



## International Journal of Bio-Inspired Computation

ISSN online: 1758-0374 - ISSN print: 1758-0366 https://www.inderscience.com/ijbic

# A novel binary multi-swarms fruit fly optimisation algorithm for the 0-1 multidimensional knapsack problem

Xin Du, Jiawei Zhou, Youcong Ni, Wentao Liu, Ruliang Xiao, Xiuli Wu

DOI: 10.1504/IJBIC.2021.10047361

#### **Article History:**

Received:	3
Last revised:	1
Accepted:	1
Published online:	0

31 August 2021 14 December 2021 16 December 2021 04 April 2023

# A novel binary multi-swarms fruit fly optimisation algorithm for the 0–1 multidimensional knapsack problem

# Xin Du, Jiawei Zhou, Youcong Ni\*, Wentao Liu and Ruliang Xiao

College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China Email: xindu79@126.com Email: chenhuankeai@163.com Email: youcongni@foxmail.com Email: 1920993165@qq.com Email: xiaoruliang@163.com \*Corresponding author

### Xiuli Wu

School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, China Email: wuxiuli@ustb.edu.cn

**Abstract:** To improve solution quality and accelerate convergence speed of traditional fruit fly optimisation algorithm in solving MKP, a novel binary multi-swarm fruit fly optimisation algorithm (bMFOA) is proposed. It comprises four novelties. Firstly, an item frequency tree (IFT) is constructed based on the idea of frequency pattern mining, and a new search strategy is proposed to obtain heuristic information. Secondly, two new heuristic operators of 'ADD' and 'DROP' are designed according to the obtained heuristic knowledge. Thirdly, a multi-swarm cooperation strategy is presented to strengthen the exploitation capability. To prevent algorithm falling into the local optimum prematurely, a swarm location escape strategy is put forward. To verify the efficiency of bMFOA, it is compared with some existing meta-heuristic methods by solving 58 MKPs from ORLIB. The experimental results show that the bMFOA performs better than existing meta-heuristic methods.

Keywords: fruit fly optimisation; multidimensional knapsack problem; binary optimisation.

**Reference** to this paper should be made as follows: Du, X., Zhou, J., Ni, Y., Liu, W., Xiao, R. and Wu, X., (2023) 'A novel binary multi-swarms fruit fly optimisation algorithm for the 0–1 multidimensional knapsack problem', *Int. J. Bio-inspired Computation*, Vol. 21, No. 1, pp.1–10.

**Biographical notes:** Xin Du is currently a Professor at the College of Computer and Cyber Security, Fujian Normal University (CN). Her research interests include intelligent algorithms and search-based software engineering.

Jiawei Zhou is currently a Master at the College of Computer and Cyber Security, Fujian Normal University (CN). His research interest is intelligent algorithm.

Youcong Ni is currently an Associate Professor at the College of Computer and Cyber Security, Fujian Normal University (CN). His research interests include intelligent algorithms and search-based software engineering.

Wentao Liu is currently a Master's student at the College of Computer and Cyber Security, Fujian Normal University (CN). His research interest is intelligent algorithm.

Ruliang Xiao is currently a Professor at the College of Computer and Cyber Security, Fujian Normal University (CN). His research interests include intelligent algorithms and search-based software engineering.

Xiuli Wu is currently a Professor at the School of Mechanical Engineering, University of Science and Technology Beijing (CN). Her research interest include intelligent algorithm.

#### 1 Introduction

The multidimensional knapsack problem (MKP) is a combinatorial optimisation problem that belongs to the class of NP-hard problems. MKP is found in various applications, such as capital budgeting (Wilbaut et al., 2007), loading problems (Nawrocki et al., 2009), resource allocation (Wu, 2014). Mathematically, MKP can be formulated as equations  $(1)\sim(3)$ .

Maximise 
$$Z = \sum_{j=1}^{n} pro_j x_j$$
 (1)

s.t.

$$\sum_{j=1}^{n} C_{i,j} x_j \le R_i, i = 1, 2, 3, ..., m$$
<sup>(2)</sup>

$$x_j \in \{0,1\}, j = 1, 2, 3, ..., n$$
(3)

where  $x_j (j = 1, 2, \dots, n)$  are binary decision variables such that  $x_j = 1$  if item j is packed in the knapsack,  $x_j = 0$  otherwise. n is the number of items and m is the number of knapsack constraints with the capacity  $R_i (i = 1, 2, \dots, m)$ . Each item j consumes  $C_{i,j}$  units of resource in the  $i^{\text{th}}$  constraint and yields  $pro_j$  units of profit when it is selected. The objective in equation (1) aims to maximise the total profits of the selected items, while the constraints in equation (2) ensure that the selected items satisfy the mcapacity constraints of the knapsack.

Existing algorithms for the MKP can be classified into exact and heuristic algorithms. The best performing exact algorithm (Mansini and Speranza, 2011) is quite successful to yield optimal solutions in an acceptable computation time for benchmark instances of limited sizes (e.g., n = 250 or 500 and  $m \in \{5, 10\}$ ). However, for larger instances with n > 250 and n > 30, heuristic algorithms are more suitable to find sub-optimal solutions. Chu and Beasley (1998) firstly applied genetic algorithm to solve the MKP and proved that high-quality solutions can be obtained whilst requiring only a modest amount of computational effort. With the development of intelligent algorithms, numerous heuristic methods to solve MKPs have been recently proposed. Chih (2018) proposed a novel particle swarm optimisation to solve the MKP, where a self-adaptive check and repair operator was designed to substitute pseudo-utility ratios. Lai et al. (2019) proposed a two-stage tabu search algorithm, the first stage aimed to locate a promising search space and the second stage tried to find improved solutions. Lai et al. (2020) integrated a distanced-based diversity-preserving strategy and local optimisation method to solve the MKP. Liu et al. (2016) proposed a binary differential search to solve 0-1 MKPs, where the stochastic search is guided by a Brownian motion-like random walk. Feng et al. (2016) proposed an improved cuckoo search algorithm, which combined cuckoo search and global harmony search to improve the accuracy and convergence speed. Wang et al. (2013) proposed an improved binary fruit fly optimisation algorithm to solve the MKPs, which used probability vectors to generate fruit fly individual and applies differential evolution formulas to update probability vectors.

FOA is a novel swarm intelligence optimisation algorithm that simulates the foraging behaviour of fruit fly. It has the advantages of simpler model, less parameters and easier implementation. It has been used to solve many problems, such as antenna array synthesis (Darvish and Ebrahimzadeh, 2018), power load forecasting (Hu et al., 2017), economic theoretical model (Lin, 2016), function optimisation (Wang et al., 2019). For multi-dimensional optimisation problems, limited searching ability and premature convergence of basic FOA make them trap into local extreme or premature. Hence, many scholars had proposed some improved variants of FOA. Although those improved versions enhanced search abilities and convergence speeds, they were still difficult to find the global optimal in limited iterations. In this work, bMFOA is proposed. It has four main contributions. Firstly, IFT is constructed based on the idea of frequency pattern mining, and a new search strategy is proposed to obtain heuristic information. Secondly, a multi-swarms collaboration strategy is proposed to enhance swarm diversity. Thirdly, two heuristic search operators are designed based on heuristic information to improve the convergence speed of algorithm. Fourthly, a swarm location escape strategy is presented to avoid premature.

#### 2 bMFOA

bMFOA is mainly combined with heuristic mutation, multi-swarm cooperation and swarm location escape based on basic FOA.

#### 2.1 Procedure of bMFOA

bMFOA mainly consists of two parts, which are osphresis search phase and visual search phase. As shown in Figure 1, N and S refer to the number of swarms, the size of each swarm respectively. In the heuristic osphresis search phase, S individuals are generated according to swarm location and formed a new sub-swarm. Then, the drop operator or add operator will work with probability 0.5 respectively. In the visual search phase, the swarm location of each sub-swarm will be updated. The special notice that each sub-swarm is parallel search so far. And the multi-swarm cooperation strategy and swarm location escape strategy will be applied. Finally, the best solution will be obtained when algorithm's termination condition is met. Let an individual  $X_i(t) = (x_1(t), x_2(t), ..., x_l(t)),$ where  $x_i \in \{0, 1\}$ ,  $x_i = 1$  means the *i*<sup>th</sup> item is selected. Otherwise, it means the  $i^{th}$  item is not selected. l and t are the number of items and evolutionary generation respectively. The fruit fly swarm is denoted as P(t) = $\{P_1(t), P_2(t), \dots, P_N(t)\}$ , where N is the number of sub-swarms. The  $j^{\text{th}}$  sub-swarm is recorded as  $P_i(t) =$  $\{X_1(t), X_2(t), \dots, X_n(t)\}$ , where n is the size of each sub-swarm.





#### 2.2 Heuristic operator

bMFOA constructs an IFT by incorporating the idea of frequent pattern mining and provides a down-to-top (DTT) search method. Based on the heuristic information obtained by searching the IFT, two heuristic operators named drop and add operators, are designed. The construction of the IFT and the DTT search process are given below. Then the drop and add operators are introduced.

#### 2.2.1 Construction of IFT tree

The data structure of IFT is similar to the traditional frequent pattern (FP) tree (Han et al., 2000). It is also composed of a prefix tree (PT) and a header table (HT) shown in Figure 2.

Figure 2 An instance of IFT tree (see online version for colours)



#### Algorithm 1 Constructing IFT tree

Input: A global transaction database DB									
Output: IF I tree									
1 Sort all individuals of current swarm $P(t)$ in descend									
order by fitness value and select the top $\varphi\%$ individuals to form the elite group;									
individuals to form the elite group;									
Analyse all individuals in elite group and get HT by									
statistic times of the selected items and sorting the all									
items in descend order;									
3 IOF each A in eine group do									
4 Parse <i>X</i> , and get the <i>id</i> set of corresponding									
selected items, denoted as $C = \{i \mid i = 1 \text{ A } i \in V\}$									
$S_s = \{i   x_i = 1 \land x_i \in A\};$									
5 Sort corresponding item of $S_s$ according to the									
order of item s <i>ia</i> in H1;									
6 Insert $S_s$ into DB as a transaction;									
7 end									
s root $\leftarrow$ null for For each $T_q$ in DB do									
9 $temp \leftarrow root \text{ for } each i_j \text{ in } T_q \text{ do}$									
10 Calculate $virValue$ of item $i_j$ by equation (4)									
11 If node temp has childNode with property of									
id equals to $i_j$ then									
12 $childNode.count \leftarrow$									
childNode.count + 1;									
13 $childNode.virValue \leftarrow$									
childNode.virValue +									
$virtual Pro(i_j T_q);$									
14 $temp \leftarrow childNode;$									
15 else									
16 Create new node <i>newNode</i> ;									
17 $newNode.id \leftarrow i_j;$									
18 $newNode.count \leftarrow 1;$									
19 $newNode.virValue \leftarrow$									
$virtualPro(i_j, T_q);$									
20 Take <i>newNode</i> as child-node of <i>temp</i> ;									
21 $temp \leftarrow newNode;$									
Find out corresponding header pointer link									
of item $i_j$ in HT, then link <i>newNode</i> to									
corresponding node chain;									
23 end									
24 end									
25 $temp \leftarrow root;$									
26 end									

 Table 1
 An instance of transaction database DB

TID	Transaction	
$T_1$	$\{3, 1, 4\}$	
$T_2$	{2, 1}	
$T_3$	$\{2, 3, 5\}$	
$T_4$	$\{2, 3, 1\}$	
$T_5$	{2, 3}	
$T_6$	{2, 4}	

The IFT is constructed based on the global transaction database  $DB = \{T_1, T_2, ..., T_s\}$ , where s is the size of DB. A transaction T consists of the set of item number(*id*) corresponding to selected items in an individual. It is denoted as  $T = \{i_1, i_2, ..., i_n\}$ , where  $i_k$  is the *id* of the  $k^{\text{th}}$  item. Each node in IFT consists of four fields: *id*, *count*, *virValue* and *nextNode*. Considering the mutual influences among different items in a transaction, the virtual

#### 4 *X. Du et al.*

value of item is named as virValue shown in equation (4), which is the sum of profits of all the items in a transaction T. In equation (4), n is the number of items and  $pro_{i_k}$  is the profit of item  $i_k$ . Therefore, the virtual value of item j in DB is defined as equation (5).

$$virtualPro(j,T) = \sum_{k=1}^{n} pro_{i_k}, i_k \in T$$
(4)

$$virtualPro(j, DB) = \sum_{k=1}^{n} virtualPro(j, T_k)$$
 (5)

Suppose that item number set is  $ids = \{1, 2, 3, 4, 5\}$ , its corresponding profit set is  $pros = \{2, 6, 13, 8, 5\}$ . Table 1 gives an instance of transaction database DB. Table 2 shows an instance of calculating the virtual value of items. The occurrence number of these items of DB in Table 1 is  $count = \{3, 5, 4, 2, 1\}$ . Each item in HT contains two fields: *id* and head pointer link of a node chain constructed by the dashed arc. The nodes with same *id* can be linked together through node chain. For example, the nodes with same id(3) are linked together through the head pointer link in the second row of HT in Figure 2. A simple example on the construction procedure of IFT tree is given in Tables 1 and 2. The transactions of DB in Table 1 are read in order. Next, their corresponding nodes are formed and inserted into the IFT tree. The process of insertion is shown as follows. Firstly, three nodes (3, 1, 23), (1, 1, 23), (4, 1, 23)are formed according to the first transaction  $T_1$  from DBand inserted into IFT tree in order. Accordingly, the left sub-tree of root node in Figure 2 is obtained. Secondly, two nodes (2,1,8), (1,1,8) are constructed according to the second transaction  $T_2$  from DB and inserted into the IFT tree in order. Thirdly, three nodes (2, 1, 24), (3, 1, 24), (5, 1, 24) are constructed according to the third transaction  $T_3$  from DB. The count and virValue of item 2 need to be updated by accumulating the two parts because item 2 has already appeared in the IFT tree. Another nodes are inserted according to above methods. Construct the rest of nodes by reading the transactions from DB. Insert these nodes into the IFT tree. Finally, an instance of IFT tree is formed in Figure 2. The construction process is given in Algorithm 1.

Table 2 An instance of calculating the virtual value of item

id	virtualPro(j,T)	virtualPro(j, DB)
1	$virtualPro(1, T_1) = 23$	52
	$virtualPro(1, T_2) = 8$	
	$virtualPro(1, T_4) = 21$	
2	$virtual Pro(2, T_2) = 8$	86
	$virtualPro(2, T_3) = 24$	
	$virtualPro(2, T_4) = 21$	
	$virtualPro(2, T_5) = 19$	
	$virtualPro(2, T_6) = 14$	
3	$virtualPro(3, T_1) = 23$	87
	$virtualPro(3, T_3) = 24$	
	$virtualPro(3, T_4) = 21$	
	$virtualPro(3, T_5) = 19$	
4	$virtualPro(4, T_1) = 23$	37
	$virtualPro(4, T_6) = 14$	
5	$virtualPro(5, T_3) = 24$	24

#### Algorithm 2 DTT search algorithm

	Input:	IFT	tree
--	--------	-----	------

**Output:** Scandi

- Decode the swarm location  $X_j(t-1)$  of sub-swarm  $P_j(t-1)$  to get corresponding *id* set of selected items, denoted as  $S_s = \{i | x_i = 1 \land x_i \in X_j(t-1)\};$
- 2 Generate a sub-set  $S_s^*$  by selecting randomly k elements from  $S_s$ ;
- 3 Find the item with the least occurrence number in  $S_s^*$  based on the item header table HT, and get its id number i;
- 4 Traverse the node chain corresponding to *i* in *HT*, and obtain a node set *nodeSet*;

```
5 Let S_{candi} = \phi;
```

8

- $\mathbf{6}\ \mathbf{for}\ each\ node\ in\ nodeSet\ \mathbf{do}$
- 7 while node is not root do
  - if node.id  $\notin S_s^*$  then

9		I there has no element in S <sub>candi</sub> whose
		value of first element is equal to node.id
		then
10		Create a new element $y$ ;
11		$y.id \leftarrow node.id;$
12		$y.count \leftarrow node.count;$
13		$y.virValue \leftarrow node.virValue;$
14		Add $y$ into $S_{candi}$ ;
15		else
16		$y.count \leftarrow y.count + node.count;$
17		$y.virValue \leftarrow$
		y.virValue + node.virValue;
18		end
19		end
20		$node \leftarrow node.parNode;$
21	e	nd
22	end	

Algorithm 3 The pseudo code of add operator

	<b>Input:</b> The frequent item set $S_{add}$ , the location of sub-swarm $X_j(t-1)$ , the frequent item set $S_{add}$ , with virValue annotation
	Output: A new individual $Y_{(t)}$
	<b>Solution</b> $A$ new individual $A_{new}(t)$
1	$\prod_{i=1}^{n}  S_{add}  = 0 \text{ then}$
2	Mutate one bit of $X_j(t-1)$ randomly and a new
	individual $X_{new}(t)$ is got;
3	else
4	Initialise the set $W = \{w_j   w_j = \langle i, weight \rangle\};$
5	for each $i$ in $S_{add}$ do
6	Get the corresponding element $y$ of the item $i$
	in the set $S_{candi}$ ;
7	Calculate $weight = y.count/totalCount +$
	y.virValue/totalPro;
8	Add $\langle i, weight \rangle$ into the set W;
9	end
10	Search the corresponding element $w_i$ of the set W
	whose first item is equal to <i>i</i> in the set $S_{add}$ ;
11	Select the item <i>i</i> with probability
	$p_i = w_i.weight / \sum_{i=1}^{ S_{add} } w_j.weight;$
12	Obtain a new individual $X_{new}(t)$ by shifting the
	value of the position in $X_i(t-1)$ corresponding
	to the $id$ of item $i$ to 1 :
2	and
3	<b>Clu</b> <b>D</b> so is the same is distingent $\mathbf{Y}_{i}$ (1) has a size a function
4	Repair the new individual $A_{new}(t)$ by using adaptive repair operator RO1 in Li and He (2019):
	1 1 (/)

Algorithm 4 The pseudo code of drop operator

**Input:** The frequent item set  $S_{drop}$ ,  $X_j(t-1)$ , the frequent item set  $S_{candi}$ **Output:** A new individual  $X_{new}(t)$ 1 if  $|S_{drop}| \leq 1$  then Mutate one bit of  $X_j(t-1)$  randomly and a new individual  $X_{new}(t)$  is got; 2 3 else Initialise the set  $W = \{w_i | w_i = \langle i, weight \rangle\};$ 4 for each i in  $S_{drop}$  do 5 Get the corresponding element y of the item i in the set  $S_{candi}$ ; 6 Calculate weight = y.count/totalCount + y.virValue/totalPro, totalCount is the sum of the counts of all the 7 items in DB, totalPro is the sum of the virValue of all the items in DB which is calculated by equation (5); Add  $\langle i, weight \rangle$  into the set W; 8 end 9 Search the corresponding element  $w_j$  of the set W whose first item is equal to i in the set  $S_{drop}$ ; Select the item i with probability  $p_i = \left(\sum_{j=1}^{|S_{drop}|} w_j.weight - w_i.weight\right) / \left((|S_{drop} - 1|) \times \sum_{j=1}^{|S_{drop}|} w_j.weight\right);$ 10 11 Obtain a new individual  $X_{new}(t)$  by shifting the value of the position in  $X_j(t-1)$  corresponding to the *id* of item *i* to 12 0:13 end 14 Repair the individual  $X_{new}(t)$  by using adaptive repair operator RO1 in Li and He (2019); 15 Output the individual  $X_{new}(t)$ ;

#### 2.2.2 DTT search algorithm

Based on the construction process of IFT tree, it can be seen that the deeper the node in the IFT tree, the fewer the occurrence number of the item corresponding to the node. To search these items with more occurrence number and larger virtual value, a DTT search strategy is proposed to obtain frequent item set with virValue annotation as heuristic information. The frequent item set is denoted as  $S_{candi} = \{y | y = \langle id, count, virValue \rangle\}$ . Algorithm 2 shows the DTT search algorithm.

#### 2.2.3 Drop operator

For each sub-swarm  $P_j(t-1)$ , a set  $S_s = \{i|x_i = 1 \land x_i \in X_j(t-1)\}$  of *ids* of selected items can be acquired by decoding its location  $X_j(t-1)$ . Then, a frequent item set  $S_{drop} = \{i|i \in S_s \land i \in S_{candi}\}$  can be obtained by projecting the set  $S_{candi}$  based on the acquired set  $S_s$ .Execute drop operator by roulette selection strategy based on the set  $S_{drop}$ . The pseudo code of drop operator is given in Algorithm 4.

#### 2.2.4 Add operator

For each sub-swarm  $P_j(t-1)$ , a set  $S_u = \{i | x_i = 0 \land x_i \in X_j(t-1)\}$  of *ids* of unselected items can be acquired by decoding its location  $X_j(t-1)$ . Then, a frequent item set  $S_{add} = \{i | i \in S_u \land i \in S_{candi}\}$  can be obtained by projecting the set  $S_{candi}$  based on the acquired set  $S_u$ . Execute Add operator by roulette selection strategy based on the set  $S_{add}$ . The pseudo code of add operator is given in Algorithm 3.

#### 2.3 Multi-swarm cooperation strategy

To improve the global search capability of bMFOA, a strategy of multi-swarm cooperation is proposed. The location  $X_i(t)$  of sub-swarm  $P_i(t)$  successively executes uniform crossover with all the individuals in sub-swarm  $P_j(t)(j \neq i)$  selected randomly. Then, a new sub-swarm Q(t) is generated. Finally, the best individual among sub-swarms  $P_i(t)$  and Q(t) is selected as the location of Q(t).





#### 2.4 Swarm location escape

To avoid the algorithm falling into the local optimum prematurely, inspired by Han et al. (2018), a novel swarm location escape strategy is designed to help the swarm tackle the issue. The basic idea is that migrating the location of swarm in a probabilistic way according to equation (6). In equation (6), c is a constant,  $X_{best}(t)$  and  $X_{worst}(t)$  represent the best and the worst individuals of each sub-swarm, and f is the fitness evaluation function. According to the experimental result of parameters setting, bMFOA can obtain better solutions while the parameter c = 20. So, c is set to 20 in bMFOA.

$$\lambda = \frac{1}{e^{\frac{f(X_{best}(t)) - f(X_{worst}(t))}{c}}}, \lambda \in (0, 1]$$
(6)

#### **3** Simulation experiments

To verify the performance of the bMFOA, three sets of classic Benchmark instances (avaiable from ORLIB) are considered. The first set involves 18 cases with n = 20 to 105 and m = 2 to 30. The second set has 30 cases with n = 30 to 90 and m = 5. The third set consists of 11 cases with n = 100 to 1,500 and m = 15 to 50. Algorithm is coded in Java and run on a 3.3GHz Intel i5-4590 CPU. For all the experiments, all the algorithms are run 30 times independently with maximum evaluation number ( $E_{max} = 100,000$ ) as the termination condition.

Table 3 Combinations of parameter values

Danamatan	Factor level							
1 urumeter	1	2	3	4				
N	5	25	50	100				
S	5	25	50	100				
$\varphi$	10	20	30	40				

Experiment		Factors	ARV	
number	N	S	$\varphi$	ЛЦ
1	1	1	1	7,517.60
2	1	2	2	7,506.56
3	1	3	3	7,508.18
4	1	4	4	7,508.16
5	2	1	2	7,531.54
6	2	2	1	7,540.08
7	2	3	4	7,542.68
8	2	4	3	7,542.90
9	3	1	3	7,534.60
10	3	2	4	7,544.04
11	3	3	1	7,543.80
12	3	4	2	7,540.62
13	4	1	4	7,357.00
14	4	2	3	7,542.12
15	4	3	2	7,538.32
16	4	4	1	7,522.12

Table 4 Orthogonal array and ARV value

#### 3.1 Parameter setting

The proposed bMFOA contains three key parameters: the number of sub-swarms (N), the size of each sub-swarm (S), the percentage of elite individuals selected from all sub-swarms  $(\varphi)$ . To investigate the influence of these parameters on the performance of bMFOA, we adopt the

Taguchi method of design of experiment (DOE) by using the instance MK\_gk05. The combinations of different values of these parameters are listed in Table 3. The average response variable value (ARV) is the average of profit. According to the number of parameters and the number of factor levels, we choose the orthogonal array  $L_{16}(4^4)$ . The orthogonal array and the obtained ARV values are listed in Table 4. The trend of each factor level is shown in Figure 3 according to the orthogonal table. Based on the comprehensive analysis, the values of parameters are set as N = 25, S = 25,  $\varphi = 10$  and used for the third test set. Due to the first test set and the second test set involve in smaller scale instances, the values of parameters are set as N = 100, S = 100 and  $\varphi = 10$ .

#### 3.2 Comparisons of other algorithms

In this section, several classic algorithms of TR-BDS and TE-BDS in Liu et al. (2016), HPSOGO in Mingo et al. (2018) and HBDE in He et al. (2021) are selected as compared algorithms. TR-BDS, TE-BDS and HBDE are improved difference algorithms, HPSOGO is a hybrid and genetic inspired particle swarm optimisation, HPSOGO applied genetic operator to generate a new particle and added penalty functions to handle constraints. Liu et al. (2016) indicated that TR-BDS and TE-BDS were superior to MBPSO in Bansal and Deep (2012) and CBPSOCTVA in Chih et al. (2014) that are efficient methods for solving MKP problems. Thus, we compare bMFOA with TR-BDS, TE-BDS, HPSOGO and HBDE based on three test sets. For sake of fairness, these indicators of Min.Dev, Ave.Dev, Var.Dev, Mean, Std and ACT used in their original paper are adopted together.

From Table 5, bMFOA can obtain the known optimal solutions for 12 out of 18 instances. Instead, TR-BDS and TE-BDS can only obtain the known optimal solutions 5 and 3 out of 18 instances respectively. HPSOGO cannot obtain the known optimal solutions in all instances, HBDE can obtain the known optimal solutions for 16 out of 18. In terms of ACT, bMFOA can get the shortest ACT for 8 out of 18 instances, HPSOGO can get the shortest ACT for 6 out of 18 instances, HBDE can get the shortest ACT for 4 out of 18 instances. From Table 6, bMFOA can obtain the known optimal solutions in all 30 cases. TR-BDS and TE-BDS can only get 10 and 3 out of 18 instances respectively. HPSOGO cannot obtain the known optimal solutions in all instances, HBDE can obtain the known optimal solutions for 28 out of 30. As for ACT, bMFOA can get the shortest ACT for 13 out of 30 instances, HPSOGO can get the shortest ACT for 14 out of 30 instances, HBDE can get the shortest ACT for 3 out of 30 instances. In Table 7, we can know that the results acquired by bMFOA are all superior to those of TR-BDS, TE-BDS and HPSOGO, HBDE in all instances in terms of Mean and 8 out of 10 instances in terms of Std. However, HPSOGO outperforms the others algorithms in terms of ACT.

Problem	$n \times m$	Rest know			TR-BDS				TE-BDS	
		Dest Milen	Mean	ļ,	Std		ACT(s)	Mean	Std	ACT(s)
Sento1	60×30	7,772	7,756.13	3	60.24		21.71	7,758.43	18.74	24.22
Sento2	60×30	8,722	8,719.9	3	4.03		21.69	8,717.43	6.07	24.80
hp1	$28 \times 4$	3,418	3,409.8	0	6.82		16.33	3,412.60	6.73	19.04
hp2	35×4	3,186	3,185.5	7	2.37		16.57	3,186.00	0.00	19.25
pb1	$27 \times 4$	3,090	3,083.1	7	6.96		16.16	3,083.60	6.97	18.67
pb2	34×4	3,186	3,185.5	7	2.37		16.52	3,185.40	3.29	19.08
pb4	29×2	95,168	95,168.0	)0	0.00		15.84	95,168.00	0.00	18.38
pb5	$20 \times 10$	2,139	2,135.6	0	6.92		16.49	2,135.03	7.31	18.86
pb6	40×30	776	769.80	)	8.12		20.16	768.87	12.28	22.82
pb7	37×30	1,035	1,034.5	3	1.36		20.14	1,033.97	2.83	22.71
Weing1	$28 \times 2$	141,278	141,278.	00	0.00		15.65	141,261.33	91.29	17.77
Weing2	$28 \times 2$	130,883	130,883.	00	0.00		15.74	130,872.33	40.59	17.65
Weing3	$28 \times 2$	95,677	95,003.8	37	248.59		15.61	94,837.67	611.37	17.60
Weing4	$28 \times 2$	119,337	119,337.	00	0.00		15.65	119,337.00	0.00	17.69
Weing5	$28 \times 2$	98,796	97,662.6	50	1,760.89		16.97	97,001.27	1,955.54	17.61
Weing6	$28 \times 2$	130,623	130,623.	00	0.00		16.75	130,610.00	71.20	17.59
Weing7	$105 \times 2$	1,095,445	1,087,354	.87	1,453.51		21.70	1,088,624.43	1,445.27	22.60
Weing8	$105 \times 2$	624,319	620,258.	83	8,671.78		20.05	619,356.73	9,046.16	21.21
Problem		HPSOGO				HBDE			<i>bMFOA</i>	
11001011	Mean	Std	ACT(s)	M	ean	Std	ACT(s	) Mean	Std	ACT(s)
Sento1	6,815.90	409.08	0.86	7,77	2.00	0.00	1.96	7,772.00	) 0.00	1.17
Sento2	6,883.40	382.61	0.87	8,72	21.91	0.29	0.81	8,722.00	0.00	6.15
hp1	3,395.53	19.14	0.59	3,41	8.00	0.00	0.76	3,413.13	8 8.19	7.64
hp2	3,176.60	9.01	0.63	3,18	86.00	0.00	0.95	3,155.67	7 12.31	21.76
pb1	3,071.37	350.80	0.55	3,09	0.00	0.00	0.76	3,088.13	3 4.76	4.22
pb2	3,160.23	26.48	0.61	3,18	86.00	0.00	0.92	3,165.07	13.94	18.36
pb4	94,796.40	350.80	0.55	95,1	68.00	0.00	0.79	95,168.0	0 0.00	0.36
pb5	2,128.93	9.95	0.52	2,13	<b>69.00</b>	0.00	0.59	2,139.00	0.00	0.19
pb6	774.00	2.43	0.76	770	5.00	0.00	1.41	776.00	0.00	0.38
pb7	1,012.57	17.83	0.71	1,03	34.96	0.20	1.25	1,035.00	0.00	0.63
Weing1	133,100.00	5,306.77	0.52	141,2	278.00	0.00	0.76	141,278.0	0.00 0.00	0.73
Weing2	126,681.30	3,156.12	3.23	130,8	883.00	0.00	0.78	130,883.0	0.00 0.00	0.67
Weing3	95,289.10	376.04	3.17	95,6	77.00	0.00	0.78	95,677.0	0 0.00	0.23
Weing4	116,661.20	2,781.42	3.19	119,3	37.00	0.00	0.76	119,337.0	0.00	0.20
Weing5	98,216.70	645.69	3.24	98,7	96.00	0.00	0.78	98,796.0	0 0.00	0.17
Weing6	126,764.13	3,282.10	3.22	130,6	523.00	0.00	0.77	130,558.0	00 145.34	3.19
Weing7	688,657.03	33,845.18	4.32	1,095	,445.00	0.00	2.73	1,095,384	.10 11.31	21.36
Weing8	617,959.17	4,752.18	4.28	624,3	819.00	0.00	2.88	624,319.0	0.00 0.00	3.50

Table 5 Comparisons among TR-BDS, TE-BDS, HPSOGO, HBDE and bMFOA based on test set 1

The results of Tables 5, 6 and 7 display that bMFOA can get better results than TR-BDS, TE-BDS and HPSOGO in solving small, medium-scale and large-scale MKP problems, bMFOA can get competitive results in solving small, medium-scale MKP problems than HBDE and get better results in large-scale MKP problems. In brief, bMFOA performs better than TR-BDS, TE-BDS and HPSOGO in most instances in terms of Mean, Std and ACT based on three test sets, better than HBDE in terms of Mean, Std based on the third test set. Based on the above experimental results, it is fair to conclude that bMFOA has a good exploitation and exploration ability in solving MKP problems with various scale.

#### 4 Conclusions

In this study, we proposed an improved FOA algorithm for solving the MKP. To handle the problem of lacking of swarm diversity in FOA, a parallel search method is applied. By introducing cooperation strategy among the sub-swarms, the exploitation ability of algorithm is greatly improved. To strength the exploration ability of the algorithm, bMFOA constructs an item frequency tree, which contains global information, and provides a novel search strategy based on the constructed frequency tree. Then according to the obtained heuristic information, two kinds of operators are designed to accelerate the process of searching in the solution space. The performance of our proposed method was evaluated using 58 different widely used benchmarks from the OR-Library, the results show that bMFOA has a good exploitation and exploration ability in solving MKP problems with different scales.

# 8 *X. Du et al.*

Table 6 Comparisons among TR-BDS, TE-BDS, HPSOGO, HBDE and bMFOA based on test set 2

Duchlom	10 \ 100	Past know		TR-BDS			TE-BDS				
Froblem	$n \times m$	Desi know	Mean	n Std AC		CT(s)	Mean	Std	ACT(s)		
weish01	30×5	4,554	4,554.00	0.00	1	5.27	4,554.00	0.00	16.74		
weish02	30×5	4,536	4,536.00	0.00	1	5.27	4,536.00	0.00	16.76		
weish03	30×5	4,115	4,115.00	0.00	1	5.22	4,111.78	13.28	16.74		
weish04	30×5	4,561	4,561.00	0.00	1	5.13	4,561.00	0.00	16.72		
weish05	30×5	4,514	4,514.00	0.00	1	5.12	4,511.66	11.62	16.66		
weish06	40×5	5,557	5,557.00	0.00	1	5.82	5,556.48	2.57	17.44		
weish07	40×5	5,567	5,566.66	2.40	1	5.81	5,565.28	5.21	17.37		
weish08	40×5	5,605	5,604.96	0.28	1	5.86	5,604.76	0.66	17.45		
weish09	40×5	5,246	5,246.00	0.00	1	5.77	5,232.54	84.01	17.26		
weish10	50×5	6,339	6,339.00	0.00	1	6.30	6,336.58	10.79	17.78		
weish11	50×5	5,643	5,636.40	25.67	1	6.18	5,628.28	29.50	17.78		
weish12	50×5	6,339	6,335.12	19.20	1	6.24	6,335.98	15.56	17.79		
weish13	50×5	6,159	6,158.26	5.23	1	6.20	6,151.14	22.02	17.74		
weish14	60×5	6,954	6,951.10	10.03	1	6.84	6,951.94	12.13	18.33		
weish15	60×5	7,486	7,484.24	8.71	1	6.83	7,471.56	62.36	18.39		
weish16	60×5	7,289	7,289.00	0.00	1	6.88	7,287.44	10.46	18.38		
weish17	60×5	8,633	8,633.00	0.00	1	7.17	8,632.52	3.39	18.97		
weish18	70×5	9,580	9,579.38	4.38	1	7.71	9,578.02	6.26	19.45		
weish19	70×5	7,698	7,696.64	7.75	1	7.42	7.689.00	22.56	19.18		
weish20	70×5	9,450	9,449.64	1.78	1	7.64	9,447.02	8.36	19.34		
weish21	70×5	9,074	9.069.04	28.85	1	7.59	9.069.76	13.17	19.23		
weish22	80×5	8,947	8,946.24	5.37	1	7.88	8,942.06	19.28	19.54		
		HPSOGO			HBDE			bMFOA			
Problem -	Mean	Std	ACT(s)	Mean	Std	ACT(s)	Mean	Std	ACT(s)		
weish01	4,275.33	153.43	0.57	4,554.00	0.00	0.80	4,554.00	0.00	0.25		
weish02	4,300.90	169.12	0.57	4,536.00	0.00	0.82	4,536.00	0.00	0.25		
weish03	3,988.03	104.67	3.25	4,115.00	0.00	0.82	4,115.00	0.00	0.21		
weish04	4,290.10	179.04	3.33	4,561.00	0.00	0.84	4,561.00	0.00	0.20		
weish05	4,286.90	155.79	0.55	4,514.00	0.00	0.84	4,514.00	0.00	0.18		
weish06	4,933.17	252.21	0.64	5,556.57	2.33	1.09	5,557.00	0.00	0.97		
weish07	4,948.13	253.10	0.63	5,567.00	0.00	1.08	5,567.00	0.00	0.35		
weish08	4,934.80	280.29	0.65	5,605.00	0.00	1.08	5,605.00	0.00	0.76		
weish09	4,949.90	188.01	0.64	5,246.00	0.00	1.10	5,246.00	0.00	0.31		
weish10	5.738.77	308.04	0.76	6.339.00	0.00	1.39	6.339.00	0.00	0.48		
weish11	5,494.17	118.32	3.60	5,643.00	0.00	1.37	5,643.00	0.00	0.37		
weish12	5,748.07	223.25	3.58	6,339.00	0.00	1.39	6,339.00	0.00	0.49		
weish13	5,772.23	233.33	3.59	6,159.00	0.00	1.39	6,159.00	0.00	0.48		
weish14	6,255.83	290.93	3.73	6,954.00	0.00	1.67	6,954.00	0.00	1.13		
weish15	6,268.90	309.40	3.66	7,486.00	0.00	1.62	7,486.00	0.00	0.63		
weish16	6,278.43	309.03	3.72	7,289.00	0.00	1.65	7,289.00	0.00	2.22		
weish17	6,347.63	263.56	3.78	8,633.00	0.00	1.59	8,633.00	0.00	5.99		
weish18	7,272.17	419.16	3.89	9,580.00	0.00	1.88	9,580.00	0.00	10.61		
weish19	7,121.40	234.27	0.95	7,698.00	0.00	1.92	7,698.00	0.00	1.70		
weish20	7,163.07	386.77	0.93	9,450.00	0.00	1.88	9,450.00	0.00	4.59		
weish21	7,137.50	382.52	0.95	9,074.00	0.00	1.88	9,074.00	0.00	2.37		
weish22	7,628.97	383.57	1.05	8,947.00	0.00	2.18	8,947.00	0.00	4.89		

Problem	$n \vee m$	Rest know	TR-BDS				TE-BDS		
11001011	$n \wedge m$	Desi know	Mean	Std	ŀ	ACT(s)	Mean	Std	ACT(s)
weish23	80×5	8,344	8,342.74	7.64		17.81	8,332.24	24.95	19.49
weish24	80×5	10,220	10,215.88	8.77		18.16	10,218.86	3.84	19.89
weish25	80×5	9,939	9,937.74	3.87		18.02	9,932.98	14.11	19.70
weish26	90×5	9,584	9,578.74	30.85		18.39	9,575.70	19.60	20.01
weish27	90×5	9,819	9,816.94	14.57		18.27	9,802.18	47.03	19.96
weish28	90×5	9,492	9,488.74	16.59		18.28	9,484.30	20.63	19.97
weish29	90×5	9,410	9,403.62	23.36		18.30	9,402.84	22.79	20.00
weish30	90×5	11,191	11,181.52	13.82		18.67	11,184.16	11.63	20.00
Problem _		HPSOGO			HBDE			<b>b</b> MFOA	
11001cm -	Mean	Std	ACT(s)	Mean	Std	ACT(s)	Mean	Std	ACT(s)
weish23	7,568.30	338.01	1.03	8,343.90	0.54	2.19	8,344.00	0.00	4.01
weish24	7,642.60	335.64	1.03	10,220.00	0.00	2.12	10,220.00	0.00	12.38
weish25	7,561.90	387.75	1.04	9,939.00	0.00	2.16	9,939.00	0.00	7.52
weish26	8,142.77	299.61	1.11	9,584.00	0.00	2.47	9,584.00	0.00	5.96
weish27	7,919.53	341.12	1.12	9,819.00	0.00	2.46	9,819.00	0.00	6.55
weish28	8,034.87	350.69	1.11	9,492.00	0.00	2.46	9,492.00	0.00	4.55
weish29	8,073.33	402.12	1.12	9,410.00	0.00	2.46	9,410.00	0.00	5.13
weish30	3,395.53	19.14	0.56	11,191.00	0.00	2.39	11,191.00	0.00	16.22

Table 6 Comparisons among TR-BDS, TE-BDS, HPSOGO, HBDE and bMFOA based on test set 2 (continued)

 Table 7
 Comparisons among TR-BDS, TE-BDS, HPSOGO, HBDE and bMFOA based on test set 3

Problem	$n \times m$	Best know	TR-BDS				TE-BDS		
			Mean	Std	A	CT(s)	Mean	Std	ACT(s)
mk_gk01	100×15	3,766	3,688.26	6.02		21.20	3,720.86	7.01	22.59
mk_gk02	100×25	3,958	3,878.88	8.09		22.61	3,905.62	8.20	24.02
mk_gk03	150×25	5,650	5,512.06	7.55		25.61	5,542.22	9.87	27.00
mk_gk04	$150 \times 50$	5,764	5,623.70	9.16		29.66	5,648.32	8.50	31.12
mk_gk05	200×25	7,557	7,349.14	11.41		28.42	7,376.84	9.56	29.96
mk_gk06	$200 \times 50$	7,672	7,488.88	9.95		32.92	7,504.88	9.34	34.37
mk_gk07	500×25	19,215	1,8588.20	22.74		42.48	18,600.36	21.70	44.16
mk_gk08	$500 \times 50$	18,801	18,299.68	14.88		48.54	18,308.58	13.09	50.09
mk_gk09	1,500×25	58,085	56,035.20	38.46	1	01.90	56,058.74	36.00	103.50
mk_gk10	1,500×50	57,292	55,719.48	25.50	1	111.60	55,746.32	30.97	113.00
Problem -	HPSOGO			HBDE			bMFOA		
	Mean	Std	ACT(s)	Mean	Std	ACT(s)	Mean	Std	ACT(s)
mk_gk01	3,688.37	64.70	1.22	3,748.47	2.59	2.80	3,758.10	2.61	3.27
mk_gk02	3,868.30	59.32	1.25	3,943.30	3.46	2.88	3,944.70	3.53	4.09
mk_gk03	5,515.80	96.13	3.14	5,610.57	6.40	4.31	5,636.37	2.81	10.05
mk_gk04	5,535.90	105.11	3.38	5,711.23	6.28	4.76	5,748.20	3.30	11.39
mk_gk05	7,035.97	169.66	3.88	7,492.97	8.07	5.79	7,541.70	3.60	14.66
mk_gk06	7,284.10	209.28	4.48	7,599.77	6.00	6.39	7,647.93	3.39	33.32
mk_gk07	16,460.07	297.48	8.82	19,004.20	10.97	14.25	19,190.13	4.62	192.73
mk_gk08	16,330.17	284.62	10.64	18,581.10	6.61	16.06	18,754.70	6.48	201.12
mk_gk09	46,851.60	705.82	26.52	57,221.30	25.42	43.65	57,986.67	14.11	4,038.73
mk_gk10	46,612.67	484.35	26.36	56,490.63	26.64	46.85	57,139.63	16.45	4,552.37

#### Acknowledgements

This work is supported by the National Natural Science Foundation of China (62172097) and the Natural Science Foundation of Fujian Province (Nos. 2020J01165, 2021J01166).

#### References

- Bansal, J.C. and Deep, K. (2012) 'A modified binary particle swarm optimization for knapsack problems', *Applied Mathematics and Computation*, Vol. 218, No. 22, pp.11042–11061.
- Chih, M. (2018) 'Three pseudo-utility ratio-inspired particle swarm optimization with local search for multidimensional knapsack problem', *Swarm and Evolutionary Computation*, Vol. 39, No. 1, pp.279–296.
- Chih, M., Chin-Jung, L., Maw-Sheng, C. and Tsung-Yin, O. (2014) 'Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem', *Applied Mathematical Modelling*, Vol. 38, No. 4, pp.1338–1350.
- Chu, P.C. and Beasley, J.E. (1998) 'A genetic algorithm for the multidimensional knapsack problem', *Journal of Heuristics*, Vol. 4, No. 1, pp.63–86.
- Darvish, A. and Ebrahimzadeh, A. (2018) 'Improved fruit-fly optimization algorithm and its applications in antenna', *IEEE Transactions on Antennas and Propagation*, Vol. 66, No. 4, pp.1756–1766.
- Feng, Y., Wang, G. and Gao, X. (2016) 'A novel hybrid cuckoo search algorithm with global harmony search for 0–1 knapsack problems', *International Journal of Computational Intelligence Systems*, Vol. 9, No. 6, pp.1174–1190.
- Han, J., Pei, J. and Yin, Y. (2000) 'Mining frequent patterns without candidate generation', SIGMOD Rec., Vol. 29, No. 2, pp.1–12.
- Han, X., Liu, Q., Wang, H. and Wang, L. (2018) 'Novel fruit fly optimization algorithm with trend search and co-evolution', *Knowledge-Based Systems*, Vol. 141, No. 2, pp.1–17.
- He, Y., Xinlu, Z., Wenbin, L., Jinghong, W. and Ning, L. (2021) 'An efficient binary differential evolution algorithm for the multidimensional knapsack problem', *Engineering with Computers*, Vol. 37, No. 1, pp.745–761.
- Hu, R., Wen, S., Zeng, Z. and Huang, T. (2017) 'A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm', *Neurocomputing*, Vol. 221, No. 1, pp.24–31.
- Lai, X., Hao, J. and Yue, D. (2019) 'Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem', *European Journal of Operational Research*, Vol. 274, No. 1, pp.35–48.

- Lai, X., Hao, J., Fu, Z. and Yue, D. (2020) 'Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem', *Expert Systems with Applications*, Vol. 149, No. 7, p.113310.
- Lin, W-Y. (2016) 'A novel 3D fruit fly optimization algorithm and its applications in economics', *Neural Computing and Applications*, Vol. 27, No. 5, pp.1391–1413.
- Liu, J., Changzhi, W., Jiang, C., Xiangyu, W. and Lay, T.K. (2016) 'A binary differential search algorithm for the 0–1 multidimensional knapsack', *Applied Mathematical Modelling*, Vol. 40, No. 23, pp.9788–9805.
- Li, Z. and He, Y. (2019) Research and Application of Grey Wolf Optimization Algorithm, Hebei Geo University.
- Mansini, R. and Speranza, M.G. (2011) 'Coral: an exact algorithm for the multidimensional knapsack problem', *INFORMS Journal* on Computing, Vol. 24, No. 3, pp.399–415.
- Mingo, L.L.F., Gacuteomez, B.N. and Arteta, A.A. (2018) 'Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations', *Soft Computing*, Vol. 22, No. 8, pp. 2567–2582.
- Nawrocki, J., Complak, W., Błażewicz, J., Kopczyńska, S. and Maćkowiaki, M. (2009) 'The knapsack-lightening problem and its application to scheduling HRT tasks', *Bulletin of the Polish Academy of Sciences: Technical Sciences*, Vol. 57, No. 1, pp.71–77.
- Wang, L. and Zheng, X. (2017) 'Research progress of fruit fly optimization algorithm', *Control Theory and Application*, Vol. 34, No. 5, pp.557–563.
- Wang, L., Xiong, Y., Li, S. and Zeng, Y. (2019) 'New fruit fly optimization algorithm with joint search strategies for function optimization problems', *Knowledge-Based Systems*, Vol. 176, No. 7, pp.77–96.
- Wang, L., Zheng, X. and Wang, S. (2013) 'A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem', *Knowledge-Based Systems*, Vol. 48, No. 4, pp.17–23.
- Wilbaut, C., Hanafi, S. and Salhi, S. (2007) 'A survey of effective heuristics and their application to a variety of knapsack problems', *IMA Journal of Management Mathematics*, Vol. 19, No. 3, pp.227–244.
- Wu, C., Xiangyu, W. and Jiang, L. (2014) 'Optimizations in project scheduling: a state-of-art survey', *Optimization and Control Methods in Industrial Engineering and Construction*, Vol. 72, No. 1, pp.161–177, Springer, Netherlands.