Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 2864607, 17 pages
https://doi.org/10.1155/2021/2864607

*Research Article*

# Different Transfer Functions for Binary Particle Swarm Optimization with a New Encoding Scheme for Discounted {0-1} Knapsack Problem

**Tung Khac Truong**

*Faculty of Information Technology, Van Lang University, 45 Nguyen Khac Nhu Street, Co Giang Ward, District 1, Ho Chi Minh City, Vietnam*

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

Received 27 June 2020; Revised 8 April 2021; Accepted 9 April 2021; Published 24 April 2021

Academic Editor: Luigi Rodino



The discounted {0-1} knapsack problem (DKP01) is a kind of knapsack problem with group structure and discount relationships among items. It is more challenging than the classical 0-1 knapsack problem. In this paper, we study binary particle swarm optimization (PSO) algorithms with different transfer functions and a new encoding scheme for DKP01. An effective binary vector with shorter length is used to represent a solution for new binary PSO algorithms. Eight transfer functions are used to design binary PSO algorithms for DKP01. A new repair operator is developed to handle isolation solution while improving its quality. Finally, we conducted extensive experiments on four groups of 40 instances using our proposed approaches. The experience results show that the proposed algorithms outperform the previous algorithms named FirEGA and SecEGA . Overall, the proposed algorithms with a new encoding scheme represent a potential approach for solving the DKP01.

## 1. Introduction

The discounted 0-1 knapsack problem (DKP01) is a new kind of knapsack problem proposed by Guldan [1]. This problem has an important role in the real-world business process. It is a part of many key problems such as investment decision-making, mission selection, and budget control. An exact algorithm based on dynamic programming for the DKP01 is first proposed in [1]. An approach combining dynamic programming with the core of the DKP01 to solve it is studied in [2]. Two algorithms based on genetic algorithm for DKP01 are named FirEGA and SecEGA in [3].

Assume that there are $n$ groups. Each group contains three items. Consider a given set of $3^*n$ items; each of them has an integer weight $w_i$ and an integer profit $p_i$. The problem is to select a subset from the set of $3^*n$ items in $n$ groups such that the overall profit is maximized without exceeding a given weight capacity $C$. We cannot choose more than 1 item in each group. It is an NP-Hard problem and hence it does not have a polynomial time algorithm unless $P$ = NP. The problem may be mathematically modelled as follows:

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

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

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

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

where $x_{3i}$, $x_{3i+1}$, and $x_{3i+2}$ represent whether the items $3i$, $3i+1$, and $3i+2$ are put into the knapsack: $x_j = 0$ indicates that the item $j$ ($j = 0, 1, \ldots, 3n-1$) is not in knapsack, while $x_j = 1$ indicates that the item $j$ is in 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 equations (2) and (3), it is a feasible solution of DKP01.

Recently, He et al. 2016 also had a detailed study of the algorithms of the DKP01 and proposed brand new deterministic algorithm and approximation algorithms. A new exact algorithm and two approximation algorithms with a greedy repair operator were proposed to solve DKP01 [4]. An algorithm based on PSO is named GBPSO using discrete particle swarm optimization [5]. An evolution algorithm combines with ring theory to solve DKP01 [6]. Multistrategy monarch butterfly optimization algorithm, Binary Moth Search algorithm [7], and hybrid teaching-learning-based optimization algorithm [8] are proposed for DKP01. Binary PSO is also developed to solve many optimization problems such as scheduling of appliances in smart homes [9], fault diagnosis of bearing [10], operation cost reduction in unit commitment problem [11], channel selection in the EEG signals and its application to speller systems [12], wireless sensor networks [13], transmission expansion planning considering $n-1$ security criterion [14], a bare-bones multiobjective particle swarm optimization algorithm for environmental/economic dispatch [15], multiobjective particle swarm optimization for feature selection with fuzzy cost [16], multiobjective particle swarm optimization approach for cost-based feature selection in classification [17], and variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data [18].

Many algorithms were proposed to solve DKP01, and each of them has its advantages and disadvantages. Further study on this problem is necessary. In this paper, we study binary particle swarm optimization (PSO) algorithms with different transfer function and a new encoding scheme for DKP01. An effective binary vector with $2^*$-dimensional length is used to represent an individual the proposed binary PSO strategies. Eight types of transfer functions are used to design binary PSO algorithms for DKP01. Finally, we conducted extensive experiments on four groups of 40 instances using our proposed approaches. The experience results demonstrate that the proposed algorithms outperform the genetic algorithm and original binary PSO in solving 40 DKP01 instances. The main contributions of this work can be listed as follows:

(i) Binary particle swarm optimization algorithms with difference binary transfer functions and new solution presentation are proposed to solve the discounted {0-1} knapsack problem.

(ii) A new encoding scheme has shorter binary vector (the length is $2n$ compared to $3n$) and also automatically satisfies the constraint that chose the most one item in each group.

(iii) A new repair operator is developed to handle isolation solution while improving its quality.

The rest of this paper is organized as follows: Section 2 presents previous algorithms for KP01. Section 3 presents the binary particle swarm optimization 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 implements a population of particles. A population of particles is randomly created initially [19, 20]. The standard molecule swarm optimizer keeps up a swarm of molecule that speaks to the potential arrangements to issue on hand. Suppose that the search space is $D$-dimensional, and the position of the $i$th particle of the swarm can be described using 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 the $i$th particle is named as $p_i = (p_{i1}, \ldots, p_{id}, \ldots, p_{iD})$. In substance, the direction of each molecule is upgraded concurring to its claim flying experience as well as to that of the finest molecule within the swarm. The fundamental PSO calculation can be depicted as

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

$$x_{i,d}^{k+1} = x_{i,d}^{k} + v_{i,d}^{k+1}, \tag{6}$$

where $v_{i,d}^{k}$ is the $d$th dimension velocity of particle $i$ in cycle $k$; $x_{i,d}^{k}$ is the $d$th dimension position of particle $i$ in cycle $k$; $p_{i,d}^{k}$ is the $d$th dimension position of personal best (pbest) of particle $i$ in cycle $k$; $p_{g,d}^{k}$ is the $d$th dimension position of global best particle (gbest) in cycle $k$; $w$ is the inertia weight; $c_1$ is the cognitive weight and $c_2$ is a social weight; $r_1$ and $r_2$ are two random values uniformly distributed in the range of [0, 1] [21].

The pseudocode of the PSO is given in Algorithm 1.

*2.2. Binary Particle Swarm Optimization.* The binary particle swarm optimization algorithm was introduced by Bansal and Deep to allow the PSO algorithm to operate in binary problem spaces [21–23]. It uses the concept of velocity as a probability that a bit (position) takes on one or zero. In the BPSO, equation (5) for updating the velocity remains unchanged, but equation (6) for updating the position is redefined by the rule using the two following equations:

$$x_{i,d}^{k+1} = \begin{cases} 0, & \text{if rand} \geq S\left(v_{i,d}^{k+1}\right), \\ 1, & \text{if rand} < S\left(v_{i,d}^{k+1}\right), \end{cases} \tag{7}$$

$$x_{i,d}^{k+1} = \begin{cases} \text{not}\left(x_{i,d}^{k}\right), & \text{if rand} \geq S\left(v_{i,d}^{k+1}\right), \\ x_{i,d}^{k}, & \text{if rand} < S\left(v_{i,d}^{k+1}\right), \end{cases} \tag{8}$$

where $S$ (.) is the sigmoid function for transforming the velocity to the probability as the following expression:

**Input**: Initial parameters
**Output**: optimal solution
(1) **for** *each particle* **do**
(2) Initialize particle
(3) **while** *stop condition is not met* **do**
(4) **for** *each particle* **do**
(5) Evaluate objective function
(6) **if** *the objective function value* ≤ *pBest* **then**
(7) current value is replace by **pBest**
(8) Calculate the **gBest** (the global best value)
(9) **for** *each particle* **do**
(10) Calculate particle velocity by equation (5)
(11) Update particle position by equation (6)

Algorithm 1: PSO algorithm.

$$
\begin{aligned}
S_1\left(v_{i,d}^{k+1}\right) &= \frac{1}{1+e^{-2v_{i,d}^{k+1}}},\\
S_2\left(v_{i,d}^{k+1}\right) &= \frac{1}{1+e^{-v_{i,d}^{k+1}}},\\
S_3\left(v_{i,d}^{k+1}\right) &= \frac{1}{1+e^{1/2v_{i,d}^{k+1}}},\\
S_4\left(v_{i,d}^{k+1}\right) &= \frac{1}{1+e^{(1/3)v_{i,d}^{k+1}}},\\
S_5\left(v_{i,d}^{k+1}\right) &= \left|\operatorname{erf}\left(\frac{\sqrt{\pi}}{2}v_{i,d}^{k+1}\right)\right|,\\
S_6\left(v_{i,d}^{k+1}\right) &= \left|\tanh\left(v_{i,d}^{k+1}\right)\right|,\\
S_7\left(v_{i,d}^{k+1}\right) &= \left|\frac{1}{\sqrt{1+v_{i,d}^{k+12}}}\right|,\\
S_8\left(v_{i,d}^{k+1}\right) &= \left|\frac{2}{\pi}\arctan\left(\frac{\pi}{2}v_{i,d}^{k+1}\right)\right|.
\end{aligned}
\tag{9}
$$

In this section, we propose 8 binary algorithms based on BSO named BPSO1 to BPSO8. The algorithm $\text{BPSO}_x$ uses transfer function $S_x$ (where $x$ is integer in [1, 8]), and BPSO1–BPSO4 use formula (7), while BPSO5–BPSO8 use formula (8) to calculate binary vector $X$.

## 3. Proposed Binary Particle Swarm Optimization for DKP01

*3.1. Solution Presentation.* At present, there are two methods to encode a solution which are using a binary vector with length equal to the $3^*n$-dimensional problem [3, 7, 24, 25], and the other method is an integer vector with length equal to number of groups $n$ to present a solution [8]. Each encoding scheme has its advantages and disadvantages. The binary scheme has an advantage when many metaheuristics have a good design to be directly applied to solution.

In this paper, a new binary encode scheme with length $2^*n$ is used to present the solution. The advantage of this encode scheme is shorter length and it automatically satisfies constraint 2. The new binary encoding scheme is presented in Table 1.

*3.2. Repair Function.* The new encoding scheme automatically satisfies constraint 2. To handle constraint 3 and improve the quality of solution, a new repair based on the idea in [3] is proposed. The advantage of this repair procedure is the balance between CPU time cost and not getting stuck in local optima. The items are sorted according to the profit-to-weight ratio $p_i/w_i$ $(i = 1, 2, \ldots, n)$ 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 increasing order of $p_j/w_j$ and drops item from knapsack if feasibility is violated. The first phase (called optimization phase) examines each variable in increasing order of $p_j/w_j$ and adds item to knapsack as long as feasibility is not violated. The aim of the repair phase is to obtain a feasible solution from an infeasible solution, while the optimization phase seeks to improve the fitness of a feasible solution. The pseudocode for the repair operator is given in Algorithm 2. The time complexity of repair operator is $O(n)$.

The overall pseudocode of the BPSO algorithms for DKP01 is given in Algorithm 3.

## 4. Simulation Results

In this paper, the experience results of eight BPSO algorithms are compared to find out the best one among them to solve DKP01. The best proposed BPSO is used to compare the results of two algorithms taken from [6] named FirEGA and SecEGA. 40 DKP01 test instances are 10 uncorrelated instances (denoted as UDKP1–UDKP10), 10 weakly correlated instances (denoted as WDKP1–WDKP10), 10 inverse strongly correlated instances (denoted as

Table 1: New binary encoding scheme.

| No. | Binary encoding scheme | Meaning |
|---|---|---|
| 1 | 00 | No item of group is chosen |
| 2 | 01 | First item of group is chosen |
| 3 | 10 | Third item of group is chosen |
| 4 | 11 | Fourth item of group is chosen |

**Input**: Solution $x$, value vector $v$, weight vector $w$, index vector id, and knapsack capacity $C$.
**Output**: Solution $x$
(1) % Repair phase
(2) $n \longleftarrow$ length $(x)/2$ **for** $i = 1:3^{*}n$ **do**
(3) $k \longleftarrow$ floor $((\mathrm{id}\ (i) - 1)/3)$;
(4) $r \longleftarrow$ mod $(\mathrm{id}\ (i) - 1, 3)$;
(5) **if** $x(2k+1) = 0 \wedge x(2k+1) = 1 \wedge r = 0 \wedge fw + w(\mathrm{id}(i)) \le C$ **then**
(6) $fv \longleftarrow fv + v(\mathrm{id}(i))$
(7) $fw \longleftarrow fw + w(\mathrm{id}(i))$
(8) **else if** $x(2k+1) = 0 \wedge x(2k+1) = 1 \wedge r = 0 \wedge fw + w(i\ d(i)) > C$ **then**
(9) $x\ (2k+1) \longleftarrow 0$
(10) $x\ (2k+2) \longleftarrow 0$
(11) **if** $x(2k+1) = 1 \wedge x(2k+1) = 0 \wedge r = 1 \wedge fw + w(\mathrm{id}(i)) \le C$ **then**
(12) $fv \longleftarrow fv + v(\mathrm{id}(i))$
(13) $fw \longleftarrow fw + w(\mathrm{id}(i))$
(14) **else if** $x(2k+1) = 1 \wedge x(2k+1) = 0 \wedge r = 1 \wedge fw + w(\mathrm{id}(i)) > C$ **then**
(15) $x\ (2k+1) \longleftarrow 0$
(16) $x\ (2k+2) \longleftarrow 0$
(17) **if** $x(2k+1) = 1 \wedge x(2k+1) = 1 \wedge r = 2 \wedge fw + w(\mathrm{id}(i)) \le C$ **then**
(18) $fv \longleftarrow fv + v(\mathrm{id}(i))$
(19) $fw \longleftarrow fw + w(\mathrm{id}(i))$
(20) **else if** $x(2k+1) = 1 \wedge x(2k+1) = 1 \wedge r = 2 \wedge fw + w(\mathrm{id}(i)) > C$ **then**
(21) $x\ (2k+1) \longleftarrow 0$
(22) $x\ (2k+2) \longleftarrow 0$
(23) % Optimization phase
(24) **for** $i = 1{:}\ 3^{*}n$ **do**
(25) $k \longleftarrow$ floor $((\mathrm{id}\ (i) - 1)/3)$;
(26) $r \longleftarrow$ mod $(\mathrm{id}\ (i) - 1, 3)$;
(27) **if** $x(2k+1) = 0 \wedge x(2k+1) = 0 \wedge fw + w(\mathrm{id}(i)) \le C$ **then**
(28) $fv \longleftarrow fv + v(\mathrm{id}(i))$
(29) $fw \longleftarrow fw + w(\mathrm{id}(i))$
(30) **if** $r = 0$ **then**
(31) $x\ (2k+1) \longleftarrow 0$; $x\ (2k+2) \longleftarrow 1$
(32) **if** $r = 1$ **then**
(33) $x\ (2k+1) \longleftarrow 1$; $x\ (2k+2) \longleftarrow 0$
(34) **if** $r = 2$ **then**
(35) $x\ (2k+1) \longleftarrow 1$; $x\ (2k+2) \longleftarrow 1$
(36) **return** $x$

Algorithm 2: Repair operator.

IDKP1–IDKP10), and 10 strongly correlated instances (denoted as SDKP1–SDKP10) [3].

All experiments of the proposed algorithms are performed on a Dell Vostro 5471 VTI5207W laptop with an Intel (R) Core (TM) i5-8250u CPU-1.6 GHz and 8 GB DDR3 memory. The operating system is Microsoft Windows 10. All the algorithms are implemented using MATLAB R2018a.

The parameters of FirEGA and SecEGA are shown in [6]. The population sizes of FirEGA and SecEGA are set to 50, and the iteration is set to be equal to the dimension of the DKP01. For a fair comparison, the parameters for BPSO algorithms are set as follows: the number of particles is 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 to be equal

**Input**: Initial parameters
**Output**: Optimal solution
(1) **for** *each particle* **do**
(2) Initialize particle
(3) **while** *stop condition is not met* **do**
(4) **for** *each particle* **do**
(5) Evalute objective function
(6) **if** *the objective function value* ≤ *pBest* **then**
(7) current value is replace by **pBest**
(8) Caculate the **gBest** (the global best value)
(9) **for** *each particle* **do**
(10) Calculate particle velocity by equation (5)
(11) Caculate $S(.)$ using a transfer function
(12) Update particle position by equations (7) or (8)
(13) Apply repair operator for current particle position.

Algorithm 3: Overall pseudocode of BPSO algorithms for DKP01.

Table 2: Comparison of 8 BPSO algorithms on IDKP1–IDKP10.

| | | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| IDKP1 | Best | 67982 | 65735 | 66130 | 65215 | 69940 | 69944 | 69905 | **69950** |
| | AVE | 66360.4 | 65219.1 | 64523.1 | 64206.4 | 69650.5 | 69689.3 | 69652.8 | **69761.0** |
| | Worst | 65724 | 64686 | 63911 | 63583 | 69345 | 69069 | 69348 | **69521** |
| | Std. dev | 450.89 | 285.19 | 468.09 | 367.69 | 127.49 | 184.26 | 151.25 | 109.15 |
| | Gap | 5.34 | 6.97 | 7.96 | 8.42 | 0.65 | 0.59 | 0.65 | 0.49 |
| IDKP2 | Best | 111220 | 109980 | 107920 | 106690 | 117840 | 117700 | 117960 | **118000** |
| | AVE | 109885.7 | 108308.3 | 106567.0 | 105800.3 | 117157.7 | 117265.3 | 117237.7 | **117418.0** |
| | Worst | 108670 | 107420 | 105580 | 104920 | 116530 | 116690 | 116880 | **117040** |
| | Std. dev | 526.83 | 583.77 | 527.62 | 375.53 | 317.69 | 220.23 | 250.72 | 228.83 |
| | Gap | 7.09 | 8.42 | 9.89 | 10.54 | 0.94 | 0.85 | 0.87 | 0.72 |
| IDKP3 | Best | 223100 | 220370 | 217660 | 217270 | 234550 | **234600** | 234510 | 234540 |
| | AVE | 221673.7 | 219018.7 | 216489.0 | 215686.7 | 234234.0 | 234193.3 | 234192.3 | **234253.7** |
| | Worst | 220330 | 218260 | 215340 | 214360 | **233870** | 233740 | 233820 | 233800 |
| | Std. dev | 655.53 | 522.10 | 621.85 | 588.95 | 170.00 | 194.82 | 196.90 | 192.52 |
| | Gap | 5.59 | 6.72 | 7.80 | 8.14 | 0.24 | 0.26 | 0.26 | 0.23 |
| IDKP4 | Best | 262250 | 261600 | 256030 | 254560 | 281630 | 281460 | **281860** | 281560 |
| | AVE | 260578.7 | 257426.7 | 254213.0 | 253443.3 | 280915.7 | 280937.0 | 280926.7 | **280953.3** |
| | Worst | 259350 | 255980 | 253070 | 251990 | 280230 | 280210 | **280310** | 280220 |
| | Std. dev | 680.61 | 1036.33 | 669.28 | 545.32 | 370.76 | 298.83 | 381.13 | 328.28 |
| | Gap | 7.79 | 8.90 | 10.04 | 10.31 | 0.59 | 0.59 | 0.59 | 0.58 |
| IDKP5 | Best | 306710 | 304080 | 300290 | 299740 | 334080 | **334460** | 334370 | 334330 |
| | AVE | 305651.3 | 302176.0 | 298676.3 | 297292.7 | 333196.0 | **333540.0** | 333450.7 | 333380.7 |
| | Worst | 304530 | 300500 | 297810 | 296300 | 331670 | 332140 | 331700 | **332480** |
| | Std. dev | 627.16 | 837.40 | 572.50 | 822.63 | 558.47 | 545.39 | 595.10 | 479.64 |
| | Gap | 8.92 | 9.96 | 11.00 | 11.41 | 0.71 | 0.61 | 0.64 | 0.66 |
| IDKP6 | Best | 424860 | 420840 | 414340 | 414270 | **451530** | 451120 | 451180 | 451390 |
| | AVE | 422738.0 | 418002.7 | 412954.3 | 410614.0 | **450461.3** | 450450.0 | 450227.7 | 450455.0 |
| | Worst | 420920 | 416490 | 411800 | 409440 | **449550** | 449230 | 449230 | 448980 |
| | Std. dev | 1094.93 | 1134.30 | 671.58 | 922.85 | 476.18 | 491.61 | 468.12 | 563.12 |
| | Gap | 6.57 | 7.62 | 8.73 | 9.25 | 0.44 | 0.44 | 0.49 | 0.44 |
| IDKP7 | Best | 445470 | 440660 | 437100 | 434590 | 485500 | 485610 | **485970** | 485920 |
| | AVE | 443578.0 | 438952.7 | 434153.7 | 432156.0 | **484351.7** | 484123.3 | 484322.3 | 484205.7 |
| | Worst | 441760 | 437450 | 432760 | 430820 | **482930** | 482580 | 482450 | 482400 |
| | Std. dev | 973.40 | 1031.28 | 1060.95 | 965.40 | 741.76 | 739.59 | 938.96 | 879.02 |
| | Gap | 9.32 | 10.26 | 11.24 | 11.65 | 0.98 | 1.03 | 0.99 | 1.01 |

Table 2: Continued.

| | | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| IDKP8 | Best | 476350 | 469590 | 465180 | 462400 | 528810 | 528720 | **528910** | 528450 |
| | AVE | 473369.7 | 467065.0 | 462243.3 | 460488.7 | **526964.3** | 526535.3 | 526150.3 | 526356.7 |
| | Worst | 470860 | 465310 | 459400 | 459070 | 524250 | **524640** | 523860 | 524150 |
| | Std. dev | 1361.11 | 1059.54 | 1294.50 | 853.73 | 1377.91 | 1070.18 | 1437.35 | 1268.72 |
| | Gap | 11.33 | 12.51 | 13.41 | 13.74 | 1.29 | 1.37 | 1.44 | 1.40 |
| IDKP9 | Best | 467480 | 460890 | 456610 | 452870 | 521210 | 521390 | 520960 | **522140** |
| | AVE | 463983.7 | 458418.0 | 453862.3 | 451666.0 | 516689.3 | 517894.3 | 518507.0 | **518834.7** |
| | Worst | 460970 | 456540 | 451430 | 450200 | 513400 | 514390 | 514080 | **515480** |
| | Std. dev | 1575.60 | 1013.39 | 1205.04 | 700.42 | 1976.73 | 1723.29 | 1353.10 | 1726.89 |
| | Gap | 12.15 | 13.20 | 14.06 | 14.48 | 2.17 | 1.94 | 1.82 | 1.76 |
| IDKP10 | Best | 512140 | 505660 | 503250 | 500520 | 571780 | **572400** | 572250 | 572260 |
| | AVE | 509844.0 | 503661.7 | 498110.3 | 495816.3 | 568518.7 | 569094.3 | 568045.0 | **569364.7** |
| | Worst | 506800 | 501750 | 495910 | 493320 | 564150 | 564620 | 563900 | **566140** |
| | Std. dev | 1072.29 | 996.78 | 1548.61 | 1635.38 | 1711.13 | 1681.26 | 1724.57 | 1446.41 |
| | Gap | 12.28 | 13.35 | 14.30 | 14.70 | 2.19 | 2.09 | 2.27 | 2.04 |

Table 3: Comparison of 8 BPSO algorithms on SDKP1–SDKP10.

| | | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| SDKP1 | Best | 91046 | 90103 | 90067 | 89558 | 94300 | **94421** | 94414 | 94363 |
| | AVE | 90484.4 | 89446.6 | 88655.5 | 88449.9 | 94089.3 | 94112.7 | 94174.4 | **94181.6** |
| | Worst | 89964 | 88772 | 87835 | 87847 | 93687 | 93807 | **93920** | 93897 |
| | Std. dev | 240.44 | 312.35 | 513.73 | 422.86 | 152.07 | 128.84 | 103.15 | 113.74 |
| | Gap | 4.21 | 5.31 | 6.14 | 6.36 | 0.39 | 0.37 | 0.30 | 0.29 |
| SDKP2 | Best | 152080 | 149910 | 148440 | 148320 | 160470 | 160490 | 160430 | **160570** |
| | AVE | 150627.7 | 148857.0 | 147307.0 | 146598.3 | 160058.0 | 160033.3 | 160127.0 | **160136.0** |
| | Worst | 149590 | 148000 | 146410 | 145830 | 159730 | 159380 | 159480 | **159780** |
| | Std. dev | 591.17 | 505.77 | 535.13 | 595.93 | 190.92 | 237.35 | 214.82 | 182.33 |
| | Gap | 6.33 | 7.43 | 8.39 | 8.83 | 0.46 | 0.48 | 0.42 | 0.42 |
| SDKP3 | Best | 225340 | 222940 | 220310 | 218570 | **237580** | 237350 | 237430 | 237370 |
| | AVE | 223456.7 | 221130.0 | 218620.0 | 217727.3 | 236629.7 | 236821.0 | **236829.3** | 236778.3 |
| | Worst | 222120 | 219740 | 217530 | 217020 | 235770 | **236390** | 236110 | 235960 |
| | Std. dev | 617.52 | 777.22 | 567.24 | 435.05 | 359.95 | 265.44 | 352.00 | 314.48 |
| | Gap | 6.21 | 7.18 | 8.24 | 8.61 | 0.68 | 0.60 | 0.60 | 0.62 |
| SDKP4 | Best | 317610 | 313630 | 310850 | 310020 | 338360 | 338480 | **338520** | 338240 |
| | AVE | 315521.0 | 311837.3 | 308411.3 | 306979.7 | 337381.0 | 337525.0 | **337545.0** | 337433.0 |
| | Worst | 314020 | 310180 | 306940 | 305490 | 336270 | **336540** | 336440 | 336210 |
| | Std. dev | 938.08 | 789.99 | 1010.63 | 1074.09 | 501.50 | 524.94 | 540.16 | 495.98 |
| | Gap | 7.21 | 8.29 | 9.30 | 9.72 | 0.78 | 0.74 | 0.73 | 0.76 |
| SDKP5 | Best | 438720 | 431560 | 427340 | 425190 | 459900 | 459890 | **460190** | 459970 |
| | AVE | 435066.7 | 429523.7 | 424714.0 | 422803.3 | 458421.7 | 458479.3 | **458505.0** | 458292.0 |
| | Worst | 432570 | 427170 | 422560 | 421120 | 456570 | 456980 | **457430** | 457130 |
| | Std. dev | 1412.53 | 1057.97 | 1127.73 | 833.64 | 724.35 | 669.22 | 548.81 | 802.76 |
| | Gap | 6.04 | 7.24 | 8.28 | 8.69 | 1.00 | 0.98 | 0.98 | 1.02 |
| SDKP6 | Best | 428070 | 424110 | 417840 | 415960 | 463040 | 462310 | **463070** | 462650 |
| | AVE | 424916.3 | 420600.0 | 415690.3 | 413662.3 | 461229.3 | 461288.0 | **461370.0** | 461310.7 |
| | Worst | 423230 | 418080 | 414020 | 412230 | 459870 | 459390 | 459610 | **459880** |
| | Std. dev | 1121.73 | 1370.31 | 1020.89 | 994.05 | 759.15 | 708.60 | 789.20 | 680.64 |
| | Gap | 8.84 | 9.76 | 10.81 | 11.25 | 1.04 | 1.03 | 1.01 | 1.03 |
| SDKP7 | Best | 580940 | 574630 | 567460 | 562990 | 615640 | **615810** | 615100 | 614830 |
| | AVE | 578574.3 | 571778.7 | 564752.3 | 561783.7 | 613474.3 | **613542.7** | 613242.0 | 613182.7 |
| | Worst | 576180 | 569410 | 563080 | 560270 | 611790 | **611970** | 611180 | 610700 |
| | Std. dev | 1210.37 | 1118.88 | 1034.42 | 708.71 | 955.25 | 1000.74 | 953.90 | 984.83 |
| | Gap | 6.75 | 7.84 | 8.98 | 9.45 | 1.12 | 1.11 | 1.16 | 1.17 |

Table 3: Continued.

| | | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| SDKP8 | Best | 619660 | 613680 | 605780 | 602220 | **665540** | 665340 | 665070 | 665230 |
| | AVE | 616572.0 | 609950.0 | 603096.3 | 600201.3 | **663894.7** | 663703.3 | 663633.7 | 663681.7 |
| | Worst | 614660 | 607940 | 601290 | 598950 | **662710** | 661970 | 661310 | 661500 |
| | Std. dev | 1288.63 | 1507.53 | 1244.25 | 876.08 | 772.79 | 865.64 | 806.70 | 925.22 |
| | Gap | 8.07 | 9.06 | 10.08 | 10.51 | 1.01 | 1.04 | 1.05 | 1.05 |
| SDKP9 | Best | 679360 | 672620 | 664590 | 661320 | **733260** | 732790 | 732800 | 732910 |
| | AVE | 676278.0 | 668958.7 | 661978.0 | 659159.0 | **731433.3** | 731057.0 | 731107.0 | 730950.3 |
| | Worst | 671400 | 667010 | 659530 | 657170 | **730050** | 729290 | 728590 | 728990 |
| | Std. dev | 1620.02 | 1566.35 | 1313.95 | 1201.62 | 645.22 | 869.96 | 892.30 | 1024.07 |
| | Gap | 8.50 | 9.49 | 10.44 | 10.82 | 1.04 | 1.09 | 1.08 | 1.11 |
| SDKP10 | Best | 691200 | 681980 | 677280 | 672420 | 758000 | 757900 | 758090 | **758110** |
| | AVE | 687130.3 | 679407.7 | 672908.3 | 669316.7 | 756100.0 | 756291.0 | **756452.3** | 756337.3 |
| | Worst | 684090 | 676750 | 669860 | 667010 | **754400** | 754250 | 754150 | 753730 |
| | Std. dev | 1737.15 | 1527.54 | 1694.79 | 1452.26 | 837.73 | 993.51 | 1023.71 | 1038.86 |
| | Gap | 10.22 | 11.23 | 12.07 | 12.54 | 1.20 | 1.18 | 1.16 | 1.17 |

Table 4: Comparison of 8 BPSO algorithms on WDKP1–WDKP10.

| | | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| WDKP1 | Best | 80619 | 79846 | 79486 | 78556 | 83086 | 83083 | 83083 | **83090** |
| | AVE | 80013.7 | 79208.4 | 78425.3 | 78067.9 | 82977.3 | 83001.4 | 82995.0 | **83014.1** |
| | Worst | 79143 | 78770 | 77952 | 77505 | 82783 | **82876** | 82866 | 82649 |
| | Std. dev | 344.73 | 308.46 | 352.70 | 246.35 | 81.16 | 59.47 | 61.93 | 82.14 |
| | Gap | 3.71 | 4.68 | 5.62 | 6.05 | 0.15 | 0.12 | 0.12 | 0.10 |
| WDKP2 | Best | 129590 | 127850 | 126590 | 126010 | 138020 | 138040 | 138060 | **138110** |
| | AVE | 128602.7 | 126549.3 | 124836.0 | 124143.0 | 137769.7 | 137740.0 | 137814.3 | **137830.0** |
| | Worst | 127680 | 125580 | 123800 | 123260 | 137390 | 137420 | **137430** | 137270 |
| | Std. dev | 534.56 | 561.11 | 643.57 | 620.55 | 154.53 | 128.04 | 171.26 | 161.59 |
| | Gap | 6.95 | 8.44 | 9.68 | 10.18 | 0.32 | 0.34 | 0.29 | 0.28 |
| WDKP3 | Best | 243760 | 240740 | 239080 | 237110 | 256120 | 256320 | 256330 | **256360** |
| | AVE | 242457.7 | 239491.0 | 236892.0 | 235980.0 | 255743.0 | 255603.0 | **255831.3** | 255778.7 |
| | Worst | 241370 | 238570 | 235560 | 235110 | 254750 | 254910 | **254930** | 254850 |
| | Std. dev | 661.12 | 581.07 | 749.34 | 481.79 | 330.32 | 377.36 | 351.57 | 325.67 |
| | Gap | 5.52 | 6.67 | 7.69 | 8.04 | 0.34 | 0.39 | 0.31 | 0.33 |
| WDKP4 | Best | 296760 | 293210 | 289800 | 288090 | 315040 | **315120** | 315020 | 315040 |
| | AVE | 294886.7 | 291169.0 | 287984.7 | 286589.7 | 314392.0 | **314444.7** | 314413.3 | 314340.7 |
| | Worst | 293860 | 289950 | 286820 | 285510 | 313470 | **313900** | 313160 | 313360 |
| | Std. dev | 660.30 | 638.33 | 728.30 | 541.86 | 360.52 | 318.05 | 423.15 | 471.89 |
| | Gap | 6.58 | 7.76 | 8.77 | 9.21 | 0.40 | 0.38 | 0.39 | 0.42 |
| WDKP5 | Best | 402060 | 397880 | 393220 | 392120 | 427390 | 427280 | **427420** | 427390 |
| | AVE | 400608.7 | 395759.3 | 391217.7 | 389422.0 | 426337.3 | 426314.7 | **426490.7** | 426349.7 |
| | Worst | 398530 | 394270 | 389620 | 387990 | 425140 | 424980 | **425170** | 424950 |
| | Std. dev | 909.19 | 915.60 | 995.13 | 928.49 | 570.29 | 569.34 | 507.68 | 640.64 |
| | Gap | 6.51 | 7.64 | 8.70 | 9.12 | 0.50 | 0.51 | 0.47 | 0.50 |
| WDKP6 | Best | 432220 | 427900 | 421630 | 419290 | **464810** | 464440 | 464430 | 464040 |
| | AVE | 429596.3 | 424036.7 | 419191.3 | 417385.0 | 463214.7 | **463253.3** | 463201.7 | 463228.3 |
| | Worst | 428160 | 421740 | 417910 | 416010 | 461760 | 461920 | **462000** | 461980 |
| | Std. dev | 1087.14 | 1200.48 | 892.42 | 762.16 | 666.55 | 627.89 | 555.16 | 487.82 |
| | Gap | 7.82 | 9.01 | 10.05 | 10.44 | 0.61 | 0.60 | 0.61 | 0.61 |
| WDKP7 | Best | 506860 | 503130 | 495310 | 494200 | **545820** | 545570 | 545570 | 545270 |
| | AVE | 505569.3 | 499531.0 | 493412.7 | 491195.3 | **544327.0** | 544235.3 | 544057.0 | 544006.7 |
| | Worst | 503840 | 496860 | 491740 | 489660 | **543130** | 542560 | 542170 | 542140 |
| | Std. dev | 786.27 | 1319.85 | 847.95 | 1281.14 | 620.46 | 884.96 | 783.31 | 779.24 |
| | Gap | 7.69 | 8.79 | 9.91 | 10.31 | 0.61 | 0.63 | 0.66 | 0.67 |

Table 4: Continued.

| | | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| WDKP8 | Best | 521960 | 515930 | 509890 | 508840 | **574630** | 574580 | 573780 | 574290 |
| | AVE | 519367.7 | 513180.0 | 507473.7 | 505650.3 | 572645.7 | **572659.0** | 572513.7 | 572652.7 |
| | Worst | 515830 | 510670 | 505790 | 503570 | 571260 | 570240 | 570270 | **571520** |
| | Std. dev | 1441.06 | 1205.03 | 1069.10 | 1391.39 | 797.36 | 771.39 | 705.13 | 693.20 |
| | Gap | 9.98 | 11.05 | 12.04 | 12.36 | 0.75 | 0.75 | 0.77 | 0.75 |
| WDKP9 | Best | 584160 | 577510 | 571210 | 568630 | 646730 | **647010** | 646690 | 646910 |
| | AVE | 580843.3 | 574534.3 | 568415.0 | 566222.3 | 645331.7 | **645445.0** | 645101.0 | 645133.0 |
| | Worst | 576840 | 572340 | 566130 | 564130 | **643900** | 643790 | 642660 | 642980 |
| | Std. dev | 1659.97 | 1273.19 | 1136.32 | 1116.77 | 811.74 | 901.38 | 900.69 | 923.83 |
| | Gap | 10.73 | 11.70 | 12.64 | 12.98 | 0.82 | 0.80 | 0.85 | 0.85 |
| WDKP10 | Best | 592210 | 585730 | 580200 | 578010 | 675100 | **675790** | 675310 | 675460 |
| | AVE | 589632.0 | 582542.0 | 576737.3 | 574541.0 | 672859.7 | **673530.0** | 673296.3 | 673135.3 |
| | Worst | 586780 | 580160 | 574670 | 572440 | 669350 | **671700** | 671090 | 670990 |
| | Std. dev | 1281.47 | 1245.80 | 1511.33 | 1320.04 | 1249.33 | 1078.37 | 959.27 | 1171.04 |
| | Gap | 13.15 | 14.20 | 15.05 | 15.38 | 0.90 | 0.80 | 0.83 | 0.86 |

Table 5: Comparison of 8 BPSO algorithms on UDKP1–UDKP10.

| | Algorithm | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| UDKP1 | Best | 79070 | 77244 | 75113 | 74482 | 85728 | **85740** | 85728 | **85740** |
| | AVE | 77771.0 | 75977.2 | 74212.6 | 73635.3 | 85526.1 | 85555.0 | 85532.2 | **85590.0** |
| | Worst | 76454 | 74811 | 72928 | 72824 | 85193 | 85282 | 85026 | **85347** |
| | Std. dev | 618.95 | 667.48 | 534.68 | 487.81 | 135.12 | 130.18 | 184.05 | **102.30** |
| | Gap | 9.29 | 11.39 | 13.44 | 14.12 | 0.25 | 0.22 | 0.24 | 0.17 |
| UDKP2 | Best | 145300 | 143840 | 138350 | 136930 | 163560 | 163500 | **163630** | 163560 |
| | AVE | 143420.3 | 139566.3 | 136063.0 | 134346.3 | 162678.3 | 162878.7 | **163002.0** | 163061.7 |
| | Worst | 141530 | 137850 | 134630 | 132520 | 161520 | 161760 | **162360** | **162260** |
| | Std. dev | 1032.82 | 1243.82 | 741.11 | 1018.84 | 466.99 | 399.78 | 304.90 | 314.45 |
| | Gap | 12.41 | 14.77 | 16.91 | 17.95 | 0.65 | 0.53 | 0.45 | 0.42 |
| UDKP3 | Best | 236780 | 231160 | 227280 | 224910 | 268270 | 267810 | **268450** | 268430 |
| | AVE | 233404.0 | 228742.0 | 224682.7 | 222441.0 | 266421.0 | 266729.3 | 266814.3 | **267008.7** |
| | Worst | 231530 | 226860 | 222570 | 220600 | 264740 | 265150 | 264980 | **265560** |
| | Std. dev | 1247.53 | 1053.30 | 1101.78 | 1077.08 | 864.86 | 715.80 | 805.32 | 681.78 |
| | Gap | 13.36 | 15.09 | 16.60 | 17.43 | 1.10 | 0.99 | 0.96 | 0.89 |
| UDKP4 | Best | 301870 | 295140 | 290750 | 288990 | 345340 | 345110 | 345980 | **346120** |
| | AVE | 299048.3 | 292933.7 | 287865.0 | 285816.7 | 343429.7 | 343510.3 | **343958.3** | 343789.0 |
| | Worst | 296850 | 290590 | 285580 | 284460 | 341160 | 341330 | **341970** | 341800 |
| | Std. dev | 1338.88 | 1104.72 | 1178.90 | 1126.14 | 999.47 | 848.45 | 1036.98 | 924.93 |
| | Gap | 13.97 | 15.73 | 17.18 | 17.77 | 1.20 | 1.18 | 1.05 | 1.10 |
| UDKP5 | Best | 377980 | 370780 | 363930 | 362700 | 436440 | 437610 | 437610 | **437870** |
| | AVE | 374958.3 | 367368.0 | 361415.3 | 358738.3 | 434067.0 | 434640.7 | **435638.0** | 435053.0 |
| | Worst | 372200 | 363700 | 358340 | 354560 | 430540 | 431110 | **432930** | 431500 |
| | Std. dev | 1687.23 | 1465.29 | 1430.94 | 1788.70 | 1415.51 | 1410.72 | 1362.97 | 1572.94 |
| | Gap | 15.29 | 17.01 | 18.35 | 18.96 | 1.94 | 1.81 | 1.58 | 1.71 |
| UDKP6 | Best | 460830 | 452640 | 441270 | 439260 | 528730 | 529110 | 528940 | **530600** |
| | AVE | 455015.3 | 446125.3 | 438852.3 | 435647.0 | 525352.3 | 525774.0 | 525666.0 | **527004.7** |
| | Worst | 451680 | 443210 | 435620 | 432550 | 521500 | 522460 | 519710 | **523880** |
| | Std. dev | 2176.80 | 1998.82 | 1368.33 | 1766.47 | 1682.21 | 1691.98 | 2340.59 | 1877.81 |
| | Gap | 15.20 | 16.86 | 18.21 | 18.81 | 2.09 | 2.01 | 2.03 | 1.78 |
| UDKP7 | Best | 545040 | 537910 | 525410 | 521590 | 624860 | 624300 | 626480 | **627350** |
| | AVE | 542066.3 | 531413.7 | 522129.3 | 518659.0 | 620906.0 | 620467.7 | **622543.3** | 621875.3 |
| | Worst | 539210 | 527820 | 518860 | 515290 | 616570 | 615210 | **618950** | 617290 |
| | Std. dev | 1425.63 | 2665.99 | 1579.61 | 1717.16 | 1898.05 | 2135.62 | 2122.03 | 2120.14 |
| | Gap | 14.75 | 16.43 | 17.89 | 18.43 | 2.35 | 2.42 | 2.09 | 2.20 |

Table 5: Continued.

| | Algorithm | BPSO1 | BPSO2 | BPSO3 | BPSO4 | BPSO5 | BPSO6 | BPSO7 | BPSO8 |
|---|---|---|---|---|---|---|---|---|---|
| UDKP8 | Best | 551650 | 540430 | 532270 | 533240 | 638060 | 638320 | **641500** | 637900 |
| | AVE | 546316.3 | 536630.3 | 528103.7 | 524527.7 | 633762.3 | 635553.7 | **635667.3** | 634388.3 |
| | Worst | 543810 | 532280 | 524740 | 521420 | 630280 | **630450** | 630150 | 629600 |
| | Std. dev | 1938.41 | 1996.29 | 1807.58 | 2466.31 | 2124.68 | 1784.87 | 2357.17 | 2121.45 |
| | Gap | 15.98 | 17.47 | 18.78 | 19.33 | 2.53 | 2.25 | 2.24 | 2.43 |
| UDKP9 | Best | 594880 | 582140 | 570220 | 565930 | 702810 | **705110** | 704700 | 704350 |
| | AVE | 588958.7 | 577306.7 | 566914.3 | 561907.7 | 699517.7 | 699452.3 | **700655.0** | 699156.3 |
| | Worst | 584970 | 574490 | 561620 | 558820 | 692160 | 693090 | **694810** | 693320 |
| | Std. dev | 2264.91 | 1811.95 | 1962.56 | 1931.90 | 2335.85 | 2628.18 | **2570.03** | 3132.54 |
| | Gap | 18.03 | 19.65 | 21.10 | 21.80 | 2.65 | 2.66 | 2.49 | 2.70 |
| UDKP10 | Best | 633620 | 619750 | 604900 | 598780 | 763810 | 760920 | 762640 | **767550** |
| | AVE | 627361.3 | 612962.3 | 598703.3 | 593618.7 | 756328.3 | 757261.7 | **757706.3** | 756768.3 |
| | Worst | 620570 | 608500 | 594060 | 589560 | 751310 | **752420** | 749190 | 749330 |
| | Std. dev | 3489.47 | 2565.51 | 2702.42 | 2415.98 | 3078.53 | 2348.74 | 3148.56 | 3865.48 |
| | Gap | 19.51 | 21.36 | 23.19 | 23.84 | 2.97 | 2.85 | 2.79 | 2.91 |

Table 6: Average ranks of 8 BPSO algorithms on 40 instances.

| Index | Algorithm | Average best rank | Average mean rank | Average worst rank |
|---|---|---|---|---|
| 1 | BPSO1 | 5.0 | 5.1 | 4.8 |
| 2 | BPSO2 | 6.1 | 5.9 | 5.3 |
| 3 | BPSO3 | 6.9 | 6.7 | 5.8 |
| 4 | BPSO4 | 8.0 | 7.2 | 6.3 |
| 5 | BPSO5 | 2.7 | 3.1 | 3.4 |
| 6 | BPSO6 | 2.5 | 2.6 | 3.6 |
| 7 | BPSO7 | 2.5 | 3.4 | 3.4 |
| 8 | BPSO8 | **2.2** | **2.0** | **3.5** |

Table 7: Comparison of BPSO8, SecEGA, and FirEGA on IDKP1–IDKP10.

| Index | Instance | OPT | Algorithm | Best | Average | Worst | Std. dev | Gap |
|---|---|---|---|---|---|---|---|---|
| 1 | IDKP1 | 70106 | FirEGA | **70106** | **70099** | **70090** | 7 | 0.01 |
| | | | SecEGA | 68663 | 68000 | 67369 | 328 | 3.00 |
| | | | BPSO8 | 69950 | 69761 | 69521 | 109 | 0.49 |
| 2 | IDKP2 | 118268 | FirEGA | **118169** | **117869** | **117625** | 102.59 | 0.34 |
| | | | SecEGA | 114434 | 113385 | 112307 | 7446.67 | 4.13 |
| | | | BPSO8 | 118000 | 117418.0 | 117040 | 228.83 | 0.72 |
| 3 | IDKP3 | 234804 | FirEGA | 234497 | 233997 | 233666 | 175.42 | 0.34 |
| | | | SecEGA | 220096 | 217982 | 216313 | 835.83 | 7.16 |
| | | | BPSO8 | **234540** | **234253.7** | **233800** | 192.52 | 0.23 |
| 4 | IDKP4 | 282591 | FirEGA | **282148** | 280695 | 278881 | 827.63 | 0.67 |
| | | | SecEGA | 263238 | 260425 | 258922 | 933.4 | 7.84 |
| | | | BPSO8 | 281560 | **280953.3** | **280220** | 328.28 | 0.58 |
| 5 | IDKP5 | 335584 | FirEGA | **335004** | **333484** | 329621 | 1173.9 | 0.63 |
| | | | SecEGA | 309573 | 306878 | 304881 | 907.19 | 8.55 |
| | | | BPSO8 | 334330 | 333380.7 | **332480** | 479.64 | 0.66 |
| 6 | IDKP6 | 452463 | FirEGA | **451680** | 449863 | 446704 | 1161.52 | 0.58 |
| | | | SecEGA | 414090 | 411367 | 408788 | 1099.31 | 9.08 |
| | | | BPSO8 | 451390 | **450455.0** | **448980** | 563.12 | 0.44 |
| 7 | IDKP7 | 489149 | FirEGA | **488009** | **485592** | 476385 | 2294.28 | 0.73 |
| | | | SecEGA | 451528 | 444316 | 442133 | 1280.31 | 9.17 |
| | | | BPSO8 | 485920 | 484205.7 | **482400** | 879.02 | 1.01 |
| 8 | IDKP8 | 533841 | FirEGA | **533035** | **529984** | 514196 | 2308.11 | 0.72 |
| | | | SecEGA | 490494 | 481831 | 478035 | 2215.66 | 9.74 |
| | | | BPSO8 | 528450 | 526356.7 | **524150** | 1268.72 | 1.40 |

Table 7: Continued.

| Index | Instance | OPT | Algorithm | Best | Average | Worst | Std. dev | Gap |
|---|---|---|---|---|---|---|---|---|
| 9 | IDKP9 | 528144 | FirEGA | **526410** | **523982** | 511651 | 2216.13 | 0.79 |
| | | | SecEGA | 489661 | 477001 | 471848 | 3656.22 | 9.68 |
| | | | BPSO8 | 522140 | 518834.7 | **515480** | 1726.89 | 1.76 |
| 10 | IDKP10 | 581244 | FirEGA | **578903** | **576772** | **568903** | 1905.18 | 0.77 |
| | | | SecEGA | 535541 | 521604 | 516445 | 4265.07 | 10.26 |
| | | | BPSO8 | 572260 | 569364.7 | 566140 | 1446.41 | 2.04 |

Table 8: Comparison of BPSO8, SecEGA, and FirEGA on SDKP1–SDKP10

| Index | Instance | OPT | Algorithm | Best | Average | Worst | Std. dev | Gap |
|---|---|---|---|---|---|---|---|---|
| 1 | SDKP1 | 94459 | FirEGA | 93235 | 93171 | 93070 | 42.15 | 1.36 |
| | | | SecEGA | 89769 | 88832 | 87463 | 594.91 | 5.96 |
| | | | BPSO8 | **94363** | **94181.6** | **93897** | 113.74 | 0.29 |
| 2 | SDKP2 | 160805 | FirEGA | 159159 | 159004 | 158859 | 61.54 | 1.12 |
| | | | SecEGA | 153821 | 152059 | 150753 | 489.39 | 5.44 |
| | | | BPSO8 | **160570** | **160136.0** | **159780** | 182.33 | 0.42 |
| 3 | SDKP3 | 238248 | FirEGA | 235454 | 235241 | 235043 | 79.86 | 1.26 |
| | | | SecEGA | 224997 | 223580 | 221918 | 543.38 | 6.16 |
| | | | BPSO8 | **237370** | **236778.3** | **235960** | 314.48 | 0.62 |
| 4 | SDKP4 | 340027 | FirEGA | 336353 | 335963 | 335709 | 122.41 | 1.20 |
| | | | SecEGA | 318510 | 315513 | 313747 | 851.14 | 7.21 |
| | | | BPSO8 | **338240** | **337433.0** | **336210** | 495.98 | 0.76 |
| 5 | SDKP5 | 463033 | FirEGA | 452900 | 447587 | 444255 | 1974.99 | 3.34 |
| | | | SecEGA | 420238 | 416964 | 413933 | 1291.65 | 9.95 |
| | | | BPSO8 | **459970** | **458292.0** | **457130** | 802.76 | 1.02 |
| 6 | SDKP6 | 466097 | FirEGA | 459254 | 458893 | 458584 | 162.94 | 1.55 |
| | | | SecEGA | 430738 | 427304 | 425504 | 1031.12 | 8.32 |
| | | | BPSO8 | **462650** | **461310.7** | **459880** | 680.64 | 1.03 |
| 7 | SDKP7 | 620446 | FirEGA | 599361 | 592279 | 579673 | 3949.03 | 4.54 |
| | | | SecEGA | 561224 | 556083 | 552007 | 1926.26 | 10.37 |
| | | | BPSO8 | **614830** | **613182.7** | **610700** | 984.83 | 1.17 |
| 8 | SDKP8 | 670697 | FirEGA | 661276 | 660104 | 659367 | 426.06 | 1.58 |
| | | | SecEGA | 611644 | 606263 | 603774 | 1446.94 | 9.61 |
| | | | BPSO8 | **665230** | **663681.7** | **661500** | 925.22 | 1.05 |
| 9 | SDKP9 | 739121 | FirEGA | 729135 | 727544 | 727064 | 343.67 | 1.57 |
| | | | SecEGA | 674885 | 667900 | 664580 | 1614.04 | 9.64 |
| | | | BPSO8 | **732910** | **730950.3** | **728990** | 1024.07 | 1.11 |
| 10 | SDKP10 | 765317 | FirEGA | 756205 | 753394 | 750757 | 985.46 | 1.56 |
| | | | SecEGA | 708935 | 695557 | 691994 | 2956.08 | 9.12 |
| | | | BPSO8 | **758110** | **756337.3** | **753730** | 1038.86 | 1.17 |

Table 9: Comparison of BPSO8, SecEGA, and FirEGA on WDKP1–WDKP10.

| Index | Instance | OPT | Algorithm | Best | Average | Worst | Std. dev | Gap |
|---|---|---|---|---|---|---|---|---|
| 1 | WDKP1 | 83098 | FirEGA | 82803 | 82693 | 82592 | 52.04 | 0.49 |
| | | | SecEGA | 80014 | 79022 | 78096 | 473.67 | 4.91 |
| | | | BPSO8 | **83090** | **83014.1** | **82649** | 82.14 | 0.10 |
| 2 | WDKP2 | 138215 | FirEGA | 137704 | 137584 | **137356** | 63.23 | 0.46 |
| | | | SecEGA | 133315 | 132276 | 131337 | 415.62 | 4.30 |
| | | | BPSO8 | **138110** | **137830.0** | 137270 | 161.59 | 0.28 |
| 3 | WDKP3 | 256616 | FirEGA | 254120 | 253657 | 253307 | 173.01 | 1.15 |
| | | | SecEGA | 238331 | 235721 | 234025 | 873.58 | 8.14 |
| | | | BPSO8 | **256360** | **255778.7** | **254850** | 325.67 | 0.33 |

Table 9: Continued.

| Index | Instance | OPT | Algorithm | Best | Average | Worst | Std. dev | Gap |
|---|---|---|---|---|---|---|---|---|
| | | | FirEGA | 313966 | 312849 | 311998 | 484.76 | 0.89 |
| 4 | WDKP4 | 315657 | SecEGA | 293640 | 290851 | 288764 | 950.06 | 7.86 |
| | | | BPSO8 | **315040** | **314340.7** | **313360** | 471.89 | 0.42 |
| | | | FirEGA | 426311 | 424548 | 423058 | 798.53 | 0.92 |
| 5 | WDKP5 | 428490 | SecEGA | 393617 | 390014 | 387992 | 1059.83 | 8.98 |
| | | | BPSO8 | **427390** | **426349.7** | **424950** | 640.64 | 0.50 |
| | | | FirEGA | 463185 | 461672 | 457718 | 1107.57 | 0.94 |
| 6 | WDKP6 | 466050 | SecEGA | 429208 | 425112 | 423269 | 1058.37 | 8.78 |
| | | | BPSO8 | **464040** | **463228.3** | **461980** | 487.82 | 0.61 |
| | | | FirEGA | 544019 | 541949 | 538126 | 1224.68 | 1.05 |
| 7 | WDKP7 | 547683 | SecEGA | 501557 | 496134 | 493845 | 1230.94 | 9.41 |
| | | | BPSO8 | **545270** | **544006.7** | **542140** | 779.24 | 0.67 |
| | | | FirEGA | 573427 | 571559 | 563253 | 1495.36 | 0.94 |
| 8 | WDKP8 | 576959 | SecEGA | 530971 | 523203 | 520350 | 2157.09 | 9.32 |
| | | | BPSO8 | **574290** | **572652.7** | **571520** | 693.20 | 0.75 |
| | | | FirEGA | 647477 | 644820 | 630086 | 2056.06 | 0.90 |
| 9 | WDKP9 | 650660 | SecEGA | 598343 | 586770 | 583854 | 2315.5 | 9.82 |
| | | | BPSO8 | 646910 | 645133.0 | 642980 | 923.83 | 0.85 |
| | | | FirEGA | 675452 | 673008 | 668239 | 1441.96 | 0.88 |
| 10 | WDKP10 | 678947 | SecEGA | 620230 | 606215 | 609964 | 3090.86 | 10.72 |
| | | | BPSO8 | **675460** | **673135.3** | **670990** | 1171.04 | 0.86 |

Table 10: Comparison of BPSO8, SecEGA, and FirEGA on UDKP1–UDKP10.

| Index | Instance | OPT | Algorithm | Best | Average | Worst | Std. dev | Gap |
|---|---|---|---|---|---|---|---|---|
| | | | FirEGA | 80593 | 79103 | 77935 | 690.01 | 7.74 |
| 1 | UDKP1 | 85740 | SecEGA | 78287 | 76807 | 75156 | 798.95 | 10.42 |
| | | | BPSO8 | **85740** | **85590.0** | **85347** | 102.30 | 0.17 |
| | | | FirEGA | 155039 | 151662 | 149875 | 1044.95 | 7.38 |
| 2 | UDKP2 | 163744 | SecEGA | 148043 | 145548 | 143833 | 883.43 | 11.11 |
| | | | BPSO8 | **163560** | **163061.7** | **162260** | 314.45 | 0.42 |
| | | | FirEGA | 246698 | 240886 | 237980 | 1491.97 | 10.58 |
| 3 | UDKP3 | 269393 | SecEGA | 228823 | 225492 | 222486 | 1353.58 | 16.30 |
| | | | BPSO8 | **268430** | **267008.7** | **265560** | 681.78 | 0.89 |
| | | | FirEGA | 321605 | 317319 | 314486 | 1426.85 | 8.71 |
| 4 | UDKP4 | 347599 | SecEGA | 305796 | 299978 | 297606 | 1435.46 | 13.70 |
| | | | BPSO8 | **346120** | **343789.0** | **341800** | 924.93 | 1.10 |
| | | | FirEGA | 405409 | 399620 | 395367 | 1692.23 | 9.72 |
| 5 | UDKP5 | 442644 | SecEGA | 376147 | 370808 | 367574 | 1611.71 | 16.23 |
| | | | BPSO8 | **437870** | **435053.0** | **431500** | 1572.94 | 1.71 |
| | | | FirEGA | 486556 | 478726 | 474015 | 2233.61 | 10.78 |
| 6 | UDKP6 | 536578 | SecEGA | 447438 | 442499 | 438809 | 1765.28 | 17.53 |
| | | | BPSO8 | **530600** | **527004.7** | **523880** | 1877.81 | 1.78 |
| | | | FirEGA | 568119 | 560948 | 556938 | 2441.8 | 11.78 |
| 7 | UDKP7 | 635860 | SecEGA | 529753 | 521401 | 518407 | 1813.04 | 18.00 |
| | | | BPSO8 | **627350** | **621875.3** | **617290** | 2120.14 | 2.20 |
| | | | FirEGA | 590137 | 585286 | 580684 | 2078.87 | 9.99 |
| 8 | UDKP8 | 650206 | SecEGA | 550645 | 546678 | 543836 | 1449.36 | 15.92 |
| | | | BPSO8 | **637900** | **634388.3** | **629600** | 2121.45 | 2.43 |
| | | | FirEGA | 655172 | 649636 | 645012 | 2023.64 | 9.59 |
| 9 | UDKP9 | 718532 | SecEGA | 613581 | 602215 | 605835 | 2003.75 | 16.19 |
| | | | BPSO8 | **704350** | **699156.3** | **693320** | 3132.54 | 2.70 |
| | | | FirEGA | 712270 | 706575 | 701545 | 2013.43 | 9.35 |
| 10 | UDKP10 | 779460 | SecEGA | 665459 | 658908 | 655645 | 1723.8 | 15.47 |
| | | | BPSO8 | **767550** | **756768.3** | **749330** | 3865.48 | 2.91 |

Table 11: Average ranks of BPSO8, SecEGA, and FirEGA on 40 instances.

| Index | Algorithm | Average best rank | Average mean rank | Average worst rank |
|---|---|---|---|---|
| 1 | FirEGA | 1.8 | 1.3 | 1.4 |
| 2 | SecGA | 2.9 | 2.1 | 2.1 |
| 3 | BPSO8 | **1.3** | **0.9** | **0.9** |

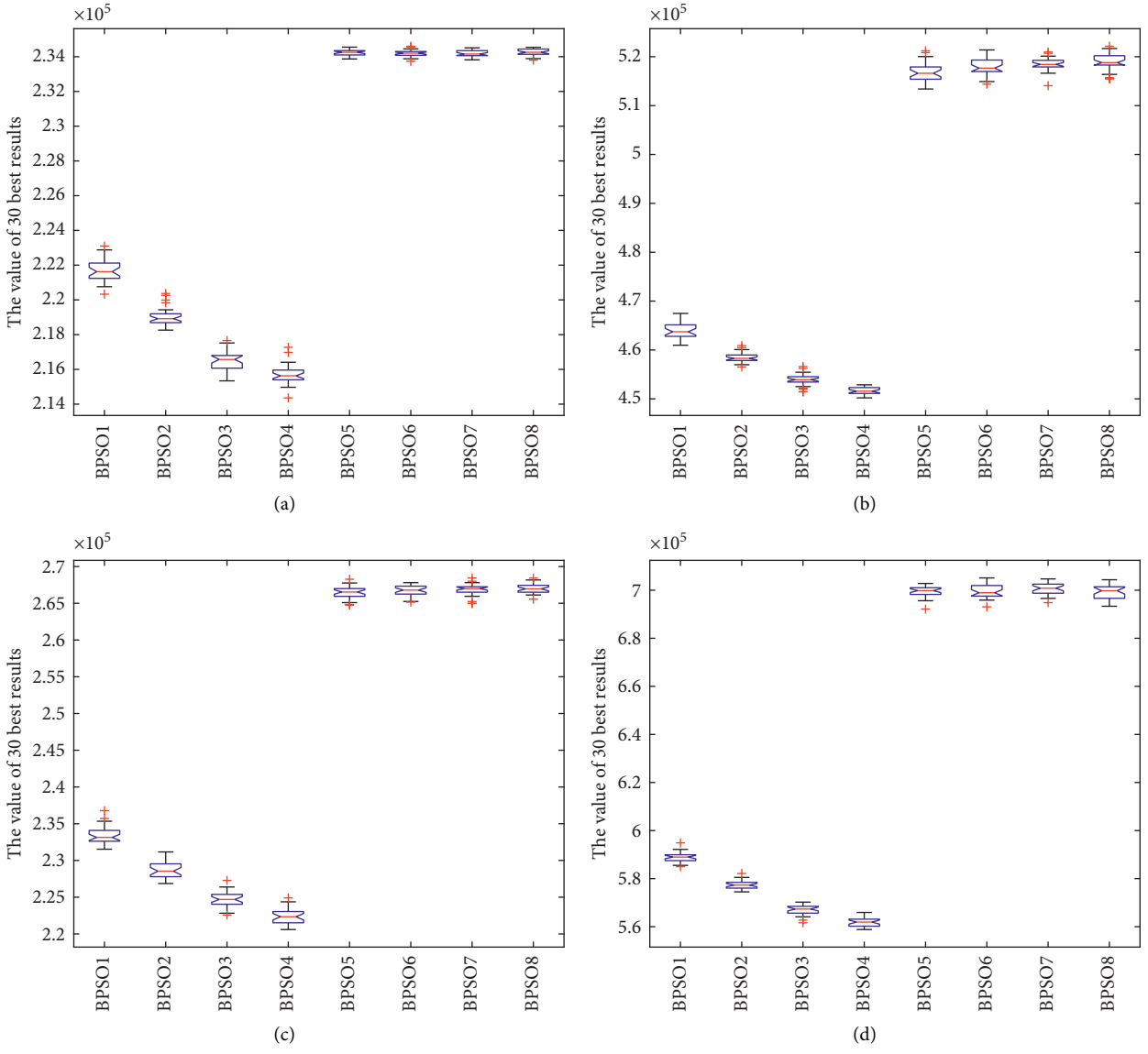


Figure 1: Continued.

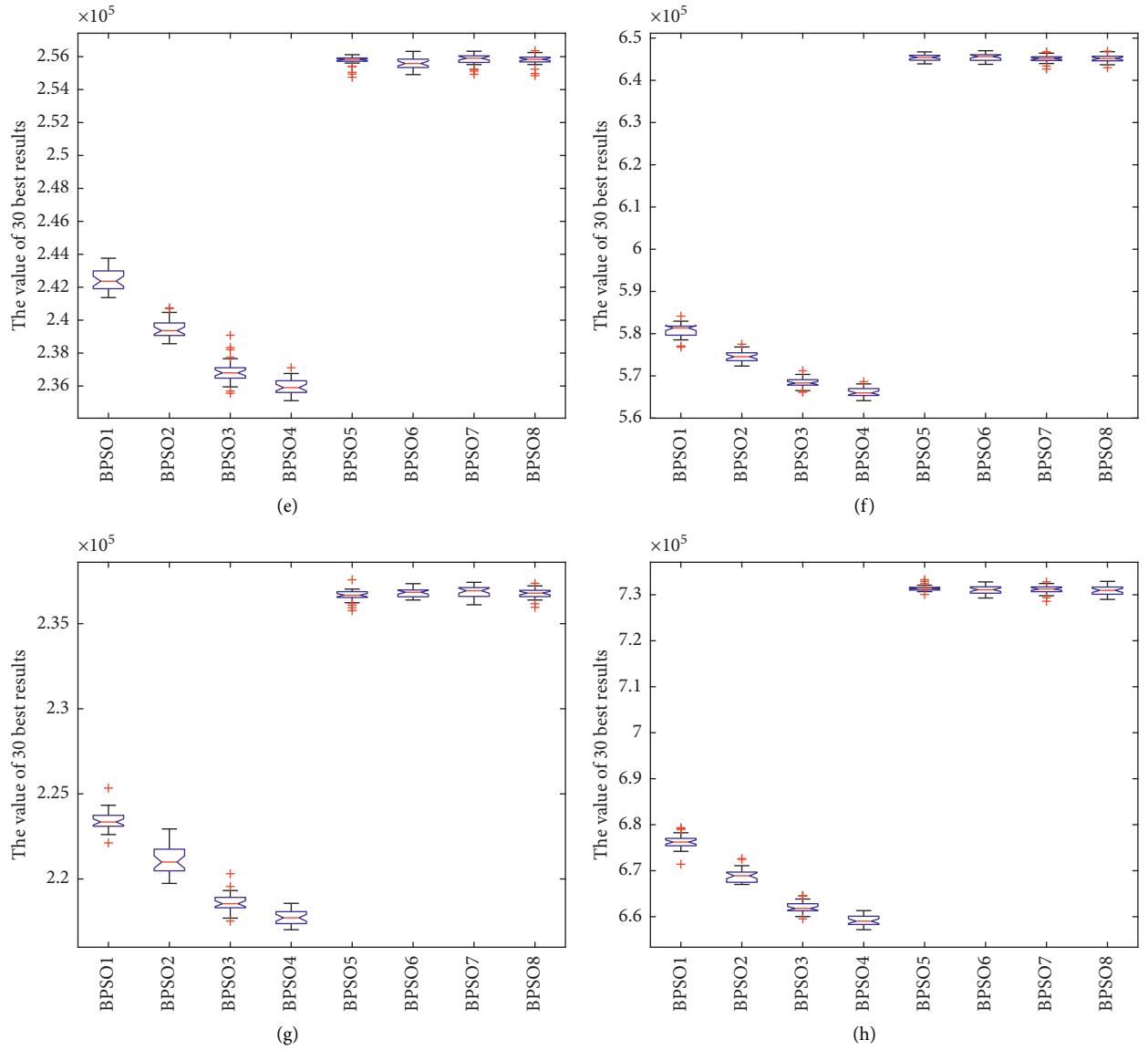


Figure 1: Box plot of 8 instances: (a) IDKP3, (b) IDKP9, (c) UDKP3, (d) UDKP9, (e) WDKP3, (f) WDKP9, (g) SDKP3, and (h) SDKP9.

to the dimension of the DKP01, and the stopping criterion is satisfied when the maximum number of iterations is reached. For all algorithms, the numbers of objective function evaluations are similar.

Tables 2–5 summarize the comparison among 8 BPSO algorithms based on the five different performance factors, that is, the best results (BEST), the average results (AVE), the worst results (Worst), the standard deviation (Std. dev), and the gap between the AVE and OPT, where OPT is the optimal value of the instance. The results are averaged over 30 independent runs, and the best results are highlighted in bold font. The formula of computing the gap is as follows:

$$\text{Gap} = \frac{|\text{OPT} - \text{AVE}|}{\text{OPT}} \times 100\%. \tag{11}$$

The results show that BPSO7 and BPSO8 have better performance compared to the other six algorithms. Table 6 summarizes the average ranks of eight BPSO algorithms on 40 instances. The results showed that BPSO8 achieved the average best rank in all three factors, that is, average best rank (rank based on Best), average mean rank (rank based on AVE), and average worst rank (rank based on Worst). Tables 7–10 summarize the comparison among the FirEGA, SecEGA, and BPSO8 based on the five different performance criteria on 30 independent runs: BEST, AVE, Worst, Std. dev, and Gap. BPSO8 is better than FirEGA and SecEGA in Best, AVE, and Worst for the instances of SDKP, UDKP, and WDKP except for instances of IDKP. The results show that BPSO8 has better performance than FirEGA and SecEGA algorithms. Table 11 summarizes the average ranks of eight BPSO8, FirEGA, and SecEGA algorithms on 40 instances. The results showed that BPSO8 achieved the average best rank in all three factors, that is, average best rank (rank based on Best), the average mean rank (rank based on AVE), and average worst rank (rank based on Worst).

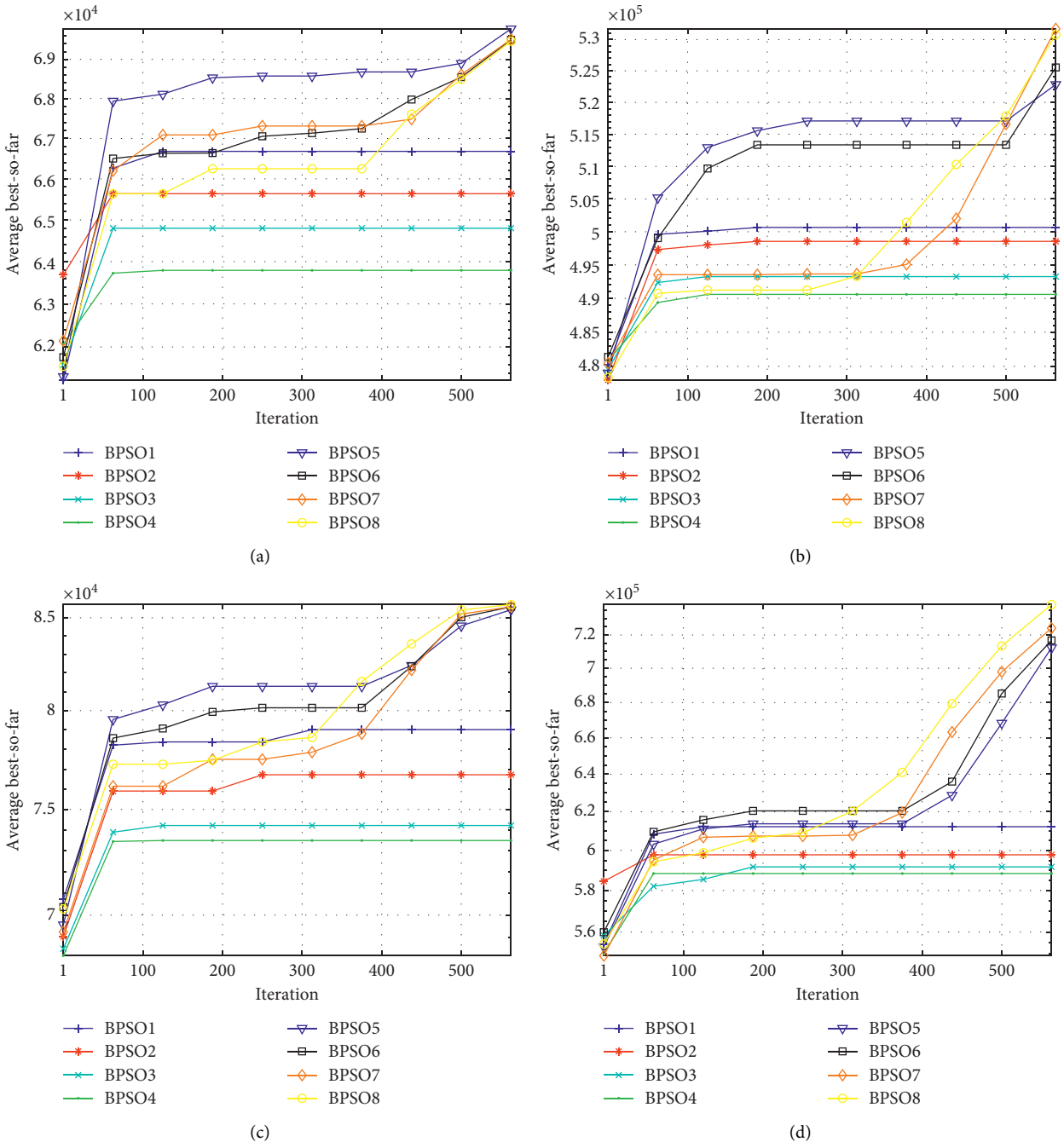


Figure 2: Continued.

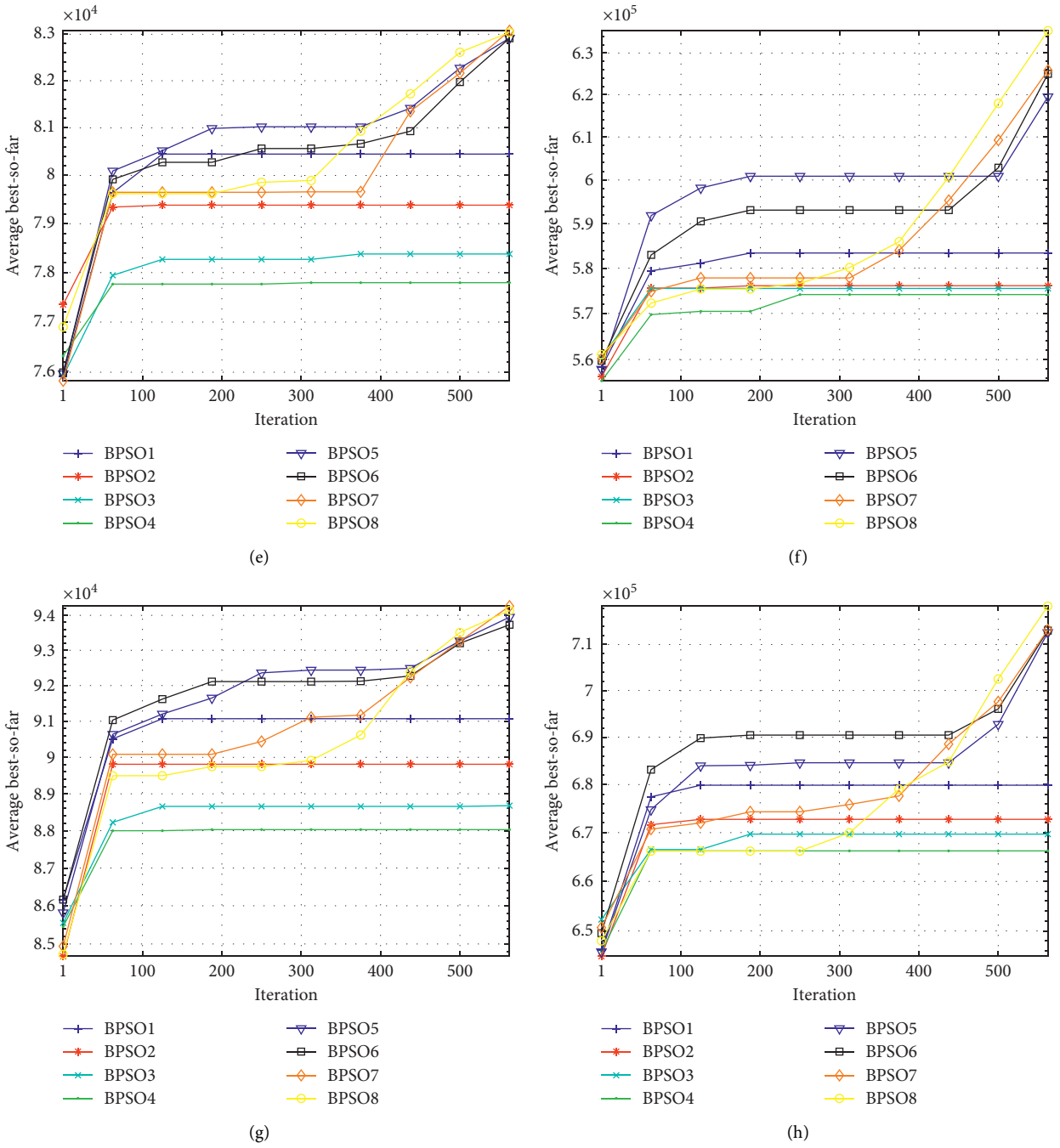


Figure 2: Convergence curve of (a) IDKP1, (b) IDKP10, (c) UDKP1, (d) UDKP10, (e) WDKP1, (f) WDKP10, (g) SDKP1, and (h) SDKP10.

For the stability, the Std. dev and Gap value from Tables 2–10 demonstrate the stability of the proposed algorithms. Figure 1 demonstrates the box plot of eight instances. The results showed that the group of algorithms BPSO5–BPSO8 is better than group of algorithms BPSO1–BPSO4. Figure 2 demonstrates the convergence curves of eight instances. The results showed that the group of algorithms BPSO5–BPSO8 has faster convergence than the group of algorithms BPSO1–BPSO4.

Therefore, the performance of BPSO8 is excellent compared to that of BPSOs for the DKP01 problem. From the above comparison, it is not difficult to see that, for the DKP01 problem, the BPSOS has the best performance, followed by BPSO8, and they are far better than FirEGA and SecEGA. This shows that the designed method of binary swarm optimization with a new binary encoding scheme is not only feasible but also effective.

## 5. Conclusion

In this paper, eight new algorithms have been proposed based on the binary particle swarm optimization with a new

repair operator to solve discounted 0-1 knapsack problem efficiently. An effective binary encoding scheme is proposed to present the solution to the problem. The new encoding scheme has two advantages, that is, helping reduce the computing effort when using shorter binary vector and also automatically satisfy the constraint that chose the most one item in each group. The simulation results on forty DKP01 instances showed that the proposed algorithms are better than the two algorithms based on genetic algorithm.

In the future, I will study the effect of transfer function combined with PSO algorithm for other optimization problems. Many other optimization algorithms are also considered to solve DKP01.

## 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.

## Additional Points

(i) Particle swarm optimization algorithms with difference binary transfer functions and new solution presentation are proposed to solve the discounted 0-1 knapsack problem. (ii) A new encoding scheme has shorter binary vector and also automatically satisfy the constraint that chose the most one item in each group. (iii) A new repair operator is developed to handle isolation solution while improving its quality. (iv) Experiment results in 40 instances of discounted 0-1 knapsack problem showed that the proposed approaches are efficient.

## Conflicts of Interest

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

## Acknowledgments

The author acknowledges 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, pp. 1872–1888, 2020.

[9] A. R. Jordehi, "Binary particle swarm optimisation with quadratic transfer function: a new binary optimisation algorithm for optimal scheduling of appliances in smart homes," *Applied Soft Computing*, vol. 78, pp. 465–480, 2019.

[10] S.-w. Fei, "The hybrid method of VMD-PSR-SVD and improved binary PSO-KNN for fault diagnosis of bearing," *Shock and Vibration*, vol. 2019, Article ID 4954920, 2019.

[11] A. N. Hussain and A. A. Ismail, "Operation cost reduction in unit commitment problem using improved quantum binary PSO algorithm," *International Journal of Electrical & Computer Engineering*, vol. 10, pp. 1149–1155, 2020.

[12] Z. Beheshti, "A time-varying mirrored S-shaped transfer function for binary particle swarm optimization," *Information Sciences*, vol. 512, pp. 1503–1542, 2020.

[13] A. Kaushik, M. Goswami, M. Manuja, S. Indu, and D. Gupta, "A binary PSO approach for improving the performance of wireless sensor networks," *Wireless Personal Communications*, vol. 113, pp. 263–297, 2020.

[14] L. F. F. Ledezma and G. G. Alcaraz, "Hybrid binary PSO for transmission expansion planning considering $n-1$ security criterion," *IEEE Latin America Transactions*, vol. 18, no. 3, pp. 545–553, 2020.

[15] Y. Zhang, D.-W. Gong, and Z. Ding, "A bare-bones multi-objective particle swarm optimization algorithm for environmental/economic dispatch," *Information Sciences*, vol. 192, pp. 213–227, 2012.

[16] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective particle swarm optimization for feature selection with fuzzy cost," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 874–888, 2020.

[17] Y. Zhang, D.-w. Gong, and J. Cheng, "Multi-objective particle swarm optimization approach for cost-based feature selection in classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 64–75, 2015.

[18] X.-F. Song, Y. Zhang, Y.-N. Guo, X.-Y. Sun, and Y.-L. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 882–895, 2020.

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

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

[21] Z. Li and N. Li, "A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem," in *Proceedings of the 21st Annual international Conference on Chinese Control and Decision Conference, Ser. CCDC'09*, pp. 3090–3095, Piscataway, NJ, USA, 2009.

[22] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for Knapsack problems," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11042–11061, 2012.
[23] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.
[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 and Applications*, vol. 30, 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.