Ali and Tawhid. Trends Artif Intell 2018, 2(1):46-59
DOI: 10.36959/643/303 | Volume 2 | Issue 1

Trends in Artificial Intelligence

Original Article Open Access

Solving Integer Programming Problems by Hybrid Bat Algorithm and Direct Search Method

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

[1]Faculty of Computers & Informatics, Department of Computer Science, Suez Canal University, Egypt

[2]Faculty of Science, Department of Mathematics and Statistics, Thompson Rivers University, Canada

[3]Faculty of Science, Department of Mathematics and Computer Science, Alexandria University, Egypt

Abstract

The goal of this work is to suggest a new hybrid algorithm to solve integer programming by incorporating the bat algorithm with direct search methods. The suggested algorithm is named hybrid bat direct search algorithm (HBDS). In HBDS, the global diversification and the local intensification process are balanced. The bat algorithm has a good capability to make intensification and diversification search. The intensification ability of the suggested algorithm is increased by employing the pattern search method as a local search method instead of the random walk method in the classic bat algorithm. In the final stage of the algorithm, the Nelder-Mead method is used to improve the best found solution from the bat and pattern search method instead of running the algorithm more iterations without any enhancements in the fitness function value. The performance of the HBDS algorithm is examined on 7 integer programming problems and compared to 10 benchmark algorithms for solving integer programming problems. The computational results show that HBDS is a promising algorithm and outperforms the other algorithms in most cases.

Keywords

Bat algorithm, Direct search methods, Pattern search, Nelder-Mead method, Integer programming problems

Introduction

Integer programming (IP) appear closely in every research area in applied operations research and mathematical programming. Variety of many real life applications for IP problems such as, scheduling problem, VLSI (very large scale integration) circuits design problems, engineering design problems, warehouse location problem, robot path planning problems, [1-3] can be formulated as IP problems.

On one hand, traditional integer programming methods such as dynamic programming or branch and bound have high computational cost, because they examine a search tree that has hundreds or more nodes when large scale real-life problems are considered. On the other hand, heuristic and metaheuritsic methods can be applied for solving integer programming problems. Swarm intelligence (SI) algorithms are novel meta-heuristics algorithms, which find their inspiration from the behavior of a group of social organisms. These algorithms are used to solve global optimization problems and their applications such as ant colony optimization (ACO) [4,5] artificial bee colony [6] particle swarm optimization (PSO) [7-9], bacterial foraging [10], bat algorithm [11,12], bee colony optimization (BCO) [13], wolf search [14], cat swarm [15], Cuckoo search [16], firefly algorithm [17], fish swarm/school [18], etc.

These algorithms have been commonly adopted to

***Corresponding author:** Mohamed A Tawhid, Faculty of Science, Department of Mathematics and Statistics, Thompson Rivers University, Kam-loops, BC V2C 0C8, Canada; Faculty of Science, Department of Mathematics and Computer Science, Alexandria University, Mo-haram Bey 21511, Alexandria, Egypt, Tel: 250-377-6041, Fax: 250-371-5675, E-mail: Mtawhid@tru.ca

Accepted: July 23, 2018; **Published online:** July 25, 2018

Citation: Ali AF, Tawhid MA (2018) Solving Integer Programming Problems by Hybrid Bat Algorithm and Direct Search Method. Trends Artif Intell 2(1):46-59.

solve unconstrained and constrained optimization problems and their applications, nevertheless there are not many SI algorithms used to solve integer programming problems [19-22]. There are some efforts to use some of meta-heuristics algorithms to solve integer programming problems such as social spider optimization [23], gravitational search algorithm [20], particle swarm optimization algorithm [7,24], firefly algorithm [25,26], cuckoo search algorithm [27-29], artificial bee colony algorithm [30-32], ant colony algorithm [33], and simulated annealing [19].

Bat algorithm (BA) is a recent population based algorithm inspired from the echolocation behavior of the microbats [12]. BA is capable to balance the global diversification and the local intensification during the search process. Because of the powerful performance of the BA, it has been used by many researchers to solve diverse applications, for example, Lin, et al. [34] used parameter estimation in dynamic biological systems using a chaotic bat algorithm by integrating Levy flights and chaotic maps. Zhang and Wang [35] improved the diversity of solutions by using the mutation with bat algorithm for image matching. Yang [36] applied BA to solve multi-objective optimization and benchmark engineering problems. Komarasamy and Wahi [37] integrated K-means and bat algorithm (KMBA) for efficient clustering. Nakamura, et al. [38] developed a discrete version of bat algorithm to solve classifications and feature selection problems. Xie, et al. [39] presented a variant of bat algorithm integrating Levy flights and differential operator to solve function optimization problems. In addition, Wang and Guo [40] integrated harmony search with bat algorithm and generated a hybrid bat algorithm for numerical optimization of function benchmarks.

The purpose of this paper is to avoid the slow convergence of the BA and avoid trapping in local minima. In order to solve these two issues, we suggest a new hybrid bat algorithm with direct search methods to solve integer programming problems [41]. The suggested algorithm is named hybrid bat direct search algorithm (HBDS). In HBDS, the pattern search is employed as a local search method to exploit the search around the best found solution at each iteration and in the final stage of the algorithm, the Nelder-Mead method is called to enhance the best obtained solution from the bat and pattern search method. Using the Nelder-Mead method can hasten the search and evade running the algorithm with more iterations without any enhancements in the results.

The rest of this paper is structured as follows. In Section 4, the integer programming problems and the applied direct search methods, the classic BA are described. In Section 5, the main concepts of the suggested HBDS algorithm are given. The numerical experimental and results are shown in Section 6. Finally, the conclusion and future work are presented in Section 7.

Definition of the Problems and an Overview of the Applied Algorithms

In this section and its subsections, the definitions of the integer programming problems are presented and an overview of the BA and the pattern search method is given as follows.

The integer programming problem definition

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 follows.

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

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

Pattern search method

The authors in [42] introduced the pattern search method (PS). Pattern search method is an applied direct search method to solve a global optimization problems. In direct search method, there is no need for any information about the gradient of the objective function to solve optimization problem. PS method has two type of moves, the exploratory moves and the pattern moves. In the exploratory moves a coordinate search is used around a chosen solution with a step length of Δ in Algorithm 1. The exploratory move is considered successful if the function value of the new solution is better than the current solution. Otherwise, the step length is reduced. If the exploratory move is successful, then the pattern search is used in order to produce the iterate solution. If the iterate solution is better than the current solution, then exploratory move is used 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 decreased. The operation is repeated until stopping criteria are satisfied. The algorithm of Hook and Jeeves (HJ) pattern search and the main steps of it are outlined in Algorithm 2. The parameters in Algorithms 1, 2 are outlined in Table 1.

A Nelder-Mead method

The main steps of the Nelder-Mead (NM) algorithm

Table 1: The parameters of the pattern search algorithm.

Parameter	Definition
Δ_0	Initial mesh size
d	Variable mesh size
σ	Reduction factor of mesh size
m	Pattern search repetition number
ε	Tolerance

[43] in two dimensions is presented as follows. There are four scalar parameters that represent the main parameters of the NM algorithm such as: Coefficients of reflection ρ, expansion χ, contraction τ, and shrinkage Table 1.

Algorithm 1: Exploratory search

INPUT: Get the values of x^0, k, Δ_0, d

OUTPUT: New base point x^k

1. Set $i = 1$
2. Set $k = 1$
3. **repeat**
4. Set $x_i^k = x_i^{k-1} + \Delta_{k-1}\, x_i^{k-1}$
5. **if** $f(x_i^k) < f\left(x_i^{k-1}\right)$ **then**
6. Set $x_i^{k+1} = x_i^k$
7. **end if**
8. Set $i = i + 1$
9. Set $k = k + 1$
10. **until** $i > d$

ϕ where $\rho > 0$, $\chi > 1$, $0 < \tau < 1$, and $0 < \phi < 1$. The main steps of the NM algorithm are shown in Algorithm 3.

Overview of the Bat algorithm

In this section, an overview of the main concepts and structure of the BA is given.

Main concepts: Bat algorithm (BA) is a population based metaheuristic algorithm, was developed by Xin-She Yang in 2010 [12]. 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 a fast convergence at a very initial stage by switching from diversification to intensification, however, switching from diversification to intensification quickly may lead to stagnation after some initial stage.

The rules of the BA: Based on the bat characteristics, Xin-She Yang developed the bat algorithm with the following rules.

- All bats can distinguish between pray and barriers/obstacles by using echolocation to sense distance.

Algorithm 2: The basic pattern search algorithm.

INPUT: Get the values of x.

OUTPUT: Best solution x^*.

1. Set the values of the initial values of the mesh size Δ_0, reduction factor of mesh size σ and termination parameter ε
2. Set $k = 1$ {**Parameter setting**}
3. Set the starting base point x^{k-1} {**Initial solution**}
4. **repeat**
5. Perform exploratory search as shown in Algorithm 1
6. **if** exploratory move success **then**
7. Go to 16
8. **else**
9. if $\|\Delta\| <$, **then**
10. Stop the search and the current point is x^*
11. **else**
12. Set $\Delta_k = \sigma\, \Delta_{k-1}$ {**Incremental change reduction**}
13. Go to 5
14. **end if**
15. **end if**
16. Perform pattern move, where $x_p^{k+1} = x^k + \left(x^k - x^{k-1}\right)$
17. Perform exploratory move with x_p as the base point
18. Set x^{k+1} equal to the output result exploratory move
19. **if** $f(x_p^{k+1}) < f(x^k)$ **then**
20. Set $x^{k-1} = x^k$
21. Set $x^k = x^{k+1}$ {**New base point**}
22. Go to 16
23. **else**
24. Go to 9 {**The pattern move fails**}
25. **end if**
26. Set $k = k + 1$
27. **until** $k > m$

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 $f(x_1) \le f(x_2) \le \ldots \le f(x_{n+1})$.

3. Reflection. Compute the reflected point x_r by

$x_r = \bar{x} + \rho\left(\bar{x} - x_{(n+1)}\right)$, where $\bar{x}$ is the centroid of the n best points,

$\bar{x} = \Sigma(x_i / n)$, $i = 1, \ldots, n$.

if $f(x_1) \le f(x_r) < f(x_n)$ **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 $f(x_n) \le f(x_r) < f(x_{n+1})$ **then**

Calculate $x_{oc} = \bar{x} + \tau\left(x_r - \bar{x}\right)$ {Outside contract}

end if

if $f(x_{oc}) \le f(x_r)$ **then**

Replace x_{n+1} with x_{oc} and go to Step 7.

else

Go to Step 6.

end if

if $f(x_r) \ge f(x_{n+1})$ **then**

Calculate $x_{ic} = \bar{x} + \tau\left(x_{n+1} - \bar{x}\right)$. {**Inside contract**}

end if

if $f(x_{ic}) \le f(x_{n+1})$ **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 $f(x_1) \le f(x_2) \le \ldots \le f(x_{n+1})$

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

- Each bat randomly moves with velocity v_i at a position x_i with a frequency f_{min} varying loudens A_0 and pulse emission rate r.
- Assume that the loudness varies from a large value A_0 to a minimum value A_{min}.

Bat movement: The BA is a population based method, where the population size consists of bats (solutions). Each bat (solution) in the population is randomly moving with velocity v_i and a location x_i. Also each bat is randomly assigned a frequency drawn uniformly from $[f_{min}, f_{max}]$. The position of each bat in the population is updated as shown in the following equations.

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

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

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

Where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution.

Variation of loudness and pulse emission rates: The loudness A_i and the pulse rate emission r_i are very important to let the algorithm switch between diversification and intensification process. When the bat has found its pray, the loudness decreases and the rate of pulse emission increases. The BA starts with an initial value of the loudness A_0 and the rate of pulse emission r_0, then these values are updated as shown in the following equations.

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

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

Where $\alpha \in [0, 1]$ and $\gamma > 0$ are constant, the parameter plays a similar role as the cooling factor in the simulated annealing algorithm.

Bat algorithm: The main steps of the classic bat algorithm are shown in Algorithm 4 and can be summarized in the following steps.

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

Step 2: The initial population is randomly generated 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 randomly generated from $[f_{min}, f_{max}]$. The initial population is evaluated by calculating the objective function for each solution via the initial population $f(x^0)$, the values of pulse rate r_i and initial loudness A_i where r

$\in [0, 1]$ and A_i varies from a large A_0 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 in (2), (3), and (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 local search method is applied 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, increasing the rate of pulse emission and decreasing the loudness. The values of A_i and r_i are updated as in (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 are satisfied and the overall best solution is produced. (**Lines 25-28**)

The Suggested HBDS Algorithm

The main steps of the suggested HBDS algorithm are given in Algorithm 5 and the description of the suggested algorithm can be summarized as follows.

Algorithm 4: The bat algorithm

1. Set the initial values of the minimum frequency f_{min}, maximum frequency f_{max}, population size P, the loudness constant α, the rate of pulse emission constant γ, the initial loudens A_0, the minimum loudness A_{min}, the initial rate of pulse emission r^0 and the maximum number of iterations Max_{itr}. {**Parameters initialization**}

2. Set $t = 0$. {**Counter initialization**}

3. **for** ($i = 1; i < P; i{+}{+}$) **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. {**Population initialization**}

9. **end for**

10. **repeat**

11. $t = t + 1$

12. Generate new bat solutions x_i^t by adjusting frequency as in (4).

13. Update the bat velocities v_i^t as in (2) and (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.

19. **end if**

20. Generate a random new solution.

21. **if** $rand < A_i \;\&\; f\left(x_i^t < f\left(x^*\right)\right)$ **then**

22. Accept the new solutions.

23. Increase the rate of pulse emission r_i and reduce the loudness A_0.

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** ($t > Max_{itr}$) {**Termination criteria are satisfied**}

28. Produce the best solution.

Algorithm 5: The HBDS algorithm

1. Set the initial values of the minimum frequency f_{min}, the minimum loudens A_{min}, the rate of pulse emission constant γ, maximum frequency f_{max}, the initial loudens A_0, population size P, the loudness constant α, the maximum number of iterations Max_{itr}, and the initial rate of pulse emission r^0. {**Parameters initialization**}

2. Set $t = 0$. {**Counter initialization**}

3. **for** ($i = 1; i < P; i{+}{+}$) **do**

4. Generate the initial bat population x_i^t randomly.

5. Generate the initial bat velocities v_i^t randomly.

6. Designate 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. {**Population initialization**}

9. **end for**

10. **repeat**

11. $t = t + 1$.

12. Generate new bat solutions x_i^t by adjusting frequency as in (2), (3) and (4).

13. Update the bat velocities v_i^t as in (3).

14. Calculate the new population by calculating the objective function $f(x_i^t)$ for each solution in the population.

15. Choose the best solution x^* from the population.

16. **if** ($rand > r_i$) **then**

17. Choose a solution among the best solutions.

18. Use the pattern search method in Algorithm 2 on the best solution from the best solutions list. {exploitation process}

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. Reduce the loudness A_0 and increase the rate of pulse emission r_i and as in (5) and (6).

24. **end if**

25. Calculate the new population by evaluating the objective function $f(x_0^t)$ for each solution in the population.

26. Rank the population and choose the best solution x^* from the population.

27. **until** $(t > Max_{itr})$ {**Stopping criteria are satisfied.**}

28. Employ Nelder-Mead method on the best solutions, N_{elite}, as shown in Algorithm 3.{**Final intensification**}

Step 1: The parameters of the minimum frequency f_{min}, maximum frequency f_{max}, population size P, 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, the maximum number of iterations Max_{itr} and the initial iteration counter are set to their initial values. (**Lines 1-2**)

Step 2: The initial population is randomly generated 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 via the initial population $f(x_i^0)$ and the values of pulse rate r_i and initial loudness A_i. (**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 in (2), (3) and (4). (**Lines 12-13**)

Step 4: The new population is determined by evaluating the objective function for each solution and the best solution x^* is selected from the population. (**Lines 14-15**)

Step 5: The pattern search method is considered as a local search method and used in Algorithm 2 in order to improve 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, increasing the rate of pulse emission and decreasing the loudness as in (5) and (6). (**Lines 21-24**)

Step 7: The new population is calculated, and the best solution is chosen from the population. The operations are repeated until stopping criteria are satisfied.

Step 8: The Nelder-Mead method is used on the best found solution in the previous stage as a final intensification process in order to speed up the search and avoid running the algorithm with more iterations without any enhancement. (**Line 28**)

Numerical Experiments

In this section, the efficiency of the HBDS algorithm

Table 2: Parameter setting.

Parameter	Definitions	Values
P	Population size	20
f_{min}	Minimum frequency	0
f_{max}	Maximum frequency	5
A_0	The initial loudness	1
r_0	The initial pulse rate	0.5
α	The loudness constant	0.9
γ	The rate of pulse emission constant	0.9
ε	Step size for checking for decent 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
m	Local PS repetition number	5
Max_{itr}	Maximum iterations number	$2d$
N_{elite}	Number of best solution for final intensification	1

is determined by giving its general performance on various benchmark functions and comparing the results of the suggested algorithm with different algorithms. In the following subsections, the parameters setting of the suggested algorithm and the properties of the applied test functions are outlined. Also, the performance analysis of the suggested algorithm is given with the comparative results between for other algorithms (Table 2).

Parameter setting

The parameters of the HBDS algorithm have been outlined with their designated values in Table 2. Note that the parameters values are based on the common setting in the literature.

Population size P: The experimental tests show that the best population size is $P = 20$, increasing this number will not improve the results, but the evaluation function values will increase.

Frequency parameter f: Bat movement relies on the value of the frequency parameter f. In HBDS algorithm, the quality of the solution is associated to the value of f parameter. The experimental tests show that the minimum value of f is $f_{min} = 0$ and the best maximum value of f is $f_{max} = 5$.

Loudness parameters A, α: Loudness parameter A is one of the most essential parameters in the BA. The acceptance of the new generated solutions is based on the value of A. The α parameter plays a comparable role as the cooling factor in the simulated annealing algorithm. The initial value of A is set to 1 and the value of α is set to 0.9.

Pulse emission rate r: The value of the rate of pulse emission parameter r is very important to apply the local search method in the algorithm. The experimental tests show that the best value of r is 0.9 and the rate of pulse emission constant is $\gamma = 0.9$.

Pattern search parameters: HBDS uses PS as a local search method in order to enhance the best obtained solution from the BA at each iteration. In PS the mesh size is initialized as Δ_0, in our experiments $\Delta_0 = (Ui - L_i)/3$ and when no enhancement accomplished in the diversification search process, the mesh size is deducted by using reduction factor σ. The experimental results show that the best value of σ is 0.01. The PS steps are repeated m times, in order to enhance the intensification process of the algorithm. In our experiment $m = 5$ as a pattern search iteration number (Table 3).

Stopping condition parameters: HBDS stops the search when the number of iterations reaches to the desired maximum number of iterations or any other termination relying on the comparison with other algorithms. In our experiment, the value of the maximum number

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

Table 4: Integer programming optimization test problems.

Test problem	Problem definition
Problem 1 [51]	$FI_1(x) = \lVert x \rVert_1 = \lvert x_1 \rvert + \ldots + \lvert x_n \rvert$
Problem 2 [51]	$FI_2(x) = x^T x = [x_1 \ldots x_n] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$
Problem 3 [52]	$FI_3(x) = [15\,27\,36\,18\,12]x + x^T \begin{bmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -32 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{bmatrix} x$
Problem 4 [52]	$FI_4(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2$
Problem 5 [52]	$FI_5(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$
Problem 6 [53]	$FI_6(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2$
Problem 7 [52]	$FI_7(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2$

of iteration $Max_{itr} = 2d$, where d is the dimension of the problems.

Final intensification: The best obtained solutions from the BA and the pattern search method are recorded in a list in order to use the Nelder-Mead method on them, the number of the solutions in this list is called N_{elit}, in order to evade increasing the value of the function evaluation value, $N_{elit} = 1$.

Integer programming optimization test problems

The efficiency of the HBDS algorithm has been examined on 7 benchmark integer programming problems (FI_1 - FI_7). The properties of the benchmark functions (the global optimal of each problem, function number, problem bound, and dimension of the problem) are outlined in Table 3 and the functions with their definitions are presented in Table 4 as follows.

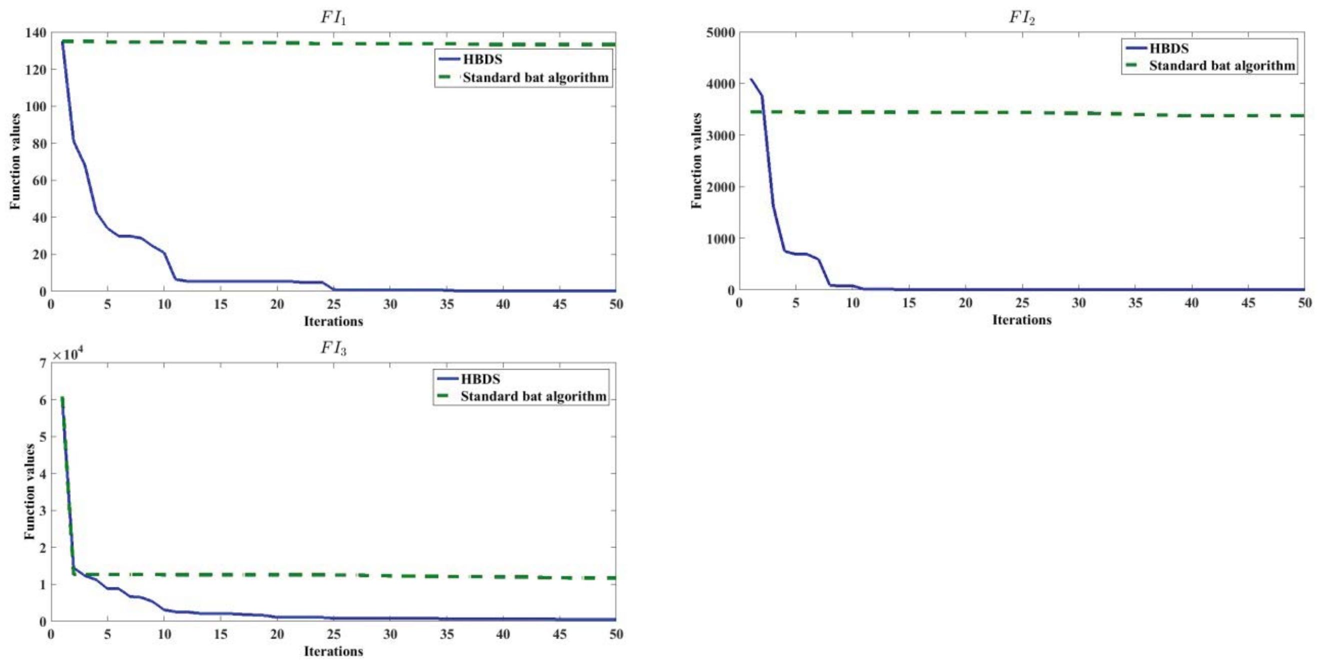

Figure 1: The efficiency of the suggested HBDS algorithm on integer programming problems.

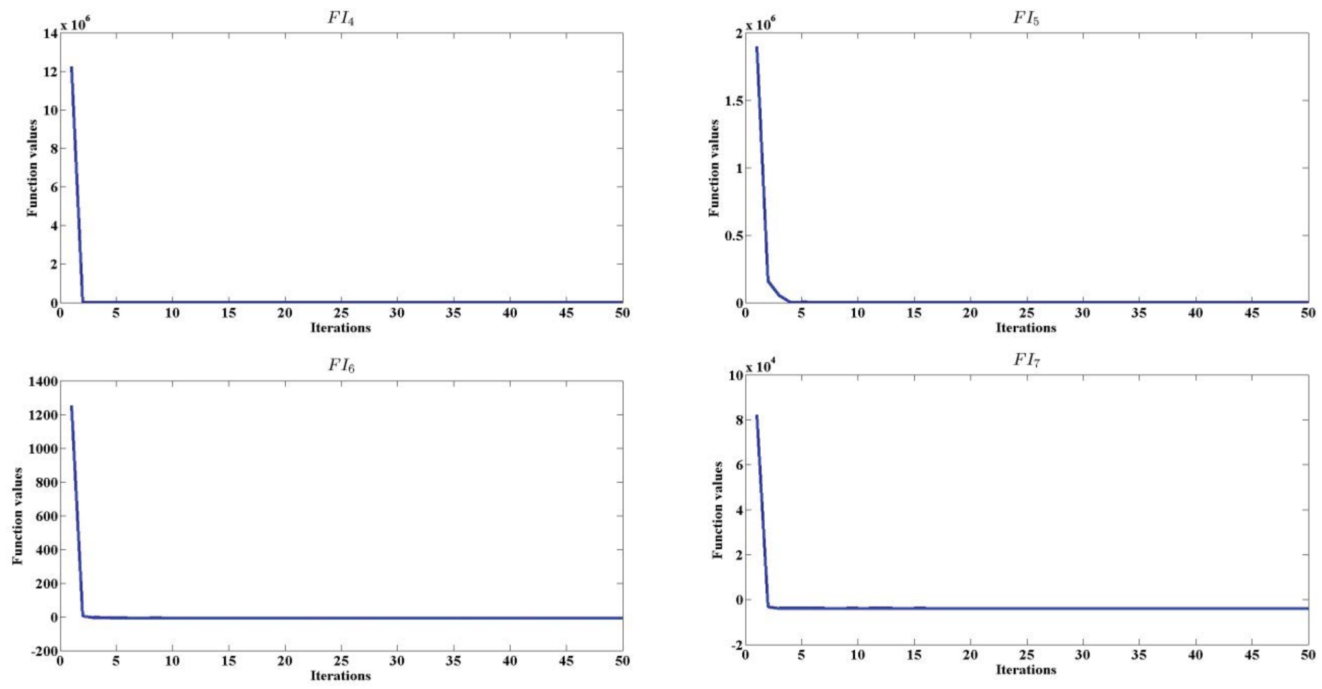

Figure 2: The performance of HBDS algorithm on integer programming problems.

Comparison of HBDS without NM and HBDS with NM

The classic BA is compared with the suggested HBDS algorithm without applying the final intensification process (Nelder-Mead method) to verify the efficiency of the suggested HBDS. The values of the parameters are designated the same for both algorithms in order to have a fair comparison. The functions FI_1, FI_2 and FI_3 have been chosen to show the efficiency of the suggested algorithm by plotting the values of function values versus the number of iterations as shown in Figure 1. In Figure 1, the solid line alludes to the suggested HBDS results, while the dotted line alludes to the classic bat results after 50 iterations. Figure 1 shows that the function values rapidly reduce as the number of iterations expands for HBDS results than those of the classic BA. From Figure 1, it can be deduced that the incorporation between the classic BA with pattern search method can enhance the performance of the classic BA and speed the convergence of the suggested algorithm (Table 4).

The general performance of the suggested algorithm on the integer programming problems has been investigated by plotting the values of function values versus the number of iterations as shown in Figure 2 for four test functions FI_4, FI_5, FI_6 and FI_7. The results in Figure 2 are the results of the suggested algorithm without applying the Nelder-Mead method in the final stage of the algorithm after 50 iterations. From Figure 2, it can be deduced that the function values of the suggested HBDS rapidly reduce as the number of iterations expands and the hybridization between the BA and the pattern search method can hasten the search and help the algorithm to get the optimal or close to optimal solution in reasonable time.

The Nelder-Mead (NM) method is used in the final stage of the suggested HBDS algorithm in order to speed the convergence of the suggested algorithm and evade running the algorithm with more iterations without any enhancement in the obtained results. The results in Table 5 show the mean evaluation function values of the suggested HBDS without and with using Nelder-Mead method, respectively. The stopping criteria mean that the algorithm reaches to the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value, are designated the same for both the algorithms. The best results are outlined in **boldface** text. In Table 5, the results show that invoking the Nelder-Mead method in the final stage can hasten the search and help the algorithm to get the optimal or near optimal solution faster than the suggested algorithm without using the Nelder-Mead method (Table 5).

Table 5: The efficiency of calling the Nelder-Mead method in the final stage of HBDS for FI_1-FI_7 integer programming problems.

Function	HBDS without NM	HBDS with NM
FI_1	1800.26	**656.56**
FI_2	780.13	**344.22**
FI_3	20,000	**1137.48**
FI_4	410.15	**260.8**
FI_5	1315.45	**1177.12**
FI_6	250.22	**149.08**
FI_7	260.15	**222.91**

HBDS and other algorithms

The HBDS algorithm is compared with four benchmark algorithms (namely, particle swarm optimization and its variants) in order to verify of the efficiency of the suggested algorithm. Before presenting the comparison results of all algorithms, a brief description about the comparative four algorithms [24] is described.

RWMPSOg: RWMPSOg is Random Walk Memetic Particle Swarm Optimization (with global variant), which incorporates the particle swarm optimization with random walk as direction exploitation.

RWMPSOl: RWMPSOl is Random Walk Memetic Particle Swarm Optimization (with local variant), which incorporates the particle swarm optimization with random walk as 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.

Comparison between RWMPSOg, RWMPSOl, PSOg, PSOl and HBDS for integer programming problems: In this subsection, the comparison results between our HBDS algorithm and the other algorithms is given in order to validate the efficiency of our HBDS algorithm. The four comparative algorithms are examined on 7 benchmark functions. The results of the comparative algorithms are considered from their original paper [24]. The minimum (min), maximum (max), average (Mean), standard deviation (St. D) and success rate (% Suc) of the evaluation function values are outlined over 50 runs in Table 6. The run is regarded successful if the algorithm gets to the global minimum of the solution within an error of 10^{-6} before the 20,000 function evaluation value. The best results between the comparative algorithms are outlined in **boldface** text. The results in Table 6, show that the suggested HBDS algorithm is successful in all runs and gets the objective value of each function faster than the other algorithms in 5 of 7 functions.

HBDS and other meta-heuristics and swarm intelligence algorithms for integer programming problems

Table 6: 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
	HBDS	382	762	**712.34**	111.12	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
	HBDS	295	412	**375.35**	61.45	50
FI_3	RWMPSOg	1361	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
	HBDS	815	1315	**1210.12**	119.48	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
	HBDS	258	311	275.22	13.22	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	50
	PSOl	800	3880	2472.4	637.5	50
	HBDS	913	1520	**1212.34**	245.38	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
	HBDS	140	175	152.18	14.25	50
FI_7	RWMPSOg	72	620	242.7	132.2	50
	RWMPSOl	70	573	248.9	134.4	50
	PSOg	100	660	421.2	134.4	50
	PSOl	100	820	466	165	50
	HBDS	215	244.15	**224.13**	14.34	50

The HBDS algorithm is investigated with various meta-heuristics and swarm intelligence algorithms (SI) such as genetic algorithm (GA) [44], standard particle swarm optimization (PSO) [8], firefly (FF) [17], cuckoo search (CS) [16] and grey wolf optimization algorithms (GWO) [45]. Having fair comparison, the population size is set 20 for all algorithms and the stopping criteria for all algorithm are the same which are the algorithm gets to the global minimum of the solution within an error of 10^{-4} before the 20,000 function evaluation value. For the GA the probability of crossover is set to $PC = 0.8$, probability of mutation $PM = 0.01$ and the roulette wheel selection is used. For the other SI algorithms, Table 6 the standard parameter setting for each algorithm is considered. The average (Avg) and standard deviation (SD) of all algorithms are outlined over 50 runs as shown in Table 7.

It can be concluded from Table 7 that the suggested algorithm can get the desired optimum values faster than the other SI algorithm.

HBATDS and the branch and bound method

In order to investigate the power of the HBATDS algorithm, it is compared with another famous method which is called branch and bound method (BB) [46-49]. Before discussing the comparative results between the proposed algorithm and BB method, the BB method and the main steps of its algorithm are presented.

Branch and bound method: The branch and bound method (BB) is one of the most widely used method for

Table 7: HBDS and other meta-heuristics and swarm intelligence algorithms for FI_1-FI_7 integer programming problems.

Function		GA	PSO	FF	CS	GWO	HBDS
FI_1	Avg SD	1640.23	20,000	1617.13	11,880.15	860.45	**656.56**
		425.18	0.00	114.77	623.41	43.66	88.65
FI_2	Avg SD	1140.15	17,540.17	834.15	7176.23	880.25	**344.22**
		345.25	1054.56	146.85	637.75	61.58	43.32
FI_3	Avg SD	4120.25	20,000	1225.17	6400.25	4940.56	**1137.56**
		650.21	0.00	128.39	819.94	246.89	85.61
FI_4	Avg SD	1020.35	16,240.36	476.16	4920.35	2840.45	**260.8**
		452.56	1484.96	31.29	247.19	152.35	10.39
FI_5	Avg	1140.75	13,120.45	1315.53	7540.38	1620.65	**1177.12**
	SD	245.78	1711.83	113.01	440.82	111.66	111.6
FI_6	Avg SD	1040.45	1340.14	345.71	4875.35	3660.25	**149.08**
		115.48	265.21	35.52	865.11	431.25	8.21
FI_7	Avg SD	1060.75	1220.46	675.48	3660.45	1120.15	**222.91**
		154.89	177.19	36.36	383.23	167.54	11.19

solving optimization problems. The main idea of BB method is the feasible region of the problem is subsequently partitioned into several sub regions, this operation is called branching. The lower and upper bounds value of the function can be determined over these partitions, this operation is called bounding. The main steps of BB method are reported in Algorithm 6, and the BB algorithm can be summarized 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 have been determined, where $\beta(M_i) \le \inf f(M_i \cap S) \le \alpha(M_i)$, f is the objective function.

Algorithm 6: The branch and bound algorithm

1. Set the feasible region M_0, $M_0 \supset S$.

2. Set $i = 0$

3. **repeat**

4. Set $i = i + 1$

5. Partition the feasible region M_0 into many subsets M_i.

6. For each subset M_i, determine lower bound β, where $\beta = \min \beta(M_i)$.

7. For each subset M_i, determine upper bound α, where $\alpha = \min \alpha(M_i)$.

8. **if** $(\alpha = \beta) \,||\, (\alpha - \beta \le)$ **then**

9. Stop

10. **else**

11. Select some of the subset M_i and partition them

12. **end if**

13. Determine new bound on the new partition elements.

14. **until** $(i \le m)$

Step 3: The algorithm is terminated, if the bounds are equal or very close, i.e. $\alpha = \beta$ (or $\alpha - \beta \le \epsilon$) 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.

Comparison between BB method and HBATDS algorithm for integer programming problems: In this subsection, the HBATDS algorithm is tested with the BB method. The results of the BB method are taken from its original paper [?]. In [?], 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. In Table 8, the comparative results between the BB method and the proposed algorithm are reported. In Table 8, the average (Mean), standard deviation (St. D) and rate of success (Suc) are reported after 30 runs. The best mean evaluation values between the two algorithms are marked in **boldface**. The results in Table 8 show that the proposed algorithm results are better than the results of the BB method, however the BB method is better than the proposed algorithm in function FI_2. The overall results in Table 8 and Figure 3 show that the proposed algorithm is faster and more efficient than the BB method.

It can be concluded from the two comparison tests between the proposed HBATDS algorithm and the 5 benchmark algorithms, that the proposed HBATDS al-

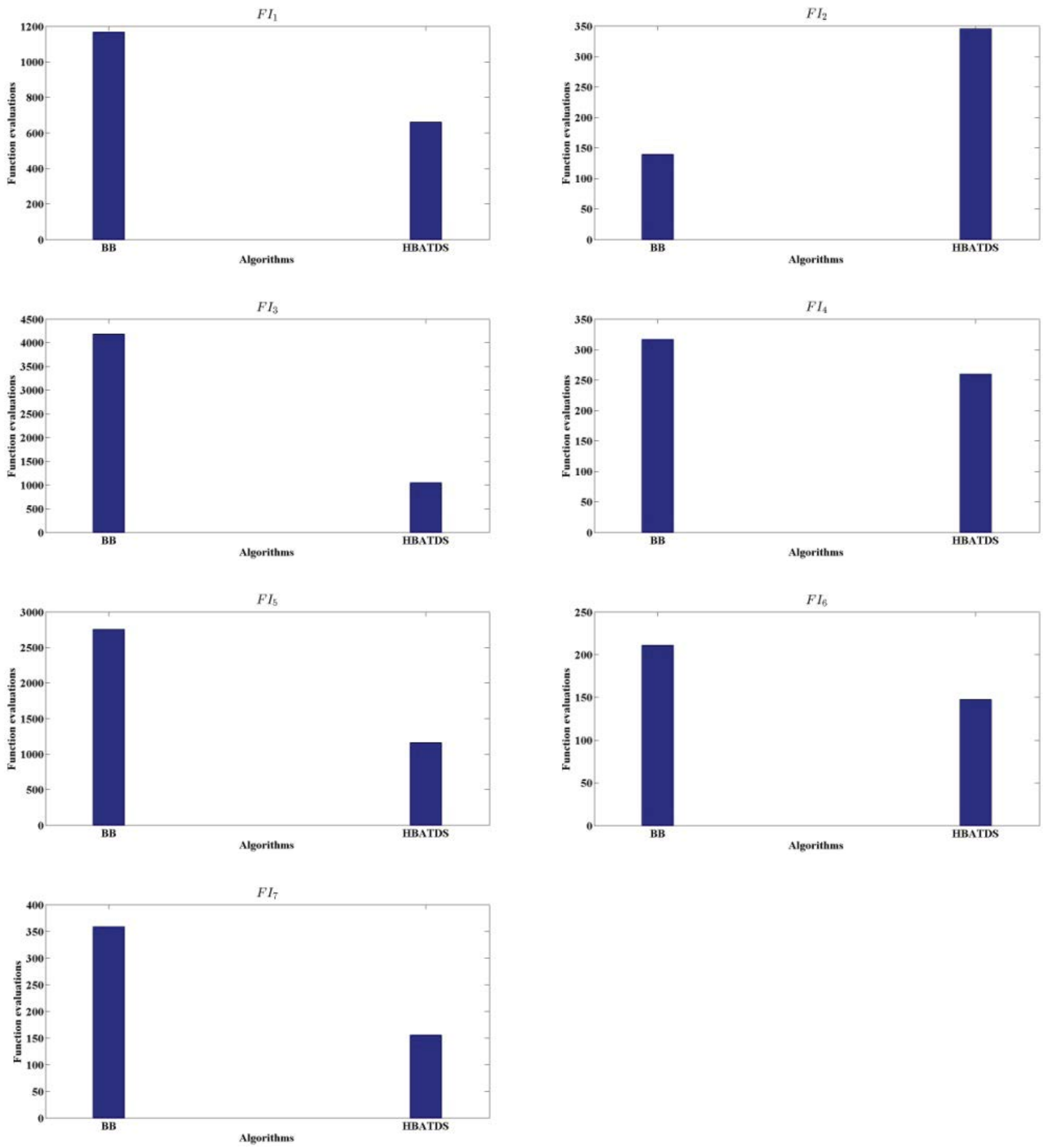

Figure 3: The comparison results (function evaluation values) between HBATDS algorithm and BB method for integer programming problems.

gorithm is a promising algorithm and can obtain the optimal or near optimal function values for most of the test functions (Table 8).

Conclusion and Future Work

In this paper, a new hybrid algorithm is suggested by incorporating the bat algorithm with direct search methods to solve integer programming problems. The suggested algorithm is named hybrid bat direct search algorithm (HBDS). In the suggested algorithm, The performance of the classic BA is accelerated by invoking the pattern search method as a local search method and the Nelder-Mead method in the final stage of the algorithm. The HBDS algorithm is intensely tested on 7 integer programming problems. The suggested algorithm is compared with other 10 algorithms to test its performance for integer programming problems. The numerical results illustrate that the suggested HBDS algorithm is a

Table 8: Experimental results (mean, standard deviation and rate of success) of function evaluation between BB method and HBATDS algorithm for FI_1- FI_7 test problems.

Function	Algorithm	Optimal	Mean	St. D	Suc
FI_1		**0.00**			
	BB		1167.83	659.8	30
	HBATDS		**663.097**	47.13	30
FI_2		**0.00**			
	BB HBATDS		**139.7**	102.6	30
			345.64	15.75	30
FI_3		**-737**			
	BB HBATDS		4185.5	32.8	30
			1150.67	36.71	30
FI_4		**0.00**			
	BB HBATDS		316.9	125.4	30
			265.24	4.93	30
FI_5		**0.00**			
	BB HBATDS		2754	1030.1	30
			1264.54	49.14	30
FI_6		**-6**			
	BB HBATDS		211	15	30
			147.77	4.45	30
FI_7		**-3833.12**			
	BB HBATDS		358.6	14.7	30
			215.48	5.14	30

potential algorithm and suitable to find a global optimal solution or close optimal solution in reasonable time.

Our work in this paper motivates us to work on the proposed algorithm to solve various optimization problems such as large scale and molecular energy function [7], other combinatorial problems, MIPLIB instances, large scale integer programming and minimax problems, and constrained optimization and engineering problems [50].

Acknowledgments

We are grateful to the referees for the detailed review of our paper, insightful comments and constructive suggestions which improve the structure of the paper. The research of the 2nd author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The postdoctoral fellowship of the 1st author is supported by NSERC.

References

1. DZ Du, PM Pardalos (1995) Minimax and applications. Kluwer.
2. GL Nemhauser, AHG Rinnooy Kan, MJ Todd (1989) Handbooks in OR & MS.
3. S Zuhe, A Neumaier, MC Eiermann (1990) Solving minimax problems by interval methods. BIT Numerical Mathematics 30: 742-751.
4. M Dorigo (1992) Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Italy.
5. X Zhang, S Wang, L Yi, et al. (2018) An integrated ant colony optimization algorithm to solve job allocating and tool scheduling problem. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 232: 172-182.
6. D Karaboga, B Basturk (2007) A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization 39: 459-471.
7. AF Ali, MA Tawhid (2017) A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems. Ain Shams Engineering Journal 8: 191-206.
8. J Kennedy, RC Eberhart (1995) Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks 4: 1942-1948.
9. A Sombat, T Saleewong, P Kumam (2018) Perspectives and experiments of hybrid particle swarm optimization and genetic algorithms to solve optimization problems. International Econometric Conference of Vietnam 290-297.
10. MK Passino (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems 22: 52-67.
11. MA Al-Betar, MA Awadallah, H Faris, et al. (2018) Bat-inspired algorithms with natural selection mechanisms for global optimization. Neurocomputing 273: 448-465.
12. XS Yang (2010) A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization 65-74.
13. D Teodorovic, M Dell'Orco (2005) Bee colony optimization - A cooperative learning approach to complex transportation problems. Advanced OR and AI Methods in Transportation 51-60.

14. R Tang, S Fong, XS Yang, et al. (2012) Wolf search algorithm with ephemeral memory. Seventh International Conference on Digital Information Management 165-172.
15. SC Chu, PW Tsai, JS Pan (2006) Cat swarm optimization. Pacific Rim International Conference on Artificial Intelligence 4099: 854-858.
16. XS Yang, S Deb (2009) Cuckoo search via Levy fights. In Proc. of World Congress on Nature & Biologically Inspired Computing, India, IEEE Publications, USA, 210-214.
17. XS Yang (2010) Firefly algorithm, stochastic test functions and design optimization. Int J Bio-Inspired Computation 2: 78-84.
18. XL Li, ZJ Shao, JX Qian (2002) Optimizing method based on autonomous animals: Fish-swarm algorithm. Systems Engineering - Theory & Practice 22: 32-38.
19. AF Ali, MA Tawhid (2016) Hybrid-simulated annealing and pattern search method for solving minimax and integer programming problems. Pacific Journal of Optimization 12: 51-184.
20. AF Ali, MA Tawhid (2016) Direct gravitational search algorithm for global optimization problems. East Asian Journal on Applied Mathematics 6: 290-313.
21. DS Chen, RG Batson, Y Dang (2011) Applied integer programming: Modeling and solution. John Wiley & Sons.
22. M Conforti, G Cornujols, G Zambelli (2014) Integer programming. Graduate Texts in Mathematics.
23. MA Tawhid, AF Ali (2016) A simplex social spider algorithm for solving integer programming and minimax problems. Memetic Computing 8: 169-188.
24. YG Petalas, KE Parsopoulos, MN Vrahatis (2007) Memetic particle swarm optimization. Ann Oper Res 156: 99-127.
25. N Bacanin, I Brajevic, M Tuba (2013) Firefly algorithm applied to integer programming problems. Recent Advances in Mathematics.
26. MA Tawhid, AF Ali (2016) Direct search firefly algorithm for solving global optimization problems. App Math Inf Sci 841-860.
27. AF Ali, MA Tawhid (2016) A hybrid cuckoo search algorithm with Nelder Mead method for solving global optimization problems. Springer Plus 5: 473.
28. M Tuba, M Subotic, N Stanarevic (2012) Performance of a modified cuckoo search algorithm for unconstrained optimization problems. WSEAS Transactions on Systems 11: 62-74.
29. R Jovanovic, M Tuba (2013) Ant colony optimization algorithm with pheromone correction strategy for minimum connected dominating set problem. Computer Science and Information Systems 10.
30. N Bacanin, M Tuba (2012) Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. Studies in Informatics and Control 21: 137-146.
31. M Tuba, N Bacanin, N Stanarevic (2012) Adjusted artificial bee colony (ABC) algorithm for engineering problems. WSEAS Transaction on Computers 11: 111-120.
32. MA Tawhid, AF Ali (2016) Simplex particle swarm optimization with arithmetical crossover for solving global optimization problems. OPSEARCH 53: 705-740.
33. R Jovanovic, M Tuba (2011) An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Applied Soft Computing 11: 5360-5366.
34. JH Lin, CW Chou, CH Yang, et al. (2012) A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. Journal Computer and Information Technology 2: 56-63.
35. JW Zhang, GG Wang (2012) Image matching using a bat algorithm with mutation. Applied Mechanics and Materials 203: 88-93.
36. XS Yang (2011) Bat algorithm for multi-objective optimization. International Journal of Bio-Inspired Computation 3: 267-274.
37. G Komarasamy, A Wahi (2012) An optimized K-means clustering technique using bat algorithm. European Journal Scientific Research 84: 263-273.
38. RYM Nakamura, LAM Pereira, KA Costa, et al. (2012) BBA: A binary bat algorithm for feature selection. In: 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE Publication, 291-297.
39. J Xie, Y Zhou, H Chen (2013) A novel bat algorithm based on differential operator and Levy flights trajectory. Comput Intell Neurosci 2013: 453812.
40. G Wang, L Guo (2013) A novel hybrid bat algorithm with harmony search for global numerical optimization. J Applied Mathematics 2013: 1-21.
41. FS Hillier, GJ Lieberman (1995) Introduction to operations research. MC Graw-Hill.
42. R Hooke, TA Jeeves (1961) Direct search, solution of numerical and statistical problems. J Ass Comput 8: 212-229.
43. JA Nelder, R Mead (1965) A simplex method for function minimization. Computer Journal 7: 308-313.
44. JH Holland (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, Michigan.
45. S Mirjalili, SM Mirjalili, A Lewis (2014) Grey wolf optimizer. Advances in Engineering Software 69: 46-61.
46. B Borchers, JE Mitchell (1992) Using an interior point method in a branch and bound algorithm for integer programming. Technical Report, Rensselaer Polytechnic Institute.
47. B Borchers, JE Mitchell (1994) An improved branch and bound algorithm for mixed integer nonlinear programs. Computers & Operations Research 21: 359-367.
48. EL Lawler, DW Wood (1966) Branch and bound methods: A Survey. Operations Research 14: 699-719.
49. VM Manquinho, JP Marques Silva, AL Oliveira, et al. (1997) Branch and bound algorithms for highly constrained integer programs. Technical Report, Cadence European Laboratories, Portugal.
50. AF Ali, MA Tawhid (2016) Hybrid PSO and DE algorithm for solving engineering optimization problems. Applied Mathematics and Information Sciences 10: 431-449.
51. G Rudolph (1994) An evolutionary algorithm for integer programming. In: Davidor Y, Schwefel HP, Mnner R, Parallel Problem Solving from Nature. 3: 139-148.
52. A Glankwahmdee, JS Liebman, GL Hogg (1979) Unconstrained discrete nonlinear programming. Engineering Optimization 4: 95-107.
53. SS Rao (1994) Engineering optimization-theory and practice. (4th edn), Wiley, New Delhi, India.

DOI: 10.36959/643/303 | Volume 2 | Issue 1