoral Fellow at the Faculty of Management, McGill University, Montreal, Quebec, Canada. Currently, he is a full professor at Thompson Rivers University. His research interests include nonlinear/stochastic/heuristic optimization, operations research, modelling and simulation, data analysis, and wireless sensor network. He has published in journals such as Computational Optimization and Applications, J. Optimization and Engineering, Journal of Optimization Theory and Applications, European Journal of Operational Research, Journal of Industrial and Management Optimization, Journal Applied Mathematics and Computation, etc. Mohamed Tawhid published more than 40 referred papers and edited 5 special issues in J. Optimization and Engineering (Springer), J. Abstract and Applied Analysis, J. Advanced Modeling and Optimization, and International Journal of Distributed Sensor Networks. Also, he has served on editorial board several journals. Also, he has worked on several industrial projects in BC, Canada.

Ahmed F. Ali Received the B.Sc., M.Sc. and Ph.D. degrees in computer science from the Assiut University in 1998, 2006 and 2011, respectively. Currently, he is a Postdoctoral Fellow at Thompson Rivers University, Kamloops, BC Canada. In addition, he is an Assistant Professor at the Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt. He served as a member of Computer Science Department Council from 2014-2015. He worked as director of digital library unit at Suez Canal University; he is a member in SRGE (Scientific Research Group in Egypt). He also served as a technical program committee member and reviewer in worldwide conferences. Dr. Ali research has been focused on meta-heuristics and their applications, global optimization, machine learning, data mining, web mining, bioinformatics and parallel programming. He has published many papers in international journals and conferences and he has uploaded some meta-heuristics lectures in slidshare website.