Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5092480, 15 pages
https://doi.org/10.1155/2021/5092480

Research Article

A New Moth-Flame Optimization Algorithm for Discounted {0-1} Knapsack Problem

Tung Khac Truong

Faculty of Information Technology, Van Lang University, Ho Chi Minh City, Vietnam

Correspondence should be addressed to Tung Khac Truong; tung.tk@vlu.edu.vn

Received 25 May 2021; Revised 6 June 2021; Accepted 18 June 2021; Published 1 November 2021

Academic Editor: Mohammad Yazdi

The discounted {0–1} knapsack problem may be a kind of backpack issue with gathering structure and rebate connections among things. A moth-flame optimization algorithm has shown good searchability combined with an effective solution presentation designed for the discounted {0-1} knapsack problem. A new encoding scheme used a shorter length binary vector to help reduce the search domain and speed up the computing time. A greedy repair procedure is used to help the algorithm have fast convergence and reduce the gap between the best-found solution and the optimal solution. The experience results of 30 discounted {0-1} knapsack problem instances are used to evaluate the proposed algorithm. The results demonstrate that the proposed algorithm outperforms the two binary PSO algorithms and the genetic algorithm in solving 30 DKP01 instances. The Wilcoxon rank-sum test is used to support the proposed declarations.

1. Introduction

The knapsack problem is a well-known problem in combinatorial optimization. There are many variants of knapsack problems such as 0-1 knapsack problem, multidimensional knapsack problem, change-making problem, generalized assignment problem, bin-packing problem, and discounted {0-1} knapsack problem (DKP01). Among the knapsack variants, the discounted {0-1} knapsack problem is new. The DKP01 is first introduced by Guldan in [1]. This problem has an important role within the modern commerce real world. It could be a portion of numerous key issues such as venture decision-making, mission determination, and budget control. A correct calculation based on energetic programming for the DKP01 is, to begin with, proposed in [1]. An approach that combined dynamic programming with the center of the DKP01 to unravel it is considered in [2]. Two calculations based on approximate methods for DKP01 are named FirEGA and SecEGA in [3].

DKP01 can be presented as follows:

$$\text{maximize } f(X) = \sum_{i=0}^{n-1} \left(x_{3i} p_{3i} + x_{3i+1} p_{3i+1} + x_{3i+2} p_{3i+2} \right), \tag{1}$$

$$\text{subject to} \quad x_{3i} + x_{3i+1} + x_{3i+2} \le 1, \quad i \in \{0, \ldots, n-1\}, \tag{2}$$

$$\left(x_{3i} w_{3i} + x_{3i+1} w_{3i+1} + x_{3i+2} w_{3i+2} \right) \le C, \tag{3}$$

$$x_{3i}, x_{3i+1}, x_{3i+2} \in \{0, 1\}, \quad \forall i \in \{1, 2, \ldots, n-1\}, \tag{4}$$

where x_{3i}, x_{3i+1}, and x_{3i+2} represent whether the items $3i, 3i+1$, and $3i+2$ are put into the a knapsack; $x_j = 0$ indicates the item $j\,(j = 0, 1, \ldots, 3n-1)$ is not in knapsack, while $x_j = 1$ indicates the item j is in the knapsack. It is worth noting that a binary vector $X = (x_0, x_1, \ldots, x_{3n-1}) \in \{0, 1\}^{3n}$ is a potential solution of DKP01. Only if X meets both (2) and (3), it is a feasible solution of DKP01. n is the number of groups, each group has three items, and each item has its profit and weight. The item is collected for knapsack aims to maximized profit while the weight capacity does not excess C. Each group does not contain more than one item.

Lately, they moreover had a point-by-point consideration of the calculations of the DKP01 and proposed greenhorn deterministic calculation and estimation calculations. A modern correct calculation and two guess calculations with an eager repair administrator were proposed to illuminate DKP01 [4]. A calculation based on PSO is named GBPSO utilizing discrete molecule swarm optimization [5]. An evolution algorithm combines with ring theory to solve DKP01 [6], binary moth search algorithm [7], and a teaching-learning-based optimization algorithm [8].

Moth-flame optimization is first proposed in [9] and it is successfully used to solve many optimization problems such as a quantum-behaved simulated annealing algorithm-based moth-flame optimization method [10], an efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing [11], a hybrid Harris hawks-moth-flame optimization algorithm including fractional-order chaos maps and evolutionary population dynamics [12], a differential moth-flame optimization algorithm for mobile sink trajectory [13], LVCI approach for optimal allocation of distributed generations and capacitor banks in distribution grids based on moth-flame optimization algorithm [14], real challenging constrained engineering optimization problems [15], parameters extraction of the three diode models for the multicrystalline solar cell [16], Alzheimer's disease diagnosis [17], profit maximization with integration of wind farm [18], and MFO with rolling mechanism to forecast the electricity consumption of inner Mongolia [19].

This research proposed a new moth-flame optimization (MFO) and a new encoding scheme for DKP01. A successful 0-1 vector with 2∗dimensional length is utilized for a solution combined with MFO. This advantageous solution present is first used by Truong [20]. The experience results on 30 discounted {0-1} knapsack problem (DKP01) instances are used to evaluate the proposed algorithm. The results demonstrate that the proposed algorithm outperforms the two binary PSO algorithms and genetic algorithm in solving 30 DKP01 instances:

(i) Moth-flame optimization algorithm has shown good searchability combined with an effective solution presentation designed to the discounted {0-1} knapsack problem.

(ii) A new encoding scheme used a shorter length binary vector to help reduce the search domain and speed up the computing time.

(iii) A greedy repair procedure is used to help the algorithm have fast convergence and reduce the gap between the best-found solution and the optimal solution.

The rest of this paper is organized in the following order: Section 2 presents related works. Section 3 proposes moth-flame optimization algorithm for DKP01. The simulated results of the proposed algorithms are presented in Section 4. We conclude this paper and suggest potential future work in Section 5.

2. Related Works

2.1. Particle Swarm Optimization. The PSO conducts its search utilizing a swarm of particles; a swarm of particles is arbitrarily made initially [21, 22]. The standard atom swarm optimizer keeps up a swarm of atoms that talk to the potential courses of action for the issue at hand. Suppose that the look space is D-dimensional and the position of ith particle of the swarm can be portrayed utilizing a D-dimensional vector, $x_i = (x_{i1}, \ldots, x_{id}, \ldots, x_{iD})$. The velocity of the particle x_i is described by a D-dimensional vector $v_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{iD})$. The last best position of ith particle is named $p_i = (p_{i1}, \ldots, p_{id}, \ldots, p_{iD})$. In substance, the heading of each atom is updated concurring to its claim flying encounter as well as to that of the finest atom inside the swarm. The basic PSO calculation can be portrayed as

$$\begin{aligned} v_{i,d}^{k+1} &= w \cdot v_{i,d}^{k} + c_1 \cdot r_1^k \cdot \left(p_{i,d}^k - x_{i,d}^k\right) + c_2 \cdot r_2^k \cdot \left(p_{g,d}^k - x_{i,d}^k\right), \\ x_{i,d}^{k+1} &= x_{i,d}^{k} + v_{i,d}^{k+1}, \end{aligned} \quad (5)$$

where $v_{i,d}^k$ is dth dimension velocity of particle i in cycle k; $x_{i,d}^k$ is the dth dimension position of particle i in cycle k; $p_{i,d}^k$ is the dth dimension position of personal best ($p$best) of particle i in cycle k; $p_{g,d}^k$ is the dth dimension position of global best particle (g_{best}) in cycle k; w is the inertia weight; c_1 is the cognitive weight and c_2 is a social weight; and r_1 and r_2 are two random values uniformly distributed in the range of [0, 1] [23].

2.2. Moth-Flame Optimization Algorithm. Mirjalili [9] proposed MFO in 2015 as a nature-inspired optimization algorithm that simulates the actions of individuals in a swarm of moths (search agents) that have unique night navigation methods. In the MFO algorithm, the candidate solutions are assumed to be search agents. In order to model how individuals move in a spiral, the m-by-d matrix namely M is used, where m stands for the number of search agents and d for the number of dimensions. It is assumed that, for each entity, there is an array for storing the value of the objective function as an m-by-one matrix, namely, OM.

The flame, which is defined in an m-by-d matrix called F, is also an important part of this algorithm. It is assumed that there is a way to store F's fitness value as OF in an array. When using the MFO algorithm, F can be thought of as M's best location in the search space. To mathematically model this action, each search agent's location is modified as follows:

$$M_i = S\left(M_i, F_j\right), \quad (6)$$

where M_i is the i^{th} search agent and F_j is the j^{th} best position found so far, and S indicates the logarithmic spiral function which is updated as follows:

Table 1: $2*n$ length binary encoding scheme.

No.	$2*n$ length binary vector	Meaning
1	00	No item of the group is chosen
2	01	The first item of the group is chosen
3	10	The second item of the group is chosen
4	11	The third item of the group is chosen

Table 2: $3*n$ length binary encoding scheme.

No.	$3*n$ length binary vector	Meaning
1	000	No item of the group is chosen
2	001	The first item of the group is chosen
3	010	The second item of the group is chosen
4	011	Violate constraint (2)
5	100	The third item of the group is chosen
6	101	Violate constraint (2)
7	110	Violate constraint (2)
8	111	Violate constraint (2)

$$S\left(M_i, F_j\right) = D_i e^{cr} \cos(2\pi r) + F_j, \tag{7}$$

where r is a random number in $[-1, 1]$, c is a constant that defines the shape of the logarithmic spiral, and D_i factor is the distance of the i^{th} search agent for the j^{th} flame, which is calculated as follows:

$$D_i = \left|F_j - M_i\right|. \tag{8}$$

M is required to use only one of the F to change its location in this algorithm, and an adaptive mechanism for the number of F is suggested as follows:

$$\text{flame no.} = \text{round}\left(N - \frac{t(N-1)}{T}\right), \tag{9}$$

where t is the current iteration number, N is the maximum number of flames, and T is the maximum iteration number.

3. The Proposed MFO for DKP01

3.1. Solution Presentation. Currently, there are two methods for presenting a solution: one uses a binary vector with a length equal to the dimensional of the problem which is $3*n$ [3, 7, 24, 25], and the other uses an integer vector with a length equal to the number of groups n [8].

The solution [20] is presented in this paper using a new binary encode scheme with a length of $2*n$. This encoding scheme has the benefit of being shorter in length and automatically satisfying constraint 2. In Table 1, a new binary encoding scheme is introduced. When compared with the previous solution presentation shown in Table 2, it has two disadvantages: it uses a longer vector to present a solution and there are four violate solutions in each scheme.

3.2. Repair Function. Constraint 2 is automatically satisfied by the current encoding scheme. A new repair based on the concept in [3] is proposed to manage restriction 3 and increase the consistency of the solution.

The benefit of this repair technique is that it strikes a balance between CPU time consumption and the avoidance of local optima. The profit-to-weight ratios are p_i/w_i $(i = 1, 2, \ldots, n)$ so that they are not increasing. It means that

$$\frac{p_i}{w_i} \geq \frac{p_j}{w_j}, \quad \text{for } i < j. \tag{10}$$

This repair operator consists of two phases. The first phase (called repair phase) examines each variable in an increasing order of p_j/w_j and drops an item from knapsack if feasibility is violated. The first phase (called optimization phase) examines each variable in an increasing order of p_j/w_j and add an item to knapsack as long as feasibility is not violated. The repair phase aims to obtain a feasible solution from an infeasible solution, whilst the optimization phase seeks to improve the fitness of a feasible solution. The details of this repair operator can be found in [20].

The overall pseudocode of the MFO algorithms for DKP01 is given in Algorithm 1.

3.3. Binary Moth-Flame Optimization Algorithm. The MFO algorithm was designed for real domain. To solve DKP01, MFO is used to redesign a search in a binary domain. Equations (11)–(13) are used to convert real vectors to binary vectors:

$$X(i, j) = \begin{cases} 0, & \text{if rand}() \geq \text{TF}(M(i, j)), \\ 1, & \text{if rand}() < \text{TF}(M(i, j)), \end{cases} \tag{11}$$

where TF (.) are the transforming functions of the probability as the following expressions:

```
Input: initial parameters
Output: optimal solution
Initialize search agents M
t = 1;
while t ≤ T do
    Update flame no. by equation (9)
    Generate binary X matrix by equation (11);
    Apply repair operator on X and assign its fitness to OM;
    if t == 1 then
        F = sort (M_1);
        OF = sort (OM_1);
    else
        F = sort (M_{t−1}, M_t);
        OF = sort (OM_{t−1}, OM_t);
    for i = 1:m do
        for j = 1:d do
            Calculate D by equation (8);
            Update M (i, j) by equations (6) and (7);
    t = t + 1.
```

Algorithm 1: The overall pseudocode MFO algorithm for DKP01.

$$TF_1(M(i,j)) = \frac{1}{1+e^{-2M(i,j)}}, \quad (12)$$

$$TF_2(M(i,j)) = \frac{1}{1+e^{-M(i,j)}}. \quad (13)$$

In this section, we proposed 2 binary algorithms based on MFO named MFO1 and MFO2. MFO1 and MFO2 use transfer function TF_1 (equation (12)) and TF_2 (equation (13)), respectively. They use formula (11) to compute binary vector X.

4. Simulation Results

In this paper, the experience results of MFO1 and MFO2 algorithms are compared to find out the best one to solve DKP01 among them. The proposed algorithms are used to compare the results of three algorithms took from [6] named as SecEGA and two PSO algorithms took from Truong [20]. The PSO1 and PSO2 algorithms are BSPO7 and BPSO8 taken from Truong [20], respectively. 30 DKP01 test instances include 10 weakly correlated instances (denoted as WDKP1–WDKP10), 10 inverse strongly correlated instances (denoted as IDKP1–IDKP10), and 10 strongly correlated instances (denoted as SDKP1–SDKP10) [3].

All experiments of the proposed algorithms are running on a Laptop ASUS with an Intel (R) Core (TM) i5-8250u CPU-1.6 GHz and 8 GB DDR4 memory. The operating system is Microsoft Windows 10. The programming language is MATLAB, version R2016a.

In MFO, the number of moths is set to 50, and the search domain is the interval [1, 10]. The parameters of SecEGA are shown in [6]. The population size of SecEGA is set to 50, and the iteration is set equal to the dimension of the DKP01. For a fair comparison, the parameters for two PSO algorithms are set as the number of particles equal to 50, C_1 and C_2 are set to 2, w is linearly decreased from 0.9 to 0.4, the maximum number of iterations is set equal to the dimension of DKP01, and the stopping criterion is satisfied when the maximum number of iterations is reached. For all algorithms, the max iteration is set equal to $2 * n$. The Gap is calculated by

$$\text{Gap} = \frac{|\text{OPT} - \text{Average}|}{\text{OPT}} 100\%. \quad (14)$$

In this section, the short terms Best, Average, Worst, and StdDev are best, average, worst, and standard deviation of 30 independent runs, respectively.

Tables 3–5 summarizes the comparison among PSO1, PSO2, MFO2, SecEGA, and MFO1 based on the 6 different performance criteria on 30 independent runs including Best, Average, Worst, StdDev, the Gap, and Average rank. The MFO1 is better than PSO1, PSO2, MFO2, and SecEGA in Best, Average, and Worst for the instances of SDKP, UDKP, and WDKP except for instances of IDKP. The algorithm MFO1 archived the best rank on Average results.

The results showed that MFO1 is the best one among the 5 algorithms. Table 6 summarizes the average rank of the 5 algorithms on 30 instances. The result showed that MFO1

Table 3: Comparison of PSO1, PSO2, SecEGA, MFO1, and MFO2 on IDKP1–IDKP10.

Instance	OPT	Algorithm	Best	Average	Worst	StdDev	Gap	Rank
		PSO1	69471	68980	68252	288.0	1.6	4
		PSO2	69530	69117	68376	237.2	1.4	3
IDKP1	70106	SecEGA	68 663	68000	67 369	328.4	3.0	5
		MFO1	70106	70106	70106	0.0	0.0	1
		MFO2	70106	70104	70090	4.9	0.0	2
		PSO1	116710	116212	115370	354.2	1.7	4
		PSO2	117200	116516	115700	337.3	1.5	3
IDKP2	118268	SecEGA	114 434	113385	112 307	7446.7	4.1	5
		MFO1	118268			0.0	0.0	1
		MFO2	118268	118251	118230	19.3	0.0	2
		PSO1	234290	233653	232350	420.4	0.5	3
		PSO2	234390	232600	232600	389.6	0.4	4
IDKP3	234804	SecEGA	220 096	217982	216 313	835.8	7.2	5
		MFO1	234770	234748	234740	7.7	0.0	1
		MFO2	234700	234544	234360	92.3	0.1	2
		PSO1	280540	279714	277810	578.1	1.0	4
		PSO2	280770	279858	279110	486.7	1.0	3
IDKP4	282591	SecEGA	263 238	260425	258 922	933.4	7.8	5
		MFO1	282590	282587	282570	5.8	0.0	1
		MFO2	282470	282210	281940	132.1	0.1	2
		PSO1	333140	331595	329340	748.8	1.2	4
		PSO2	332710	331896	329280	691.2	1.1	3
IDKP5	335584	SecEGA	309 573	306878	304 881	907.2	8.6	5
		MFO1	335580	335580	335580	0.0	0.0	1
		MFO2	335280	335000	334780	107.2	0.2	2
		PSO1	450290	449287	447540	681.7	0.7	4
		PSO2	450880	449350	447890	683.3	0.7	3
IDKP6	452463		414 090	411367	408 788	1099.3	9.1	5
		MFO1	452430	452415	452390	9.7	0.0	1
		MFO2	451750	451293	450990	198.3	0.3	2
		PSO1	483180	481656	478830	944.5	1.5	3
		PSO2	483170	481578	479910	1034.9	1.5	4
IDKP7	489149	SecEGA	451 528	444316	442 133	1280.3	9.2	5
		MFO1	489150	489132	489120	9.7	0.0	1
		MFO2	488520	487889	487030	288.1	0.3	2
		PSO1	523300	520939	517720	1480.0	2.4	4
		PSO2	526240	521844	519190	1540.0	2.2	3
IDKP8	533841	SecEGA	490 494	481831	478 035	2215.7	9.7	5
		MFO1	533840	533825	533820	6.3	0.0	1
		MFO2	533050	532345	531940	284.3	0.3	2
		PSO1	515680	511908	507210	1937.0	3.1	4
		PSO2	516550	512575	509090	1727.0	2.9	3
IDKP9	528144	SecEGA	489 661	477001	471 848	3656.2	9.7	5
		MFO1	528140	528136	528120	7.2	0.0	1
		MFO2	527140	526734	526370	205.8	0.3	2
		PSO1	563960	560214	556100	2204.1	3.6	4
		PSO2	566670	562000	559540	1950.2	3.3	3
IDKP10	581244	SecEGA	535 541	521604	516 445	4265.1	10.3	5
		MFO1	581240	581233	581230	4.5	0.0	1
		MFO2	580620	579589	578870	365.0	0.3	2

Table 4: Comparison of PSO1, PSO2, SecEGA, MFO1, and MFO2 on SDKP1–SDKP10.

Instance	OPT	Algorithm	Best	Average	Worst	StdDev	Gap	Rank
SDKP1	94459	PSO1	94219	93874	93489	184.3	33.9	4
		PSO2	94205	93999	93703	130.5	34.1	3
		SecEGA	89 769	88832	87463	594.9	6.0	5
		MFO1	94286	94274	94258	12.6	34.5	1
		MFO2	94286	94222	94121	43.2	34.4	2
SDKP2	160805	PSO1	160280	159531	158810	360.1	34.9	4
		PSO2	160090	159617	159030	307.4	35.0	3
		SecEGA	153 821	152059	150753	489.4	5.4	5
		MFO1	159980	159895	159800	47.3	35.2	1
		MFO2	159840	159667	159390	93.9	35.0	2
SDKP3	238248	PSO1	237340	236389	235320	440.2	0.7	3
		PSO2	237300	236428	235620	371.4	0.7	1
		SecEGA	224 997	223580	221918	543.4	6.2	5
		MFO1	236530	236404	236310	51.8	0.7	2
		MFO2	236140	235855	235600	128.8	0.4	4
SDKP4	340027	PSO1	337960	337013	335880	585.0	19.3	1
		PSO2	337860	336811	335890	508.3	19.2	3
		SecEGA	318 510	315513	313 747	851.1	7.2	5
		MFO1	336980	336865	336800	39.0	19.2	2
		MFO2	336390	335989	335730	172.9	18.9	4
SDKP5	463033	PSO1	459780	458216	456130	728.5	36.5	3
		PSO2	459420	458086	456840	615.1	36.5	4
		SecEGA	420 238	416964	413 933	1291.7	10.0	5
		MFO1	460190	460096	460010	45.5	37.1	1
		MFO2	459240	458554	458240	225.4	36.6	2
SDKP6	466097	PSO1	462350	460874	459340	677.1	1.9	2
		PSO2	462000	460989	459690	602.6	1.9	1
		SecGA	430 738	427304	425 504	1031.1	8.3	5
		MFO1	461000	460862	460750	64.3	1.9	3
		MFO2	460060	459245	458780	226.8	1.5	4
SDKP7	620446	PSO1	614510	612746	610360	1059.2	25.3	4
		PSO2	614780	612902	610930	928.1	25.3	3
		SecEGA	561 224	556083	552 007	1926.3	10.4	5
		MFO1	615900	615756	615630	69.2	25.9	1
		MFO2	613930	613268	612870	281.1	25.4	2
SDKP8	670697	PSO1	663730	661988	659770	984.4	24.0	4
		PSO2	664250	662529	660340	992.7	24.1	2
		SecEGA	611 644	606263	603 774	1446.9	9.6	5
		MFO1	664750	664590	664450	76.0	24.5	1
		MFO2	662910	662053	661640	303.5	24.0	3
SDKP9	739121	PSO1	731830	730283	727770	1058.9	38.3	3
		PSO2	732320	730619	728570	1060.1	38.3	2
		SecEGA	674 885	667900	664 580	1614.0	9.6	5
		MFO1	731630	731502	731360	68.3	38.5	1
		MFO2	728790	728306	727650	315.5	37.9	4
SDKP10	765317	PSO1	756580	755021	753220	806.2	29.9	2
		PSO2	757430	754798	752470	1402.8	29.9	3
		SecEGA	708 935	695557	691 994	2956.1	9.1	5
		MFO1	756190	755966	755650	120.7	30.1	1
		MFO2	753740	753027	752270	336.6	29.6	4

Table 5: Comparison of PSO1, PSO2, SecEGA, MFO1, and MFO2 on WDKP1–WDKP10.

Instance	OPT	Algorithm	Best	Average	Worst	StdDev	Gap	Rank
WDKP1	83098	PSO1	82998	82764	82465	140.5	18.1	4
		PSO2	83002	82797	82549	125.7	18.1	3
		SecEGA	80014	79022	78096	473.7	4.9	5
		MFO1	82962	82894	82848	22.5	18.2	1
		MFO2	82950	82862	82798	34.8	18.2	2
WDKP2	138215	PSO1	137880	137278	136770	254.2	16.1	4
		PSO2	137860	137381	136780	247.2	16.2	3
		SecEGA	133315	132276	131337	415.6	4.3	5
		MFO1	137920	137873	137850	22.0	16.6	1
		MFO2	137890	137795	137720	40.7	16.5	2
WDKP3	256616	PSO1	256160	255362	254210	422.3	8.8	4
		PSO2	255990	255386	254670	356.3	8.8	3
		SecEGA	238331	235721	234025	873.6	8.1	5
		MFO1	255970	255891	255820	28.4	9.0	1
		MFO2	255660	255463	255260	90.0	8.8	2
WDKP4	315657	PSO1	314790	313860	313010	434.8	11.1	4
		PSO2	314750	314108	313390	399.3	11.2	3
		SecEGA	293640	290851	288764	950.1	7.9	5
		MFO1	315040	314980	314930	28.0	11.5	1
		MFO2	314630	314400	314190	125.6	11.3	2
WDKP5	428490	PSO1	426910	425683	424470	701.9	26.8	4
		PSO2	426680	425736	424520	501.1	26.9	3
		SecEGA	393617	390014	387992	1059.8	9.0	5
		MFO1	427710	427666	427620	23.0	27.4	1
		MFO2	427260	426687	426270	214.5	27.1	2
WDKP6	466050	PSO1	463690	462092	460590	641.3	2.1	4
		PSO2	463350	462284	460930	583.1	2.2	3
		SecGA	429208	425112	423269	1058.4	8.8	5
		MFO1	464880	464819	464760	28.9	2.7	1
		MFO2	463820	463485	463080	203.2	2.4	2
WDKP7	547683	PSO1	544700	542724	538860	1154.9	11.0	4
		PSO2	544730	542765	539860	922.5	11.0	3
		SecEGA	501557	496134	493845	1230.9	9.4	5
		MFO1	546500	546425	546380	29.8	11.7	1
		MFO2	545300	544705	544110	295.4	11.4	2
WDKP8	576959	PSO1	572530	570187	567530	1216.6	6.8	4
		PSO2	571870	570226	568570	825.3	6.8	3
		SecEGA	530971	523203	520350	2157.1	9.3	5
		MFO1	575590	575463	575360	45.5	7.8	1
		MFO2	574520	573672	572880	299.1	7.5	2
WDKP9	650660	PSO1	643950	641272	638210	1501.3	21.4	4
		PSO2	645140	641658	638230	1502.3	21.5	3
		SecEGA	598343	586770	583854	2315.5	9.8	5
		MFO1	648760	648672	648600	35.9	22.8	1
		MFO2	647040	646510	646030	268.1	22.4	2
WDKP10	678967	PSO1	672380	668923	666420	1429.2	15.1	3
		PSO2	671880	668830	665740	1553.1	15.1	4
		SecEGA	620230	606215	609964	3090.9	10.7	5
		MFO1	677450	677388	677330	34.5	16.5	1
		MFO2	676110	675115	674680	284.2	16.2	2

Table 6: Average rank of PSO1, PSO2, SecEGA, MFO1, and MFO2 on 30 instances.

Algorithms	Mean rank of 10 IDKP instances	Mean rank of 10 SDKP instances	Mean rank of 10 WDKP instances
PSO1	3.8	3.0	3.9
PSO2	3.2	2.5	3.1
SecEGA	4.5	4.5	4.5
MFO1	1.0	1.4	1.0
MFO2	2.0	3.1	2.0

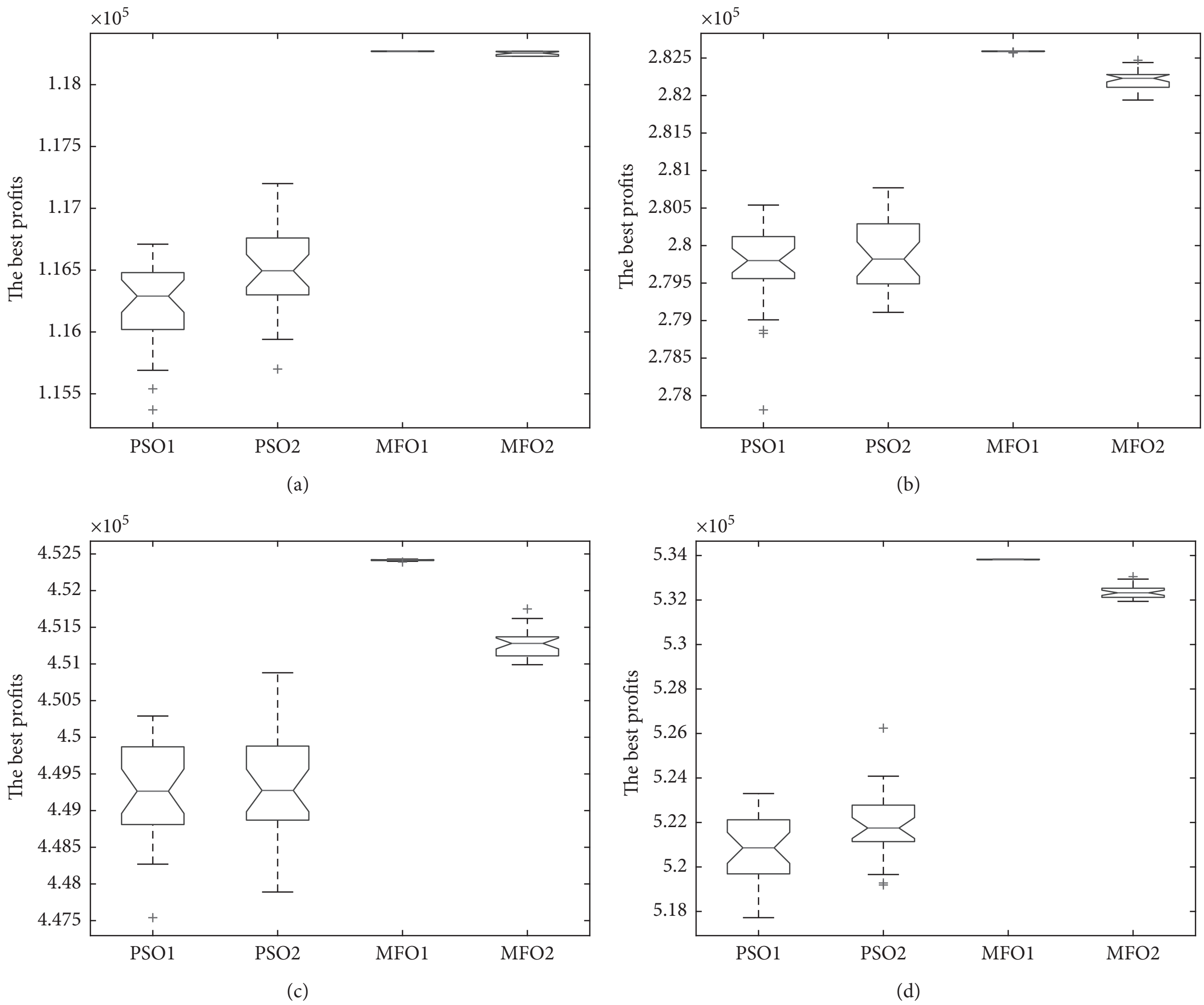
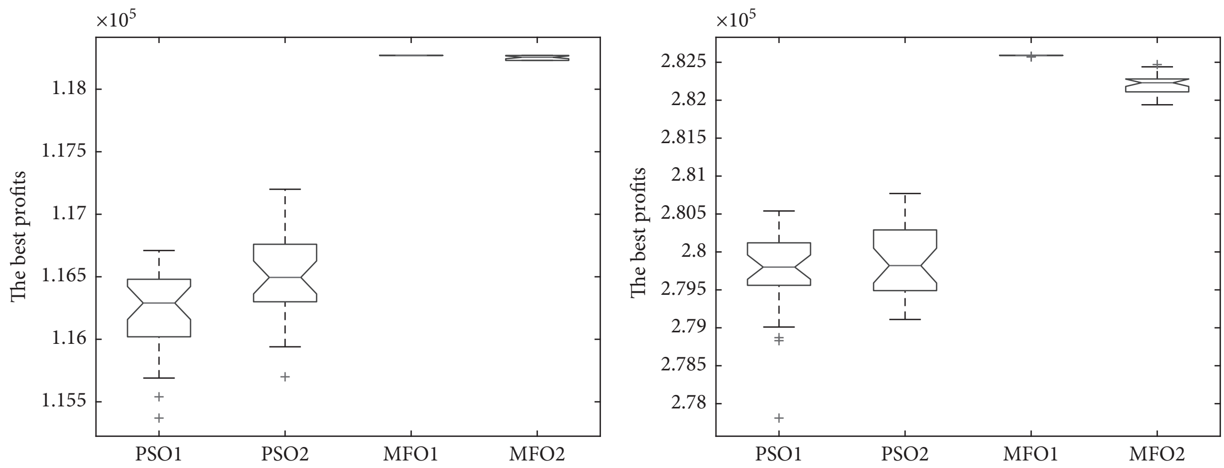
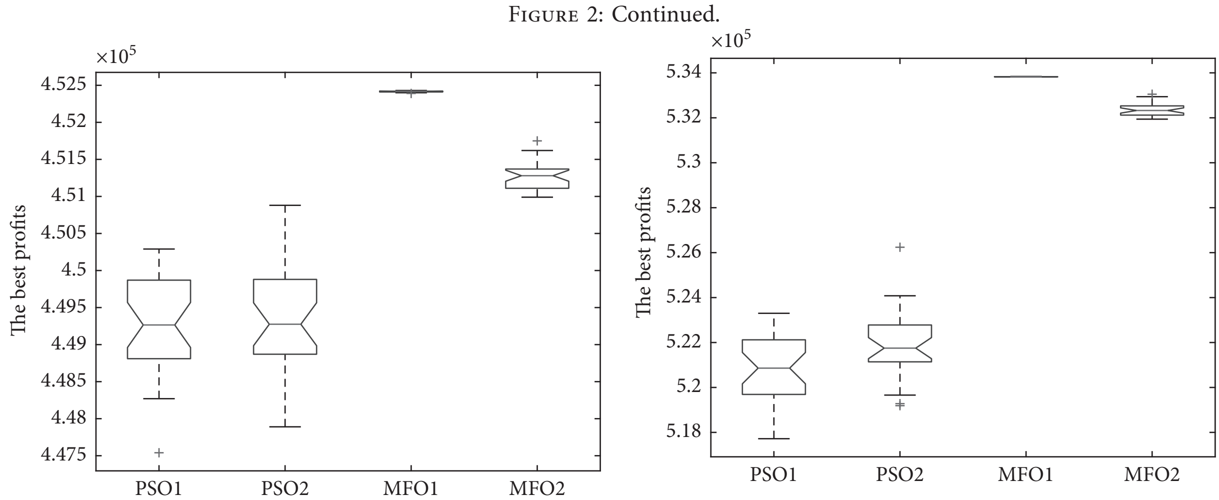


Figure 1: Box plot of four algorithms on IDKP instances. (a) IDKP2. (b) IDKP4. (c) IDKP6. (d) IDKP8.



(a) (b)

Figure 2: Continued.



(c) (d)

Figure 2: Box plot of four algorithms on WDKP instances. (a) WDKP2. (b) WDKP4. (c) WDKP6. (d) WDKP8.

achieved the average best rank in all the three test instances on average mean rank. Figure 1 demonstrates the boxplot of the four algorithms on IDKP instances: IDKP2, IDKP4, IDKP6, and IDKP8. Figure 2 demonstrates the boxplot of the four algorithms on WDKP instances: WDKP2, WDKP4, WDKP6, and WDKP8. Figure 3 demonstrates the boxplot of the four algorithms on SDKP instances: SDKP2, SDKP4, SDKP6, and SDKP8. These box plot figures showed that MFO1 obtained the best result.

Figure 4 demonstrates the convergence curves of the four algorithms on IDKP instances: IDKP2, IDKP4, IDKP6, and IDKP8. Figure 5 demonstrates the convergence curves of the four algorithms on WDKP instances: WDKP2, WDKP4, WDKP6, and WDKP8. Figure 6 demonstrates the convergence curves of the four algorithms on SDKP instances: SDKP2, SDKP4, SDKP6, and SDKP8. These convergence curves demonstrated that MFO1 has faster convergence than group algorithms PSO1, PSO2, and MFO2.

Therefore, the performance of MFO1 is better than that of the other algorithm for the DKP01 problem. From the above comparison, MFO1 showed far better result than those of PSO1, PSO2, MFO2, and SecEGA. The evidence supports that MFO1 is a potential method for solving DKP01.

4.1. Wilcoxon Rank Sum Test. With the observable measures, I am ready to prove beyond a shadow of a doubt that the outcomes are not the product of chance. The nonparametric Wilcoxon statistical test is used and the calculated p values are reported as metrics of significance as well. Any p values <0.05 evidence the statistical significant superiority of the

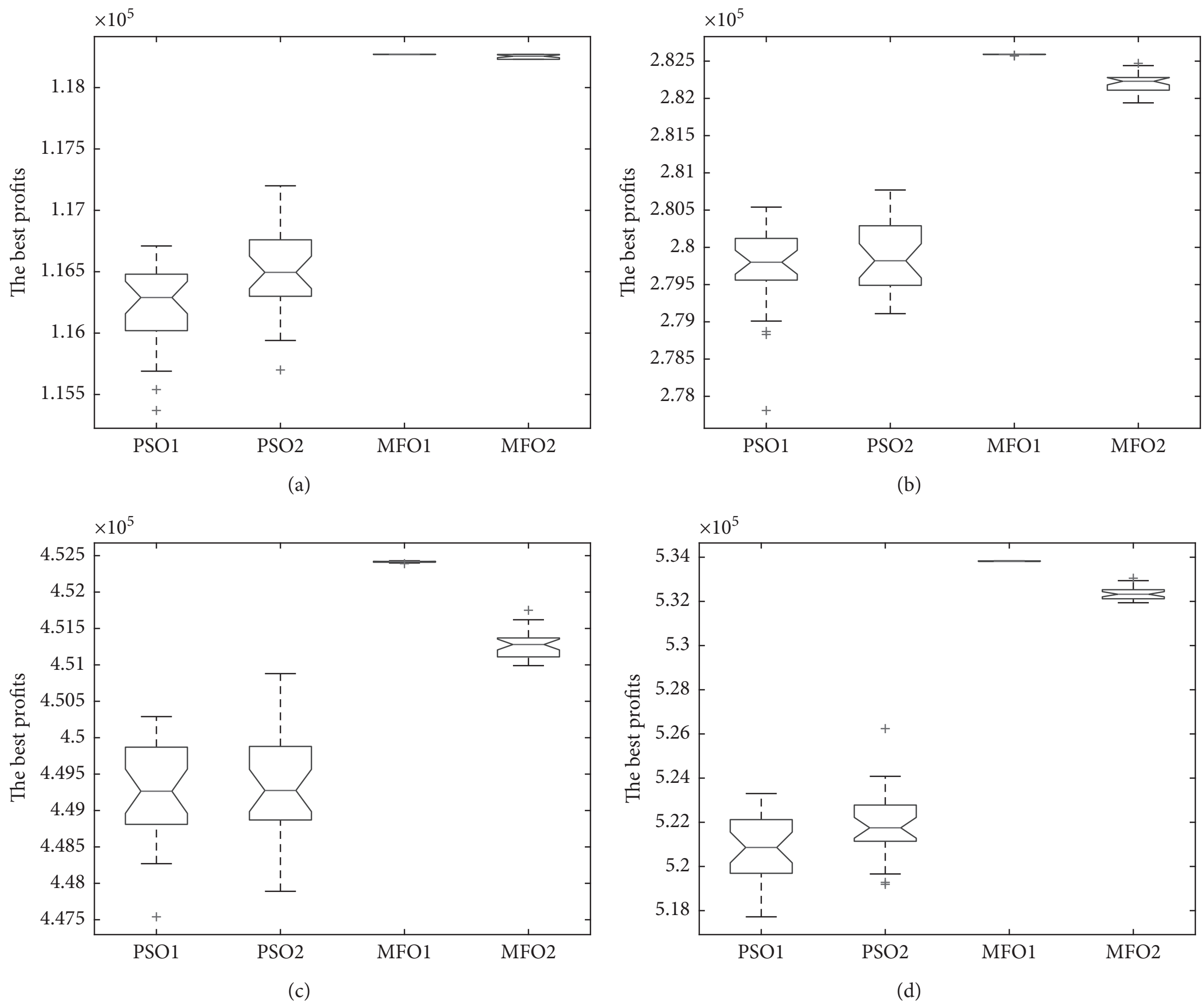


Figure 3: Box plot of four algorithms on SDKP instances. (a) SDKP2. (b) SDKP4. (c) SDKP6. (d) SDKP8.

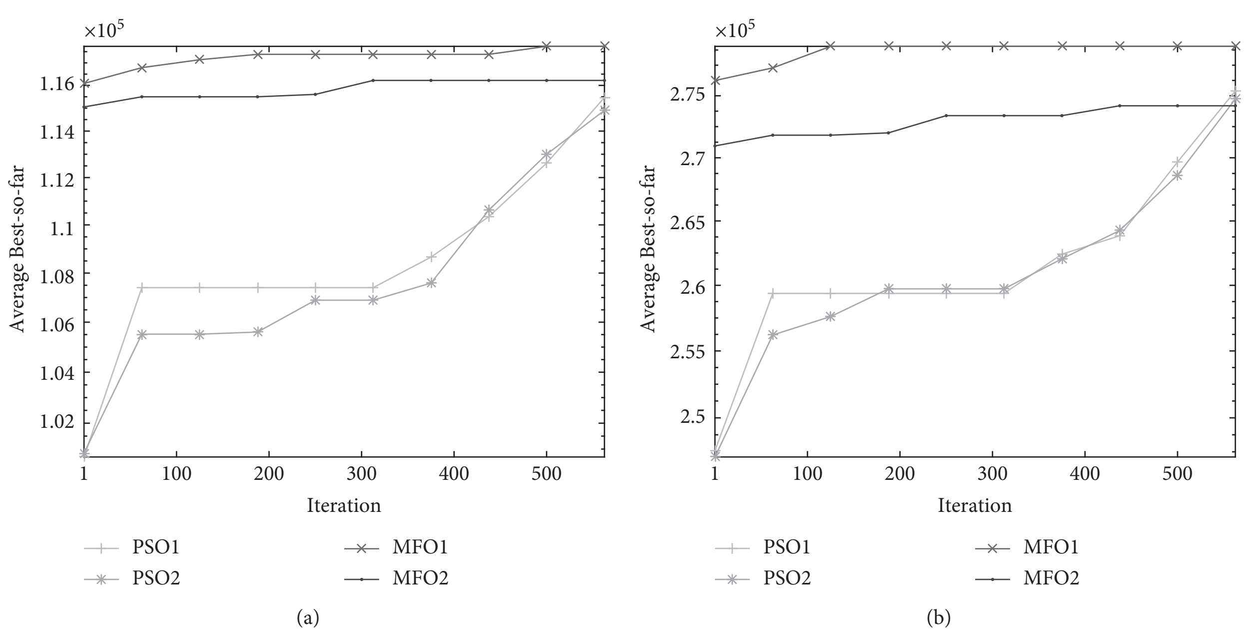
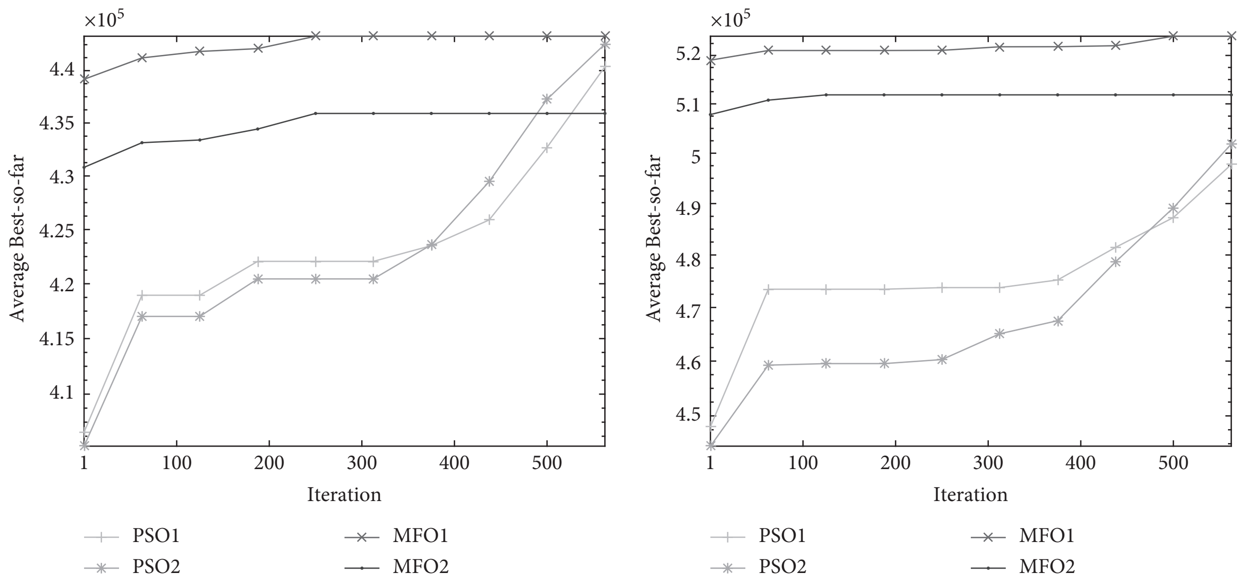
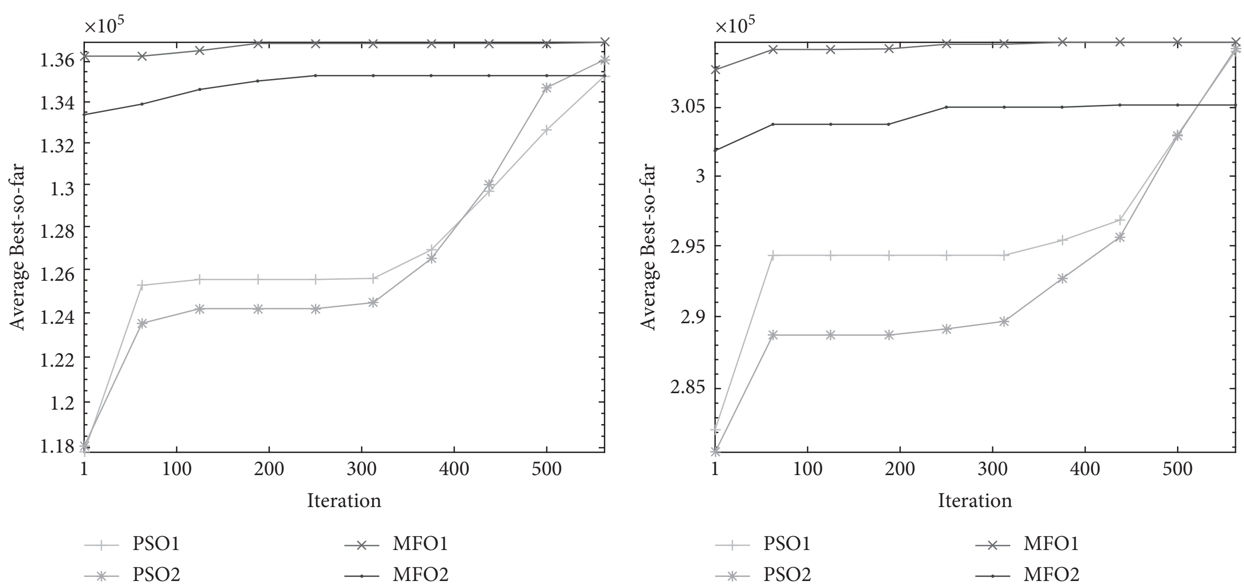


Figure 4: Continued.



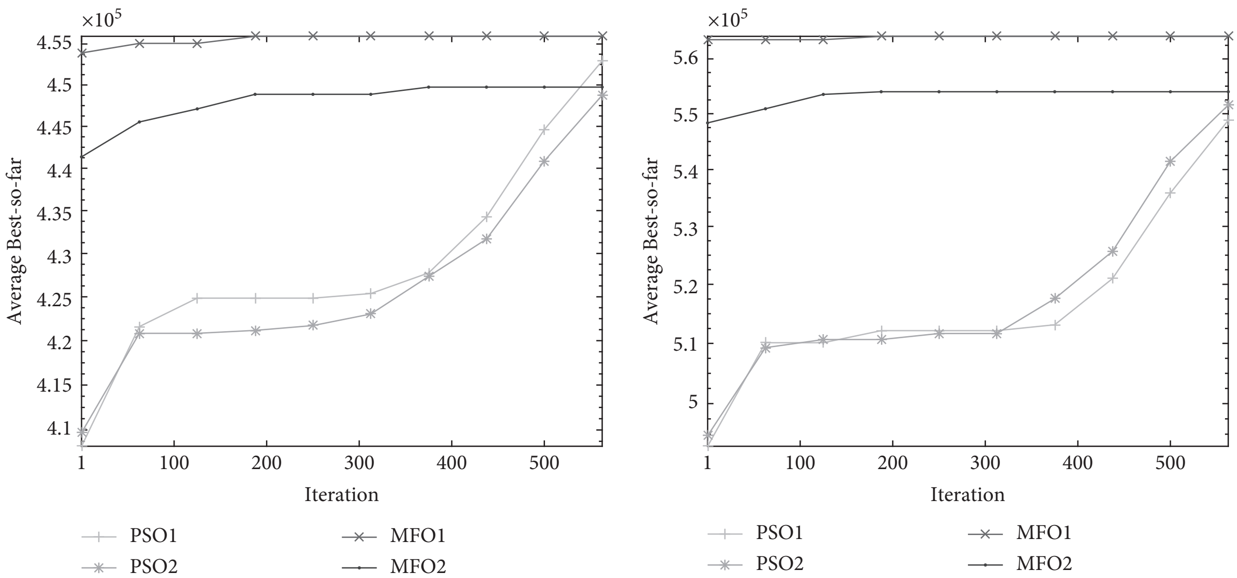
(c) (d)

Figure 4: Convergence curves of four algorithms on IDKP instances. (a) IDKP2. (b) IDKP4. (c) IDKP6. (d) IDKP8.



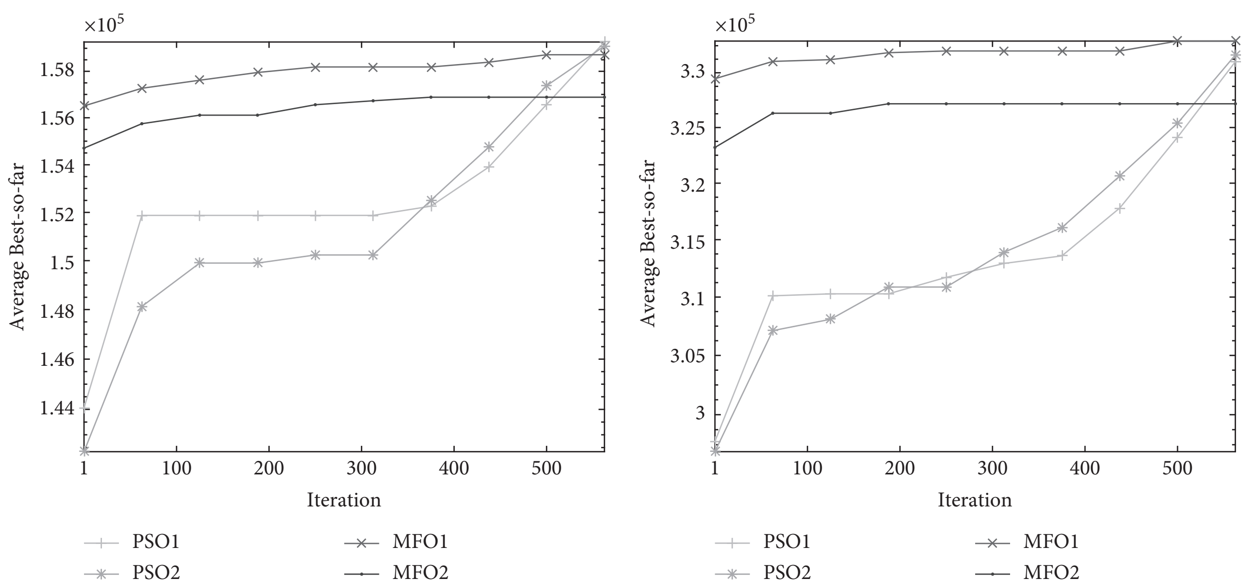
(a) (b)

Figure 5: Continued.



(c) (d)

Figure 5: Convergence curves of four algorithms on WDKP instances. (a) WDKP2. (b) WDKP4. (c) WDKP6. (d) WDKP8.



(a) (b)

Figure 6: Continued.

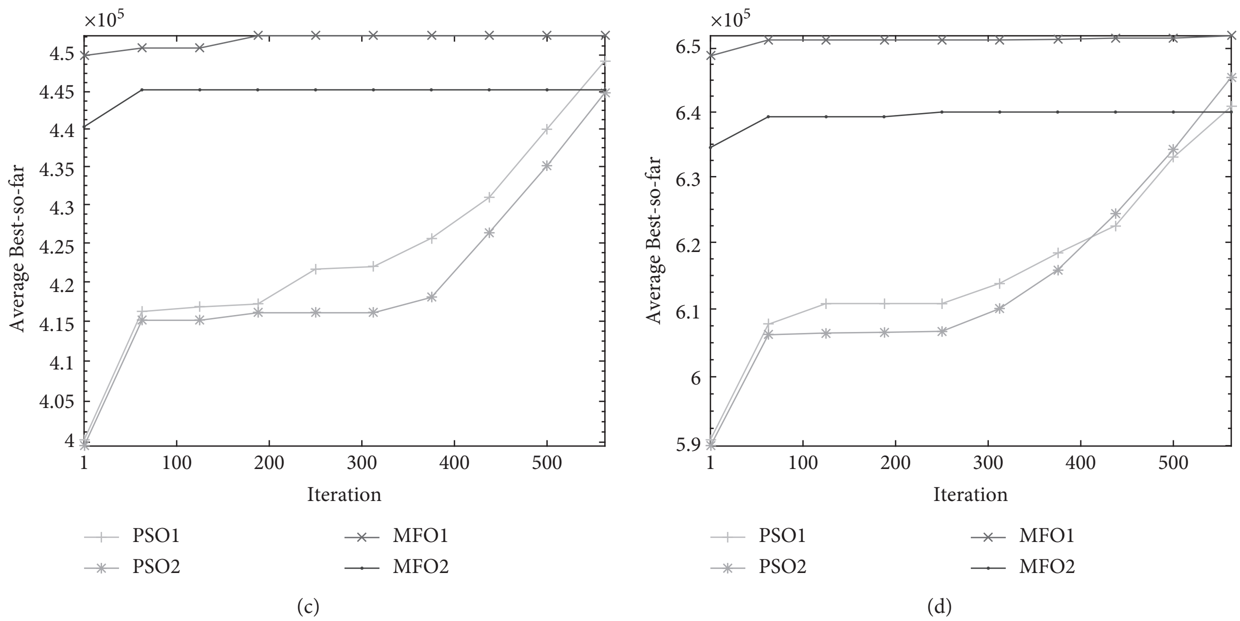


Figure 6: Convergence curves of four algorithms on SDKP instances. (a) SDKP2. (b) SDKP4. (c) SDKP6. (d) SDKP8.

Table 7: p values of the Wilcoxon rank-sum test over 30 runs.

Instance	MFO2	PSO1	PSO2
IDKP1	**0.081404**	1.21E − 12	1.21E − 12
IDKP2	1.14E − 05	1.21E − 12	1.21E − 12
IDKP3	1.86E − 11	1.87E − 11	1.87E − 11
IDKP4	6.34E − 12	6.42E − 12	6.42E − 12
IDKP5	1.2E − 12	1.21E − 12	1.21E − 12
IDKP6	1.23E − 11	1.24E − 11	1.24E − 11
IDKP7	2.25E − 11	2.25E − 11	2.25E − 11
IDKP8	1.45E − 11	1.45E − 11	1.45E − 11
IDKP9	7.77E − 12	7.8E − 12	7.8E − 12
IDKP10	8.84E − 12	8.87E − 12	8.87E − 12
SDKP1	4.45E − 07	2.11E − 11	2.11E − 11
SDKP2	5.56E − 11	4.32E − 06	0.000351
SDKP3	2.89E − 11	**0.482158**	**0.578909**
SDKP4	2.76E − 11	0.030575	0.109566
SDKP5	2.93E − 11	2.94E − 11	2.94E − 11
SDKP6	2.95E − 11	**0.164345**	**0.26713**
SDKP7	2.96E − 11	2.97E − 11	2.97E − 11
SDKP8	2.96E − 11	2.96E − 11	2.96E − 11
SDKP9	2.97E − 11	3.91E − 08	0.000431
SDKP10	2.99E − 11	1.02E − 06	3.36E − 05
WDKP1	2.68E − 05	0.000774	0.000401
WDKP2	2.19E − 10	1.87E − 10	8.09E − 11
WDKP3	2.21E − 11	2.41E − 07	6.25E − 10
WDKP4	2.71E − 11	2.74E − 11	2.73E − 11
WDKP5	2.86E − 11	2.87E − 11	2.87E − 11
WDKP6	2.87E − 11	2.86E − 11	2.88E − 11
WDKP7	2.85E − 11	2.85E − 11	2.85E − 11
WDKP8	2.9E − 11	2.92E − 11	2.92E − 11
WDKP9	2.83E − 11	2.84E − 11	2.83E − 11
WDKP10	2.92E − 11	2.93E − 11	2.93E − 11

$p > 0.05$ is indicated in bold.

results when comparing MFO1 with MFO2, PSO1, and PSO2. After all, the statistical results for 30 instances are provided in Table 7.

5. Conclusion

A moth-flame optimization algorithm that showed good searchability is combined with an effective solution presentation designed to the discounted {0-1} knapsack problem. A new encoding scheme used a shorter length binary vector to help reduce the search domain and speed up the computing time. A greedy repair procedure is used to help the algorithm have fast convergence and reduce the gap between the best-found solution and th eoptimal solution. The simulation results of 30 DKP01 instances showed that the proposed algorithms are better than the two particle swarm optimization algorithms and one genetic algorithm. In the future, some variants of the moth-flame optimization algorithm are considered for study for the discounted {0-1} knapsack problem.

Data Availability

The data used to support the findings of this study are included within the article or are made publicly available to the research community at https://www.researchgate.net/publication/336126537_Four_kinds_of_D0-1KP_instances.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author thanks Van Lang University for supporting this work.

References

[1] B. Guldan, *Heuristic and Exact Algorithms for Discounted Knapsack Problems*, University of Erlangen-Nürnberg, Erlangen, Germany, 2007.

[2] A. Rong, J. R. Figueira, and K. Klamroth, “Dynamic programming based algorithms for the discounted {0-1} knapsack problem,” *Applied Mathematics and Computation*, vol. 218, no. 12, pp. 6921–6933, 2012.

[3] Y. He, X. Wang, W. Li, X. Zhang, and Y. Chen, “Research on genetic algorithms for discounted 0–1 knapsack problem,” *Chinese Journal of Computers*, vol. 39, no. 12, pp. 2614–2630, 2016.

[4] Y.-C. He, X.-Z. Wang, Y.-L. He, S.-L. Zhao, and W.-B. Li, “Exact and approximate algorithms for discounted {0-1} knapsack problem,” *Information Sciences*, vol. 369, pp. 634–647, 2016.

[5] J. Kenedy and R. Eberhart, “A discrete binary version of the particle swarm optimization,” *Computational cybernatics and simulation*, vol. 5, no. 1, pp. 4104–4108, 1997.

[6] Y. He, X. Wang, and S. Gao, “Ring theory-based evolutionary algorithm and its application to *D*{0-1} KP,” *Applied Soft Computing*, vol. 77, pp. 714–722, 2019.

[7] Y. H. Feng and G. G. Wang, “Binary moth search algorithm for discounted 0-1 knapsack problem,” *IEEE Access*, vol. 6, pp. 10708–10719, 2018.

[8] C. Wu, J. Zhao, Y. Feng, and M. Lee, “Solving discounted {0-1} knapsack problems by a discrete hybrid teaching-learning-based optimization algorithm,” *Applied Intelligence*, vol. 50, no. 6, pp. 1872–1888, 2020.

[9] S. Mirjalili, “Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm,” *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[10] C. Yu, A. A. Heidari, and H. Chen, “A quantum-behaved simulated annealing algorithm-based moth-flame optimization method,” *Applied Mathematical Modelling*, vol. 87, pp. 1–19, 2020.

[11] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, “An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, Article ID e3770, 2020.

[12] M. Abd Elaziz, D. Yousri, and S. Mirjalili, “A hybrid harris hawks-moth-flame optimization algorithm including fractional-order chaos maps and evolutionary population dynamics,” *Advances in Engineering Software*, vol. 154, Article ID 102973, 2021.

[13] S. Sapre and S. Mini, “A differential moth flame optimization algorithm for mobile sink trajectory,” *Peer-to-Peer Networking and Applications*, vol. 14, no. 1, pp. 44–57, 2021.

[14] M. A. Tolba, A. A. Z. Diab, V. N. Tulsky, and A. Y. Abdelaziz, “LVCI approach for optimal allocation of distributed generations and capacitor banks in distribution grids based on moth-flame optimization algorithm,” *Electrical Engineering*, vol. 100, no. 3, pp. 2059–2084, 2018.

[15] N. Jangir, M. H. Pandya, I. N. Trivedi, R. Bhesdadiya, P. Jangir, and A. Kumar, “Moth-flame optimization algorithm for solving real challenging constrained engineering optimization problems,” in *Proceedings of the 2016 IEEE Students’ Conference on Electrical, Electronics and Computer Science (SCEECS)*, pp. 1–5, IEEE, Bhopal, India, March 2016.

[16] D. Allam, D. A. Yousri, and M. B. Eteiba, “Parameters extraction of the three diode model for the multi-crystalline solar cell/module using moth-flame optimization algorithm,” *Energy Conversion and Management*, vol. 123, pp. 535–548, 2016.

[17] G. I. Sayed, A. E. Hassanien, T. M. Nassef, and J.-S. Pan, “Alzheimer’s disease diagnosis based on moth flame optimization,” in *Proceedings of the International Conference on Genetic and Evolutionary Computing*, pp. 298–305, Springer, Fuzhou, China, November 2016.

[18] S. Gope, S. Dawn, A. K. Goswami, and P. K. Tiwari, “Profit maximization with integration of wind farm in contingency constraint deregulated power market using moth flame optimization algorithm,” in *Proceedings of the 2016 IEEE Region 10 Conference (TENCON)*, pp. 1462–1466, IEEE, Singapore, November 2016.

[19] H. Zhao, H. Zhao, and S. Guo, “Using GM (1, 1) optimized by MFO with rolling mechanism to forecast the electricity consumption of inner Mongolia,” *Applied Sciences*, vol. 6, no. 1, p. 20, 2016.

[20] T. K. Truong, “Different transfer functions for binary particle swarm optimization with a new encoding scheme for discounted 0-1 knapsack problem,” *Mathematical Problems in Engineering*, vol. 2021, Article ID 2864607, 17 pages, 2021.

[21] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the MHS'95 Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, December 1995.

[23] Z. Li and N. Li, "A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem," in *Proceedings of the 2009 Chinese Control and Decision Conference*, pp. 3042–3047, Guilin, China, June 2009.

[24] Y. Feng, G. G. Wang, W. Li, and N. Li, "Multi-strategy monarch butterfly optimization algorithm for discounted {0-1} knapsack problem," *Neural Computing & Applications*, vol. 30, no. 10, pp. 3019–3036, 2018.

[25] H. Zhu, Y. He, X. Wang, and E. C. C. Tsang, "Discrete differential evolutions for the discounted {0-1} knapsack problem," *International Journal of Bio-Inspired Computation*, vol. 10, no. 4, p. 219, 2017.