Accelerated Bat Algorithm for Solving Integer Programming Problems

Ahmed Fouad Ali

Computer Science Department, Faculty of Computers and Information
Suez Canal University, Ismailia, Egypt.
ahmed fouad@ci.suez.edu.eg

Abstract

In this paper, we present a new hybrid algorithm for solving integer-programming problems. The proposed algorithm is called accelerated bat algorithm (ABATA). In ABATA, we try to accelerate the search process by invoking the Nelder-Mead method as a local search method in order to refine the best obtained solution at each iteration. The bat algorithm has a good ability to perform a wide exploration and a deep exploitation search, while the Nelder-Mead method has a powerful performance as a local search method and can enhance the exploitation ability of the proposed algorithm. The general performance of ABATA is tested on seven integer-programming problems and compared against four benchmark algorithms. The experimental results indicate that ABATA is a promising algorithm and can obtain a global optimal solution or near optimal solution in reasonable time.

Keyboard: *Bat algorithm, Nelder-Mead method, Integer programming problems, Optimization problems.*

1. Introduction

In the past two decades, many meta-heuristic algorithms have been applied to solve global optimization problems, these algorithms are inspired from the behavior of a group of social organisms. They are called nature inspired algorithms or swarm intelligence (SI) algorithms, such as Ant Colony Optimization (ACO) [4], Artificial Bee Colony (ABC) [10], Particle Swarm Optimization (PSO) [11], Bacterial foraging [19], Bat Algorithm (BA) [30], Bee Colony Optimization (BCO) [24], Wolf search [23], Cat swarm [3], Cuckoo search [29], Firefly algorithm (FA) [28], [29], Fish swarm/school [14], etc. SI algorithms have been widely used to solve unconstrained and constrained problems and their applications. However they have been applied in a few works to solve integer-programming problems, although the variety of many real life applications for this problem such as warehouse location problem, VLSI (very large scale integration) circuits design problems, robot path planning problems, scheduling problem, game theory, engineering design problems, [5], [6], [17], [31].

Bat algorithm (BA) is a promising nature inspired algorithm inspired from the echolocation behavior of the microbats [30]. BA has a good capability to balance the global exploration and the local exploitation during the search process. In this work, we propose a new hybrid bat algorithm and Nelder-Mead method by combining the bat algorithm with its powerful capability of performing a wide exploration and a deep exploitation and the Nelder-Mead method as a local search method to refine the best-obtained solution at each iteration. The proposed algorithm is called accelerated bat algorithm (ABATA). Invoking the Nelder-

Mead method in the proposed algorithm can accelerate the convergence instead of running the algorithm more iteration without any improvement in the results.

Branch and Bound (BB) is one of the most famous exact integer programming algorithm [1], [13], [15], however it suffer from high complexity, since they explore a hundred of nodes in a big tree structure when solving 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 [8], [9], artificial bee colony algorithm [2], [25], particle swarm optimization algorithm [12], [18], [20], cuckoo search algorithm [26] and firefly algorithm [27].

The main objective of this paper is to produce a new hybrid SI algorithm by combining the bat algorithm with the Nelder-Mead method in order to solve integer-programming problems [7]. In the proposed algorithm, we try to overcome the main problem of applying other SI algorithm, which is consuming expensive computation time.

Moreover, the general performance of the proposed ABATA is tested on well-known benchmark functions and has been compared against different algorithms. The experimental results indicate that ABATA is a promising algorithm and outperforms the other algorithms.
The rest of this paper is organized as follow. In Section 2, we present an overview on the related work for the proposed algorithm. Section 3 describes the proposed ABATA. Section 4 discusses the general performance of the proposed algorithm and reports the comparative experimental results on the benchmark functions. Finally, the conclusion makes up Section 5.

2. Related Work

In this section, we present an overview on the related work for the proposed algorithm and we highlight the definition of the integer-programming problem and the Nelder-Mead method with the main structure of its algorithm as follow.

2.1 The Definition of the Integer Programming Problem

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.

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

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

2.2 Nelder-Mead Method

In 1965, Nelder and Mead proposed the Nelder-Mead method (NM) [16], which is one of the most popular derivative-free nonlinear optimization algorithms. The NM method starts with $n+1$ points (vertices) $x_1, x_2, \dots, n+1$, these 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 method, these parameters are reported in Table 1 and are chosen to satisfy $\rho > 0$, $\chi > 1$, $0 < \zeta < 1$, and $0 < \sigma < 1$.

The main steps of the Nelder-Mead method are shown in Algorithm 1 and the summarization of these steps is shown as follow.

Step 1. The algorithm of the Nelder-Mead method starts with $n + 1$ vertices $x_{i,}\ i = 1, \dots, n + 1$, which are evaluated and ordered according to their fitness function.

Step 2. The refection process starts by computing the reflected point $x_r = \rho\ (\bar{x} - x_{(n+1)})$, where $\bar{x}$ is the average of all points except the worst and ρ is a coefficients of reflection parameter, $\rho > 0$.

Step 3. If the reflected point x_r is lower than the n^{th} point $f(x_n)$ and greater than the best point $f(x_1)$, then the reflected point is accepted and the iteration is terminated.

Step 4. If the reflected point is better than the best point, then the algorithm starts the expansion process by calculating the expanded point x_e, where $x_e = \chi\ (x_r - \bar{x})$, is an expansion parameter and $\chi > 1$.

Step 5. If x_e is better than the reflected point n^{th}, the expanded point x_e is accepted, otherwise the reflected point is accepted and the iteration will terminated.

Step 6. If the reflected point x_r is greater than the n^{th} point x_n, the algorithm starts a contraction process by calculating an outside contracted point x_{oc}, where $x_{oc} = \bar{x} + \zeta\ (x_r - \bar{x})$ or an inside contracted point x_{ic} by calculating an inside contracted point x_{ic}, where $x_{ic} = \bar{x} + \zeta\ (x_{n+1} - \bar{x})$, ζ is a contraction parameter and $0 < \zeta < 1$. The selection of the outside contraction process and the inside contraction process is depending on the values of the reflected point x_r and the n^{th} point x_n.

Step 7. If the outside contracted point x_{oc} or the inside contracted point x_{ix} is greater than the reflected point x_r, the shrinkage process is starting by calculating the shrink point. In the shrink process, the points are evaluated and the new vertices of simplex at the next iteration will be $\dot{x}_2, \dots, \dot{x}_{n+1}$, where $\dot{x} = x_1 + \sigma\ (x_i - x_1), i = 2, \dots, n + 1,\ \sigma$ is a shrinkage parameter and $0 < \sigma < 1$.

Algorithm 1 The Nelder-Mead Algorithm

1: Let x_i denote the list of vertices in the current simplex, $i = 1, \dots, n + 1$.
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 $f(x_1) \leq f(x_2) \leq \dots \leq f(x_{n+1})$.
3: Reflection. Compute the refection 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, $\bar{x} = \sum(x_i\ /n), i = 1, \dots, n$.
IF $f(x_1) \leq f(x_r) < f(x_n)$ T**hen**
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 expansion 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 $f(x_r) - f(x_{n)}$ **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 $x_{oc} = \bar{x} + \zeta(x_r - \bar{x})$ **Outside contract**
End IF
IF $f(x_n) \leq f(x_r) < f(x_{n+1})$ **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 $x_{ic} = \bar{x} + \zeta(x_{n+1} - \bar{x})$ **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, $\acute{x} = x_1 + \sigma(x_i - x_1), \ \ i = 2, \dots, n+1$.
Replace the vertices $x_2, \dots, x_{n+1}$ with the new vertices $x_2^{'}, \dots, x_{n+1}^{'}$.
7: Stopping Condition. Order and re-label the vertices of the new simplex as $x_1, x_2, \dots, x_{n+1}$ such that $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$
IF $f(x_{n+1}) - f(x_1) < \epsilon$ **Then**
Stop, where $\epsilon > 0$ is a small-predetermined tolerance.
Else
Go to **Step 3**.
End IF

2.3. An Overview of the Bat Algorithm

Bat algorithm (BA) is a population based meta-heuristics algorithm developed by Xin-She Yang in 2010 [30]. BA is based on the echolocation of microbats, which use a type of sonar (echolocation) to detect prey and avoid obstacles in the dark. The main advantage of the BA is that it can provide fast convergence at an initial stage by switching from exploration to exploitation process. However, switching from exploration to exploitation quickly may lead to stagnation after some initial stage. The main characteristics and steps of the bat algorithm are presented in Algorithm 2 and can be summarized as follow.

2.3.1. Velocity and Movement of Virtual Bats

In simulation, at iteration t, each bat in the population moves randomly with a velocity v_i^t and a position x_i^t. The position of each bat (solution) is evaluated by calculating its fitness function value $f(x_i)$ and the overall best solution x^* is assigned according to this value. The position x_i and the velocity v_i for each solution in the population are adjusted as shown in Equations 2, 3, 4. Bat algorithm is considered as a frequency-tuning algorithm, since each bat is randomly assigned a frequency$f, f \in [\, f_{min}, f_{max}\,]$. The frequency parameter is very important to balance between the exploration and the exploitation processes in bat algorithm.

In the initial population, each bat is randomly assigned a frequency then these values are adjusted as shown in Equation 2.

$$f_i = f_{min} + (f_{max} - f_{\min}) \beta \quad (2)$$

Where $\beta \in [0,1]$ is a random vector drawn from a uniform distribution. The initial velocity and position of each bat in the population are assigned randomly then they are updated as shown in Equations 3, 4, respectively.

$$v_i^t = v_i^{(t-1)} + (x_i^{(t-1)} - x^*) f_i \quad (3)$$

Where x^* is the best solution in the population.

$$x_i^t = x_i^{(t-1)} + v_i^t \quad (4)$$

2.3.2. Loudness and Pulse Emission Rate

The loudness parameter A_i and the pulse emission rate parameter r_i are very important parameters in a bat algorithm, since they control and switch between exploration and exploitation process during the search. The loudness decreases once a bat has found its prey, while the pulse emission rate increases. The values of loudness have to vary between A_{max} and A_{min}, when $A_{min} = 0$ means that a bat has found the prey. The value of loudness can be updated during the search as follow.

$$A_i^{(t+1)} = \propto A_i^t \quad (5)$$

Where $\propto$ is a constant and has the same effect as the cooling factor in simulated annealing algorithm. In addition, the pulse emission rate parameter can be updated as follow.

$$r_i^t = r_i^0 [1 - \exp(-\gamma t)] \quad (6)$$

Where γ is a constant and $\gamma > 0$.

2.3.3 Local Search Method

Each bat (solution) in the population is evaluated by calculating its fitness function value and the overall best solution is selected as a current best solution x^*. Once the best solution is selected, each bat in the population generates a new solution by using a random walk method as follow.

$$x_{new} = x_{old} + \epsilon A^t \quad (7)$$

Where ϵ is a random number and $\epsilon \in [-1,1]$, $A^t = < A_i^t >$ is the average loudness of all the bats at the current iteration.

2.3.4. Bat Algorithm

The main steps of the bat algorithm are presented in Algorithm 2 and the description of it can be summarized as follow.

Algorithm 2 Bat Algorithm

1: Set the initial values of the minimum frequency f_{min}, maximum frequency f_{max}, population size PS, the loudness constant α, the rate of pulse emission constant γ, the initial loudness A_0, the minimum loudness A_{min}, the initial rate of pulse emission r_0 and the maximum number or iterations Max_{itr}.
2: Set $t = 0$.
3: **For** $(i = 1; i < PS; i++)$ **do**
4: Generate the initial bat population x_i^t randomly.
5: Generate the initial bat velocities v_i^t randomly.
6: Assign the initial frequency f_i to each x_i^t.
7: Evaluate the initial population by calculating the objective function $f(x_i^t)$ for each solution in the Population.
8: Set the initial values of the pulse rates r_i and loudness A_i.
9: **End For**
10:**Repeat**
11: $t = t + 1$.
12: Generate new bat solutions x_i^t by adjusting frequency as shown in **Equation 4**.
13: Update the bat velocities v_i^t as shown in **Equations 2, 3**.
14: Evaluate the new population by calculating the objective function $f(x_i^t)$ for each solution in the population
15: Select the best solution x^* from the population.
16: **IF** $rand > r_i$ **Then**
17: Select a solution among the best solutions
18: Generate a local search solution around the selected best solution as shown in **Equation 7.**
19: **End IF**
20: Generate a random new solution
21: **IF** $rand < A_i$ & $f((x_i^t) < f(x^*))$ **Then**
22: Accept the new solutions.
23: Increase the rate of pulse emission r_i and reduce the loudness A_0 as shown in **Equations 5, 6**.
24: **End IF**
25: Evaluate the new population by calculating the objective function $f(x_i^t)$ for each solution in the population.
26: Rank the population and select the best solution x^* from the population.
27: **Until** $(t < Max_{itr})$
28: Produce the best solution.

Step 1. The algorithm starts by setting the initial values of its parameters and the main iteration counter is set to zero **(lines 1-2)**.

Step 2. The initial population is generated randomly by generating the initial position x_0 and the initial velocity v_0 for each bat (solution) in the population, the initial frequency f_i is assigned to each solution in the population, where f is generated randomly from $[f_{min}, f_{max}]$. The initial population is evaluated by calculating the objective function for each solution in the initial population $f(x_i^0)$ and the values of pulse rate r_i and loudness A_i is initialized, where $r \in [0,1]$ and A_i varies from a large A_{max} to A_{min} **(lines 3-9)**.

Step 3. The new population is generated by adjusting the position x_i and the velocity v_i for each solution in the population as shown in Equations 2, 3, 4, where $\beta \in [0,1]$ is a random vector drawn from a uniform distribution **(lines 12-13)**.

Step 4. The new population is evaluated by calculating the objective function for each solution and the best solution x^* is selected from the population **(lines 14-15)**.

Step 5. The local search method is applied by using a random walk method as show in Equation 7 in order to refine the best-found solution at each iteration **(lines 16-19)**.

Step 6. The new solution is accepted with some proximity depending on parameter A_i, the rate of pulse emission increases and the loudness decreases. The values of A_i and r_i are updated as shown in Equations 5 and 6.

Step 7. The new population is evaluated and the best solution is selected from the population. The operations are repeated until termination criteria satisfied and the overall best solution is produced **(lines 25-28)**.

3. Accelerated Bat Algorithm (ABATA)

In the proposed algorithm, we combine the bat algorithm, which has a good capability of exploring the search space and the Nelder-Mead method, which is one of the most important direct search method and has a powerful performance as a local search method. The proposed algorithm is called Accelerated BAT Algorithm (ABATA). ABATA starts with an initial population, which is generated randomly and consists of PS bats (solutions). These solutions are updated by moving randomly with a velocity v_i^t and a position x_i^t. Each solution is evaluated by calculating its fitness function and the best solution is selected from the population according to its fitness function value. ABATA uses a Nelder-Mead method as a local search instead of a random walk method, which is applied in the standard bat algorithm in order to refine the best solution found so far at each iteration. The structure of ABATA is shown in Algorithm 3 and more details of ABATA structure are given bellow.

3.1. ABATA Algorithm

The formal detailed description of ABATA is given in the Algorithm 3 and we can summarize the main steps of it as follow.

Algorithm 3 ABATA Algorithm

1: Set the initial values of the minimum frequency f_{min}, maximum frequency f_{max}, population size PS, the loudness constant α, the rate of pulse emission constant γ, the initial loudness A_0, the minimum loudness A_{min}, the initial rate of pulse emission r_0 and the maximum number or iterations Max_{itr}.
2: Set $t = 0$.
3: **For** $(i = 1; i < PS; i++)$ **do**
4: Generate the initial bat population x_i^t randomly.
5: Generate the initial bat velocities v_i^t randomly.
6: Assign the initial frequency f_i to each x_i^t.
7: Evaluate the initial population by calculating the objective function $f(x_i^t)$ for each solution in the Population.
8: Set the initial values of the pulse rates r_i and loudness A_i.
9: **End For**
10:**Repeat**
11: $t = t + 1$.
12: Generate new bat solutions x_i^t by adjusting frequency as shown in **Equation 4**.
13: Update the bat velocities v_i^t as shown in **Equations 2, 3**.
14: Evaluate the new population by calculating the objective function $f(x_i^t)$ for each solution in the population
15: Select the best solution x^* from the population.
16: **IF** $rand > r_i$ **Then**
17: Select a solution among the best solutions
18: Refine the best obtained solution by using **Nelder-Mead method** as shown in **Algorithm 1.**
19: **End IF**
20: Generate a random new solution
21: **IF** $rand < A_i$ & $f((x_i^t) < f(x^*))$ **Then**
22: Accept the new solutions.
23: Increase the rate of pulse emission r_i and reduce the loudness A_0 as shown in **Equations 5, 6**.
24: **End IF**
25: Evaluate the new population by calculating the objective function $f(x_i^t)$ for each solution in the population
26: Rank the population and select the best solution x^* from the population
27: **Until** $(t < Max_{itr})$
28: Produce the best solution.

Step 1. The parameters of the minimum frequency f_{min}, maximum frequency f_{max}, population size PS, the loudness constant A_{min}, the rate of pulse emission constant , the initial loudness A_0, the minimum loudness A_{min}, the initial rate of pulse emission r_0, the maximum number or iterations Max_{itr} and the initial iteration counter are set to their initial values **(lines 1-2)**.

Step 2. The initial population is generated randomly by generating the initial position x_0 and the initial velocity v_0 for each bat (solution) in the population, the initial frequency f_i^0 is assigned to each solution in the population. The initial population is evaluated by calculating the objective function for each solution in the initial population $f(x_i^0)$ and the values of pulse rate r_i and loudness A_i is initialized **(lines 3-9)**.

Step 3. The new population is generated by adjusting the position x_i and the velocity v_i for each solution in the population as shown in Equations 2, 3, 4 **(lines 12-13)**.

Step 4. The new population is evaluated by calculating the objective function for each solution and the best solution x^* is selected from the population **(lines 14-15)**.

Step 5. The Nelder Mead method is applied as shown in Algorithm 1 in order to refine the best-found solution at each iteration **(lines 16-19)**.

Step 6. The new solution is accepted with some proximity depending on parameter A_i, the rate of pulse emission increases and the loudness is decreased as shown in Equations 5 and 6. **(lines 21-24)**.

Step 7. The new population is evaluated and the best solution is selected from the population. The operations are repeated until termination criteria satisfied. Finally, the overall best solution is produced **(lines 25-28)**.

In order to investigate the correctness of the proposed algorithm, we test it on seven benchmark functions with **known optimal values** as shown in Table 2. The proposed algorithm terminates the search when it reaches to the optimal value or near optimal value as shown in Subsection 4.1. In the following section, we test the general performance of the proposed algorithm and compare it with other algorithms to verify from its correctness.

4. Numerical Experiments

The performance of ABATA is tested on seven benchmark functions, which are reported in Subsection 4.2 and their properties are reported in Table 2. ABATA was programmed in MATLAB and the results of it are averaged over 50 runs. Before discussing the results, we present the parameter tuning and performance analysis of ABATA as follows.

4.1. Parameter Setting

Before discussing the results, we summarize the setting of ABATA parameters and there values as shown in Table 1.

Table 1. Parameter setting.

Parameters	Definitions	Values
PS	population size	20
f_{min}	minimum frequency	0
f_{max}	maximum frequency	5
A^0	initial loudness	1
r^0	initial pulse emission rate	0.5
α	loudness constant	0.95
γ	rate of pulse emission constant	0.9
ρ	reflection parameter	1
χ	expansion parameter	2
ζ	contraction parameter	0.5
σ	shrinkage parameter	0.5
Max_{itr}	maximum number of iteration	1000
Max_{it}	no. of maximum iterations used in NM method	$100\ d$
N_{elite}	no. of best solution for local search	1

These values are based on the common setting in the literature or determined through our preliminary numerical experiments. These parameters are categorized in the following groups.

- **Population size parameter**. The experimental tests show that the best population size is $PS = 20$, increasing this number will increase the value of the evaluation function without any improvement in the obtained results.
- **Frequency parameters**. ABATA can be considered as a frequency-tuning algorithm. Each bat (solution) is assigned a random frequency f, where $f \in [f_{min}, f_{max}]$. The experimental results show that the best values of f_{min} and f_{max} are set to 0, 5, respectively.
- **Loudness and pulse emission rate parameters.** The values of loudness parameter A and pulse emission rate r are very important to control the wide exploration and deep exploitation process. The experimental results show that the best initial values for the parameters A^0 and r^0 are set to 1 and 0.5, respectively. The values of the parameters A and r are updated as shown in Equations 5, 6. The parameter α is similar to the cooling factor in simulated annealing. The best values of the parameters α and γ are set to 0.95 and 0.9, respectively.
- **Local search parameters.** ABATA uses a Nelder-Mead method as a local search method starting from N_{elite} best solutions, we set $N_{elite} = 1$. The experimental results show that the best parameter values of the Nelder-Mead method are $\rho = 1$, $\chi = 2$, $\zeta = 0.5$, $\sigma = 0.5$ and the maximum number of iterations used in the Nelder-Mead method is $100\ d$, where d is a problem dimension.
- **Termination parameters**. The main termination criterion in ABATA is that the maximum number of iterations is set to 1000. However, there are other two termination criteria in order to make a fair comparison with other algorithms. The first termination criterion is if $|f_{best} - f_{min}| < \varepsilon$, where f_{best} and f_{min} represent the best solution found by the algorithm and the global minimum, respectively, the value of ε is set to 10^{-4}. The second termination criterion is that the maximum evaluation function value is set to 20,000.

4.2. Integer Programming Optimization Test Problems

ABATA is tested on seven benchmark integer-programming problems ($f_1 - f_7$) and compared against different algorithms in order to investigate its performance when applied to solve integer-programming problems. The properties of the benchmark functions (function number, dimension of the problem, problem bound and the global optimal of each problem) are listed in Table 2 and the functions with their definitions are reported as follows.

Test problem 1 [22] This problem is defined by

$$f_1(x) = \|x\|_1 = |x_1| + \cdots + |x_d|,$$

Test problem 2 [22] This problem is defined by

$$f_2(x) = x^T x = [x_1 \ \ldots \ x_d] \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix},$$

Test problem 3 [6] This problem is defined by

$$f_3(x) = -[15 \ \ 27 \ \ 36 \ \ 18 \ \ 12]x + x^T \begin{bmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{bmatrix} x,$$

Test problem 4 [6] This problem is defined by

$$f_4(x) = (9x_1^2 + 2x_2^2 - 10)^2 + (3x_1 + 4x_2^2 - 7)^2,$$

Test problem 5 [6] This problem is defined by

$$f_5(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,$$

Test problem 6 [21] This problem is defined by

$$f_6(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2,$$

Test problem 7 [6] This problem is defined by

$$f_7(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2,$$

Table 2. The properties of the Integer programming test functions.

Function	Dimension	Bound	Optimal
f_1	5	$[-100 \ 100]$	0
f_2	5	$[-100 \ 100]$	0
f_3	5	$[-100 \ 100]$	-737
f_4	2	$[-100 \ 100]$	0
f_5	4	$[-100 \ 100]$	0
f_6	2	$[-100 \ 100]$	-6
f_7	2	$[-100 \ 100]$	-3833.12

4.3. The General Performance of ABATA with Integer Programming Problems

The first test to investigate the general performance of the proposed algorithm with the integer programming problems has been applied by plotting the values of function values versus the number of iterations as shown in Figure 1 for three-test functions f_3, f_4, f_7. The results in Figure 1 show that the function values of ABATA are rapidly decreases as the number of iterations increases and the hybridization between the bat algorithm and the Nelder-Mead method can accelerate the search and help the algorithm to obtain the optimal or near optimal solution in reasonable time.

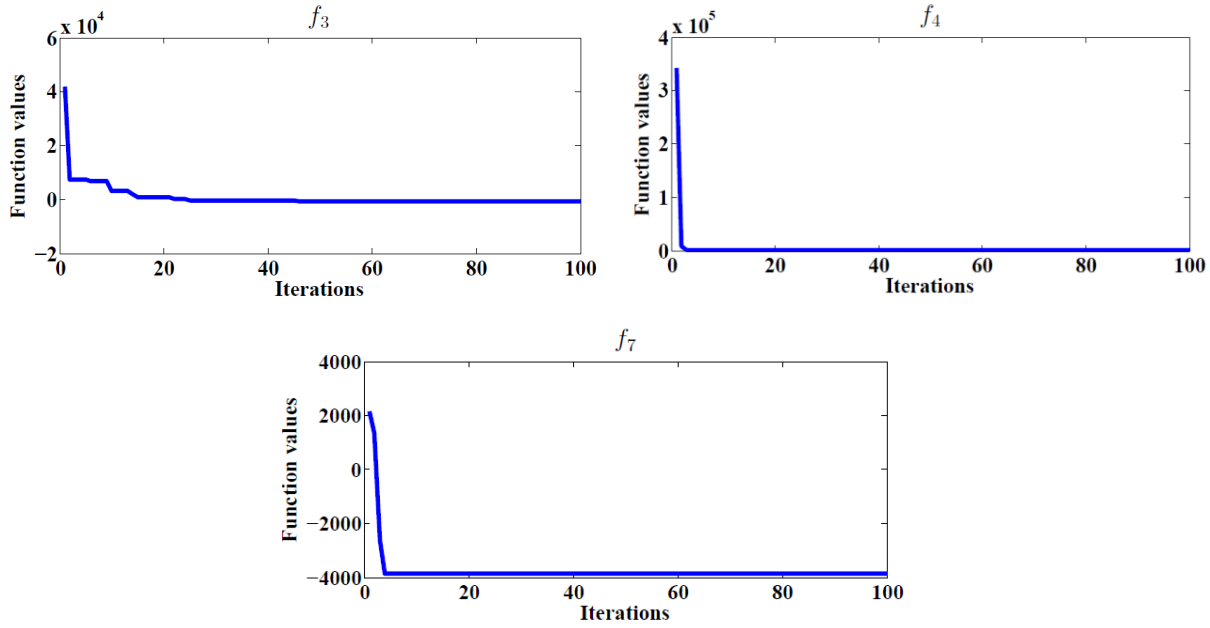

Fig. 1. The general performance of ABATA with integer programming problems.

4.4. The Efficiency of ABATA with Integer Programming Problems

The second test in order to verify the efficiency of ABATA, has been applied by comparing the proposed ABATA against the standard BA using the same BA parameters and the same termination criteria. The results are shown in Figure 2. The solid line refers to ABATA results, while the dotted line refers to the standard BA results. The results in Figure 2 represent the general performance of ABATA and the standard BA on four functions f_1, f_2, f_5, f_6 by plotting the values of function values versus the number of iterations. Figure 2 shows that the function values are rapidly decrease as the number of iterations increases for ABATA results than those of the standard BA. We can conclude from Figures 1, 2 that the combination between the bat algorithm and the Nelder-Mead method is effective and accelerate the convergence of the proposed algorithm.

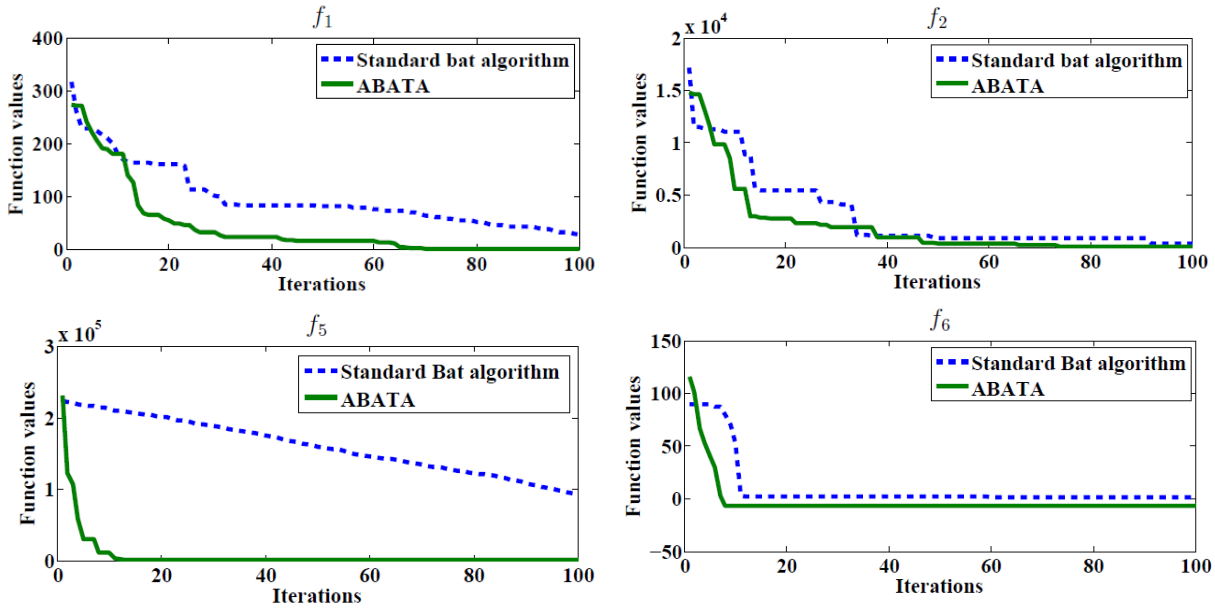

Fig. 2. The efficiency of ABATA with integer programming problems.

4.5. ABATA and Other Algorithms

ABATA is compared against four benchmark algorithms (particle swarm optimization with different variants) in order to verify of the efficiency of the proposed algorithm. Before discussing the comparison results of all algorithms, we present a brief description about the comparative four algorithms [20] as follows.

- **RWMPSOg.** RWMPSOg is a RandomWalk Memetic Particle Swarm Optimization (with global variant), which combines the particle swarm optimization with random walk with direction exploitation by applying the random walk method in the overall best global solution.
- **RWMPSOl.** RWMPSOl is a Random Walk Memetic Particle Swarm Optimization (with local variant), which combines the particle swarm optimization with random walk with direction exploitation by applying the random walk method in the local best solution.
- **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.

4.5.1. Comparison between RWMPSOg, RWMPSOl, PSOg, PSOl and ABATA.

In this subsection, we present the comparison results between ABATA and the other algorithms. The four comparative algorithms are tested on seven benchmark functions, which are reported in Subsection 4.2. The results of the comparative algorithms are taken from their original papers [20]. The minimum (min), maximum (max), average (Mean), standard deviation (St.D) and Success rate (%Suc) of the evaluation function values are reported over 50 runs and reported in Table 3. The run is considered success if the algorithm reached to the

global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. The best results between the comparative algorithms are reported with boldface text. The results in Table 3 show that ABATA is successes in all runs and obtains the objective value of each function faster than the other algorithms.

Table 3. Experimental results of function evaluation for $f_1 - f_7$ test problems

Function	Algorithm	Min	Max	Mean	St.D	Suc
f_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
	ABATA	1005	1225	**1073.75**	**102.17**	50
f_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
	ABATA	425	645	**511.25**	**98.773**	50
f_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
	ABATA	202	4512	**644.7**	**1358.85**	50
f_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
	ABATA	104	150	**123.5**	**19.63**	50
f_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
	ABATA	656	1546	**1059.25**	**381.02**	50
f_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
	ABATA	80	125	**101.25**	**20.15**	50
f_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
	ABATA	80	425	**187.5**	159.60	50

5. Conclusion

In this paper, a new hybrid algorithm has been proposed in order to solve integer-programming problems by combining the bat algorithm with the Nelder-Mead method. The proposed algorithm is called accelerated bat algorithm (ABATA). ABATA has a good ability to perform a wide exploration and a deep exploitation. Invoking the Nelder-Mead method as a local search method in ABATA accelerate the search by refining the best obtained solution at each iteration. ABATA has been intensely tested on seven integer-programming problems and compared against other 4 algorithms in order to test its performance for solving integer programming problems. The numerical results indicate that the proposed ABATA is a promising algorithm and suitable to find a global optimal solution or near optimal solution in reasonable time.

References

[1]. 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.

[2]. N. Bacanin, 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.

[3]. 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), 4099 LNAI:854-858, 2006.

[4]. M. Dorigo, "Optimization, Learning and Natural Algorithms", Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[5]. A. R. Fletcher, "Practical method of optimization", Vol.1 & 2, John Wiley and Sons, 1980.

[6]. Glankwahmdee, J.S. Liebman and G.L. Hogg, "Unconstrained discrete nonlinear programming", Engineering Optimization, 4, 95-107, 1979.

[7]. F.S. Hillier and G. J. Lieberman, "Introduction to operations research", MCGraw-Hill, 1995.

[8]. R. Jovanovic, 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. 53605366, 2011.

[9]. R. Jovanovic, 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.

[10]. D. Karaboga and B. Basturk, "A powerful and e_cient algorithm for numerical function optimization", artificial bee colony (abc) algorithm. Journal of global optimization, 39(3):459-471, 2007.

[11]. J. Kennedy, RC. Eberhart, "Particle Swarm Optimization", Proceedings of the IEEE International Conference on Neural Networks, Vol 4, pp 19421948, 1995.

[12]. E.C. Laskari, K.E. Parsopoulos, M.N. Vrahatis, "Particle Swarm Optimization for Integer Programming", Proceedings of the IEEE 2002 Congress on Evolutionary Computation, Honolulu (HI), pp. 1582-1587, 2002.

[13]. E.L. Lawler and D.W. Wood, "Branch and Bound Methods", A Survey, Operations Research, Vol. 14, pp. 699-719, 1966.

[14]. 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(11):32, 2002.

[15]. 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.

[16]. JA. Nelder and R. Mead, "A simplex methods for function minimization", Computer journal 7:308-313, 1965.

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

[18]. 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, 53-59, 2005.

[19]. M.K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control", Control Systems, IEEE 22(3):52-67, 2002.

[20]. Y.G. Petalas, K.E. Parsopoulos. M.N. Vrahatis, "Memetic particle swarm optimization", Ann oper Res, 156:99-127, 2007.

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

[22]. 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 3, 1994.

[23]. 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, pages 165-172, 2012.

[24]. D. Teodorovic and M. DellOrco, "Bee colony optimization a 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, pages 51-60, 2005.

[25]. M. Tuba, N. Bacanin, N. Stanarevic, "Adjusted artificial bee colony (ABC) algorithm for engineering problems", WSEAS Transaction on Computers, Volume 11, Issue 4, pp. 111-120, 2012.

[26]. M. Tuba, M. Subotic, N. Stanarevic, "Performance of a modified cuckoo search algorithm for unconstrained optimization problems", WSEAS Transactions on Systems, Volume 11, Issue 2, pp. 62-74, 2012.

[27]. N. Bacanin, I. Brajevic, M. Tuba, "Firefly Algorithm Applied to Integer Programming Problems", Recent Advances in Mathematics, 2013.

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

[29]. X.S. Yang and S. Deb, "Cuckoo search via levy flights", In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pages 210-214. IEEE, 2009.

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

[31]. S. Zuhe, A. Neumaier and M.C. Eiermann, "Solving Minimax problems by Interval Methods", BIT, Vol. 30, PP. 742-751, 1990.