ICIC Express Letters
Volume **15**, Number **3**, March **2021**
 ISSN 1881-803X
pp. **257–264**

A BINARY SOCIAL SPIDER ALGORITHM FOR DISCOUNTED {0-1} KNAPSACK PROBLEM

Viet Ngoc Tran[1] and Tung Khac Truong[2,*]

[1]Department of Information Technology
Van Lang University
80/68 Duong Quang Ham Street, Ward 5, Go Vap District, HCM City 71411, Vietnam
tranngocviet@vanlanguni.edu.vn

[2]Department of Information Technology
Industrial University of Ho Chi Minh City
12 Nguyen Van Bao Street, Ward 4, Go Vap District, HCM City 71408, Vietnam
*Corresponding author: tungtk@iuh.edu.vn

Received August 2020; accepted October 2020

Abstract. *This paper proposed a new binary social spider algorithm with repair operator solving discounted {0-1} knapsack problem (DKP01). The solution of DKP01 is presented by a binary vector. Social spider algorithm is a simple and powerful optimization algorithm. A new function is used to convert real vector to binary vector to design binary social spider algorithm. We conducted extensive experiments on two types of 20 instances using our proposed approach. The experiments proved that the new method is efficient for solving DKP01.*
Keywords: Discounted {0-1} knapsack, SSA algorithm, Optimization algorithm, Artificial intelligence, Heuristic

1. **Introduction.** A new binary social spider algorithm is proposed to solve discounted {0-1} knapsack problem (DKP01). DKP01 formula is as the following:

$$\text{Maximize } f(X) = \sum_{i=0}^{n-1}(x_{3i}v_{3i} + x_{3i+1}v_{3i+1} + x_{3i+2}v_{3i+2}); \tag{1}$$

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

$$\text{Subject to } (x_{3i}w_{3i} + x_{3i+1}w_{3i+1} + x_{3i+2}w_{3i+2}) \leq C, \tag{3}$$

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

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 the item j ($j = 0, 1, \ldots, 3m-1$) is not in knapsack, while $x_j = 1$ indicates the item j is in knapsack; w_{3i}, w_{3i+1}, and w_{3i+2} are the weight of items $3i$, $3i+1$, and $3i+2$ respectively. It is worth noting that a binary vector $X = (x_0, x_1, \ldots, x_{3m-1}) \in \{0, 1\}^{3m}$ is a potential solution of DKP01. Only if X meets both Equations (2) and (3), it is a feasible solution of DKP01.

The DKP01 is a new knapsack problem introduced by Guldan [1]. This problem has many applications in investment decision-making, mission selection, and budget control. Dynamic programming for solving DKP01 is first studied in [1]. [2] introduced the DKP01 by using the core concept of the {0-1} knapsack problem, and combined dynamic programming with the core of the DKP01. Two algorithms of FirEGA and SecEGA are proposed by He et al. for DKP01 [3]. Recently, they [4] also had a detailed study of the algorithms of the DKP01 and proposed a brand new deterministic algorithm and approximation algorithms. They proposed PSO-GRDKP based on particle swarm optimization

DOI: 10.24507/icicel.15.03.257

with encoding discrete [5], multi-strategy monarch butterfly optimization algorithm and binary moth search algorithm for discounted {0-1} knapsack problem [6, 7].

Social spider algorithm (SSA) is a new algorithm proposed by Yu and Li for global optimization [8]. SSA inspired the foraging behaviour of the social spider that can be described as the cooperative movement of the spiders towards the food source position. SSA has outperformed other state-of-the-art metaheuristics on many benchmark functions [9, 10, 11, 12, 13].

In this paper, a novel binary social spider algorithm (BSSA) is proposed to solve D-KP01. The proposed algorithm combined the exploration of SSA and the exploitation of a repair operator to solve DKP01. The simulation results on five state-of-the-art benchmark instances data sets demonstrate that the proposed algorithm has superior performance compared with previous algorithms.

The paper is structured in five parts: Section 1 gives an overview of the problem to be solved, Section 2 presents social spider algorithm, Section 3 introduces BSSA algorithm for DKP01, Section 4 introduces the simulation results, and finally the conclusions are drawn in Section 5.

2. Social Spider Algorithm.

SSA [8] may be a metaheuristic propelled by the behavior of social spiders. In SSA, the solution of an optimization issue could be a recreation by the position of another spider on the hyper-dimension spider web. The spiders move on the Internet whereas they share the position data utilizing vibration of its. Depending on the receiving vibration from other ones, direct the spider forward to the ideal position. Points of interest of SSA will be portrayed within the taking after subsections.

2.1. Spider.

The manufactured spiders are the fundamental working specialists of SSA. Each insect has a position on the hyper-dimension spider web, and the wellness esteem of this position is relegated to the spider. Each spider holds memory is putting away its status as well as optimization parameters, to be specific, its current position, a current wellness esteem, taking after vibration at past emphasis, dormant degree, past developments, and measurement cover. All these characters guide the insect to explore for the ideal arrangement.

2.2. Vibration.

Vibration could be an exceptionally vital concept in SSA. It is one of the most characteristics that recognize SSA from other metaheuristics. In SSA, we utilize two properties to characterize a vibration, to be specific, the source position and the source escalated of the vibration.

The source vibration intensity is calculated by Equation (5).

$$I(i) = \log\left(\frac{1}{f(i) - B} + 1\right) \tag{5}$$

where $f(i)$ is the fitness value of spider i, and B is a constant parameter.

Vibration attenuation when transmitting from spider i to spider i' is calculated as Equation (6).

$$I\left(i, i'\right) = I(i) \times \exp\left(\frac{D\left(i, i'\right)}{\overline{\sigma} \times r_a}\right) \tag{6}$$

where $r_a \in (0; \infty)$ is a given factor. This factor controls the attenuation rate of the vibration intensity over distance. $\overline{\sigma}$ is the standard deviation of all spider locations.

2.3. Detail of SSA.

SSA manipulates a population of artificial spiders via a series of optimization steps. Specifically, each iteration of SSA can be divided into the following steps.

At the beginning of each iteration, the fitness values of the positions possessed of all spiders in the population are reevaluated. Then, new vibration is generated for each spider

and propagated to all the other spiders in the population with attenuation. Depending on the receipt vibrations, the largest attenuated vibration intensity is selected, and compare it with the previous one. The best so far intensity vibration is updated. Based on the vibrations, the position of spiders is updated. Each spider holding mask which is a binary vector with length is the solution dimension of the optimization problem. Each iteration, the mask is changed and the solutions are changed based on the mask. Following, a random walk procedure is executed to aim to improve diversity.

3. Proposed Binary Social Spider Algorithm (BSSA) for DKP01.

3.1. Encoding solution.

Original SSA is designed for the real value space. For DKP01, the solution is presented in a binary vector. So SSA is modified to work for discrete binary space. A binary vector with $3 * n$ dimensional is used to present a solution. Bit jth is equal to 1 if item jth is selected, otherwise item jth is not selected.

Algorithm 1: Binary social spider algorithm

Input: Initial parameters
Output: Optimal solution
Initialize parameters: *pop*, v_i^{tar} for each spider $i \in pop$.
While (stop criteria not met)
for *spider* $i \in pop$ **do**
    Caculate objective function.
    Generate a vibration of i.
for *spider* $i \in pop$ **do**
    Compute the power of the vibrations V produced by all spiders.
    Select the best vibration v_i^{best} from V.
    if $v_i^{best} > v_i^{star}$ **then**
        $v_i^{best} \leftarrow v_i^{star}$
    Upadate c_i.
    Create a random number $r \in [0, 1]$.
    if $r > v_c^{c_s}$ **then**
        Update m_i.
    Create v_i^{f0}.
    Excute a random walk proceduce.
    Caculate binary vector X by Equations (7), and (8).
    Handle constraint.

3.2. Real to binary convert function.

In this algorithm, a sigmoid function is used to convert real values to binary values, and the position in the real vector is converted to binary vector by Equation (7).

$$X_{i,j}(t+1) = \begin{cases} 0 & \text{if } rand() \geq TF(v_{i,j}(t+1)) \\ 1 & \text{if } rand() < TF(v_{i,j}(t+1)) \end{cases} \tag{7}$$

where $TF(.)$ is function for converting real vector to binary vector by following equation:

$$TF(v_{i,j}(t+1)) = \frac{1}{1 + e^{v_{i,j}(t+1)}} \tag{8}$$

3.3. **Objective function and repair operator.**

3.3.1. *Objective function.* The DKP01 is maximization problem. However, SSA is designed for a minimization problem. To change DKP01 problem to minimization problem a big constant Θ is added to the fitness function as follows:

$$Objective = \Theta - \sum_{i=0}^{m-1}(x_{3i}v_{3i} + x_{3i+1}v_{3i+1} + x_{3i+2}v_{3i+2}) \tag{9}$$

3.3.2. *Repair operator.* The repair operator includes two phases: DROP phase and ADD phase. When the total weight exceeds the knapsack, the DROP is used. The ADD phase works to improve the quality of the solution when the knapsack is not full. The details of this function can be found in [14].

The advanced repair operator when comparing to penalty function is that the repair function not only repairs the violate solutions, it also helps improve the quality of potential solutions.

4. **Simulation Results.** To evaluate the proposed algorithm, we test on 20 DKP01 instances taken from [14]. Instances f_1-f_{10} are 10 IDKP instances, and f_{11}-f_{20} are 10 SDKP instances taken from [14].

For the BSSA, the parameters are turning by trial and error. The parameters are set as: $r_a = 1$, $v_c = 0.7$, $v_m = 0.1$, and $popsize = 30$.

All the algorithms are implemented in Matlab 2018a. The test environment is set up on a Desktop with Core i5 8250 CPU at 1.6 GHz, 8 G RAM, running on Windows 10 (64 bit).

Figures 1, 2, 3, and 4 show the convergence curves of the best profits of BSSA for instances f_{11}, f_{17}, f_1, and f_7. The BSSA shows better diversification and intensification when it is fast convergence and finds out the better profit value compared with other ones.

Tables 1, and 2 show the experimental results of the instances. We adopt the same termination criterion, and the function evaluation limit is set to $30*3*n$ (n is the number of

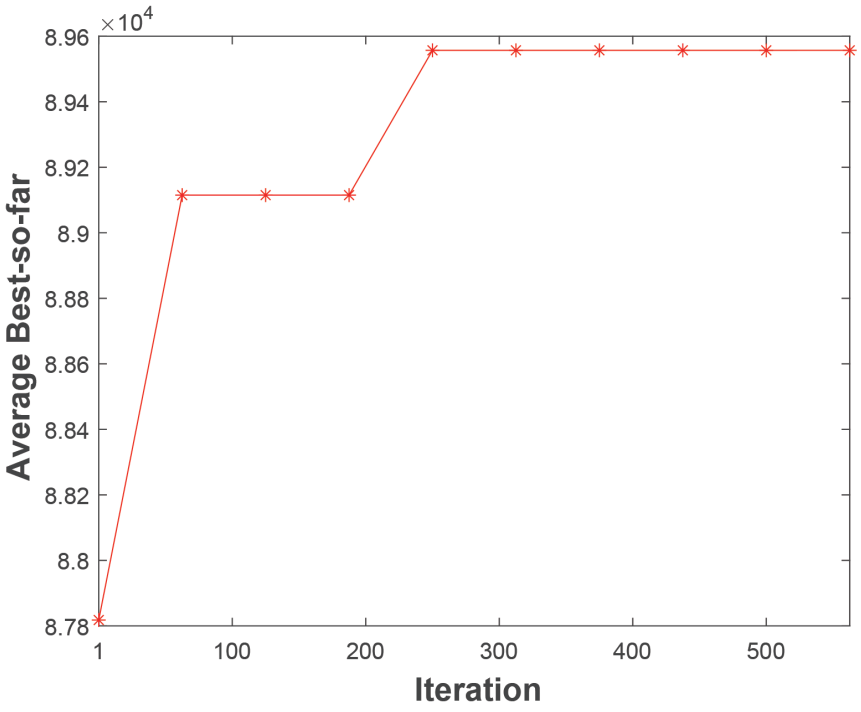

FIGURE 1. The convergence curve test function f_{11}

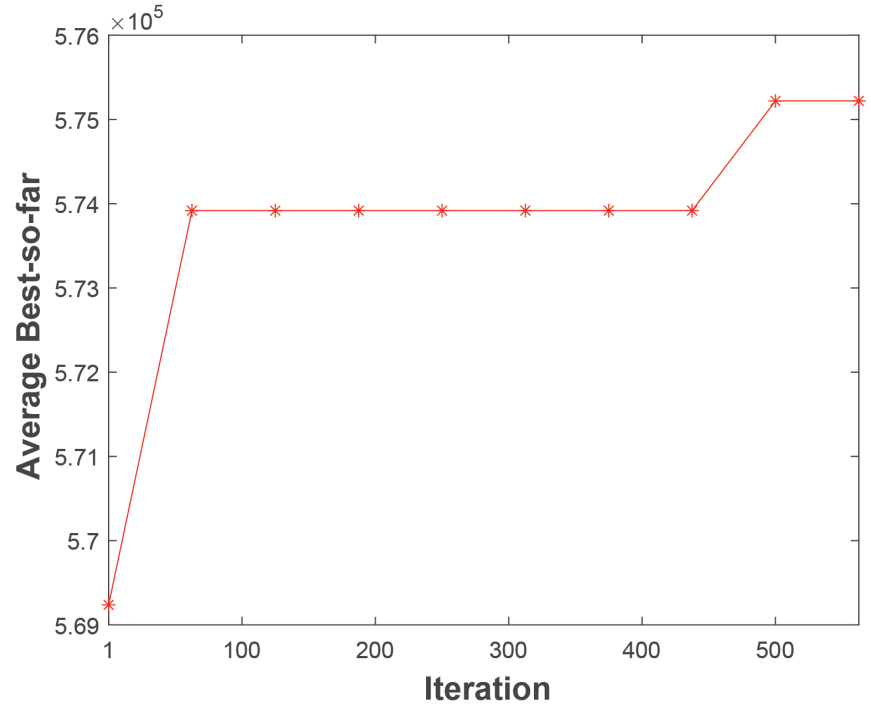

FIGURE 2. The convergence curve test function f_{17}

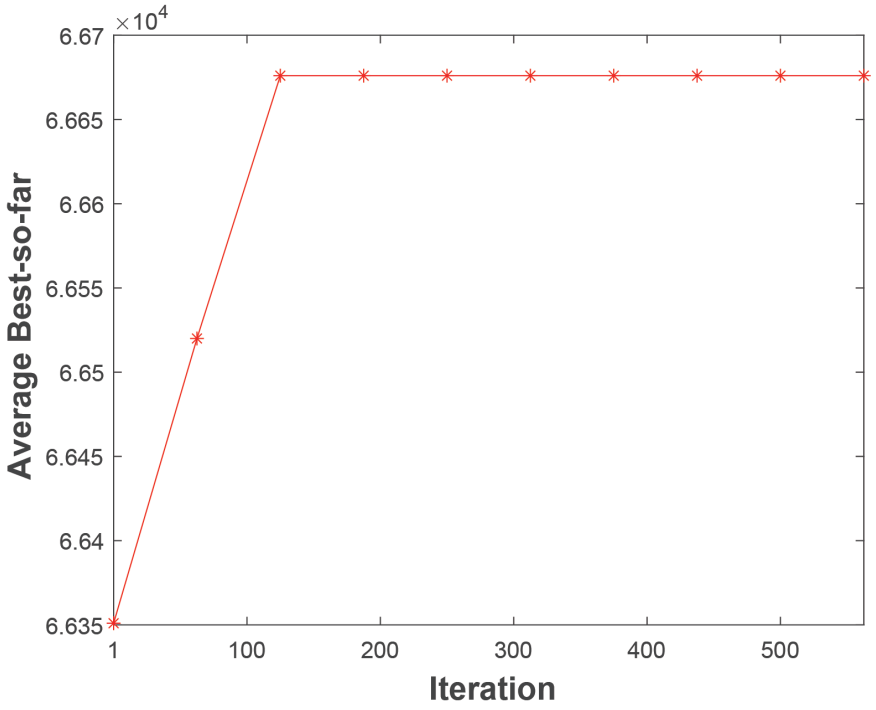

FIGURE 3. The convergence curve test function f_1

groups), for all the tests. For all the instances, the BSSA yields superior results compared with the other ones. The series of experimental results demonstrates the superiority and effectiveness of BSSA. The experimental results show that BSSA outperforms the other algorithms in solution quality. The reason for this superior performance of BSSA is that our proposed algorithm has good search ability and a greedy repair operator.

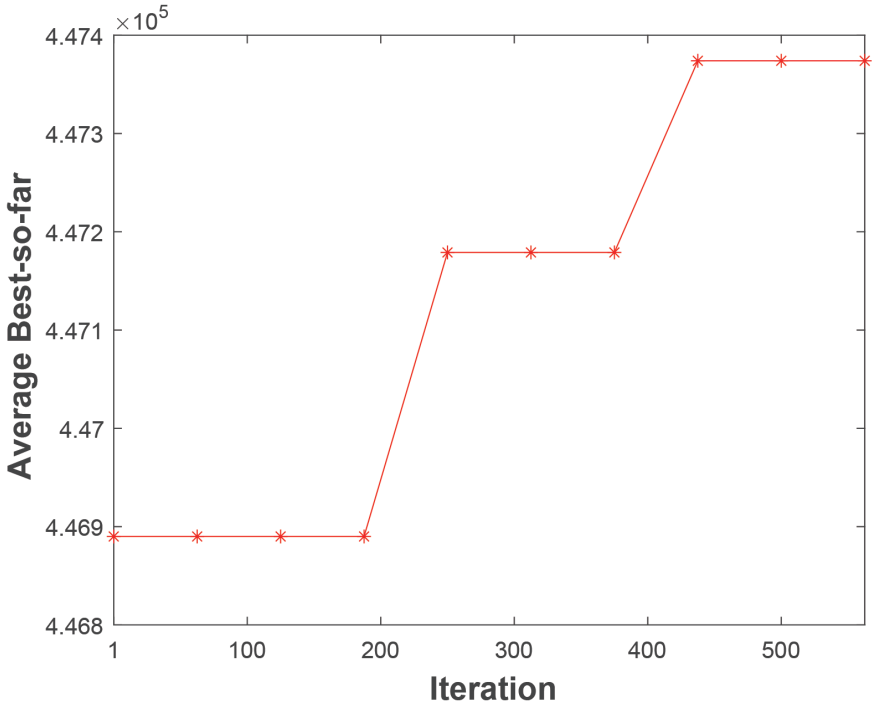

FIGURE 4. The convergence curve test function f_7

TABLE 1. Experiment results of BSSA, and SecGA on f_1-f_{10} (IDKP instances)

Index	Instance	Algorithm	Best	Average	Worst	StdDev
1	f_1	SecGA	68663	68000	67369	328.4
		BSSA	68001	66622	65617	557.2
2	f_2	SecGA	114434	113385	112307	7446.7
		BSSA	112520	111110	110030	655.0
3	f_3	SecGA	220096	217982	216313	835.8
		BSSA	220400	218001	216500	972.3
4	f_4	SecGA	263238	260425	258922	933.4
		BSSA	263430	261630	260490	870.9
5	f_5	SecGA	309573	306878	304881	907.2
		BSSA	311980	308916	306990	1345.0
6	f_6	SecGA	414090	411367	408788	1099.3
		BSSA	418960	417083	415600	694.1
7	f_7	SecGA	451528	444316	442133	1280.3
		BSSA	453120	449744	447760	1336.4
8	f_8	SecGA	490494	481831	478035	2215.7
		BSSA	491750	488140	485610	1477.7
9	f_9	SecGA	489661	477001	471848	3656.2
		BSSA	486230	484368	482730	754.5
10	f_{10}	SecGA	535541	521604	516445	4265.1
		BSSA	534250	530765	528570	1354.3

5. **Conclusion.** This article proposed a binary social spider algorithm with a repair operator to solve the discounted {0-1} knapsack problem efficiently. The repair operator helps the algorithm speed up convergence while improving the quality of the best solution. The proposed algorithm demonstrated good ability in two issues of fast convergence and

TABLE 2. Experimental results of BSSA, and SecGA on f_{11}-f_{20} (SDKP instances)

Index	Instance	Algorithm	Best	Average	Worst	StdDev
1	f_{11}	SecGA	89769	88832	87463	594.91
		BSSA	90622	89666	88518	545.30
2	f_{12}	SecGA	153821	152059	150753	489.39
		BSSA	150880	149119	148110	601.90
3	f_{13}	SecGA	224997	223580	221918	543.38
		BSSA	223610	221278	219890	939.83
4	f_{14}	SecGA	318510	315513	313747	851.14
		BSSA	314660	311071	308960	1321.38
5	f_{15}	SecGA	420238	416964	413933	1291.65
		BSSA	433490	431725	429840	1109.59
6	f_{16}	SecGA	430738	427304	425504	1031.12
		BSSA	425150	422036	418500	1729.48
7	f_{17}	SecGA	561224	556083	552007	1926.26
		BSSA	578680	575727	574030	1120.69
8	f_{18}	SecGA	611644	606263	603774	1446.94
		BSSA	617570	611351	606300	3056.41
9	f_{19}	SecGA	674885	667900	664580	1614.04
		BSSA	672920	669295	665290	1826.39
10	f_{20}	SecGA	708935	695557	691994	2956.08
		BSSA	697260	693486	690500	1768.60

high quality of the found solution. The simulation results on the state-of-the-art benchmark instances proved that the proposed algorithm has superior performance compared with previous algorithms. In the future, we will compare with more the state-of-the-art algorithms.

REFERENCES

[1] B. Guldan, *Heuristic and Exact Algorithms for Discounted Knapsack Problems*, Master Thesis, University of Erlangen-Nürnberg, 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.-C. He, X.-Z. Wang, W.-B. Li, X.-L. Zhang and Y.-Y. Chen, Research on genetic algorithms for discounted {0-1} knapsack problem, *Chinese J. Comput.*, 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.-H. Feng and G.-G. Wang, Binary moth search algorithm for discounted {0-1} knapsack problem, *IEEE Access*, vol.6, pp.10708-10719, 2018.

[7] 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, no.10, pp.3019-3036, 2018.

[8] J. J. Yu and V. O. Li, A social spider algorithm for global optimization, *Applied Soft Computing*, vol.30, pp.614-627, 2015.

[9] J. James and V. O. Li, A social spider algorithm for solving the non-convex economic load dispatch problem, *Neurocomputing*, vol.171, pp.955-965, 2016.

[10] W. Elsayed, Y. Hegazy, F. Bendary and M. El-Bages, Modified social spider algorithm for solving the economic dispatch problem, *Engineering Science and Technology, An International Journal*, vol.19, no.4, pp.1672-1681, 2016.

[11] S. Z. Mirjalili, S. Saremi and S. M. Mirjalili, Designing evolutionary feedforward neural networks using social spider optimization algorithm, *Neural Computing and Applications*, vol.26, no.8, pp.1919-1928, 2015.

[12] M. El-Bages and W. Elsayed, Social spider algorithm for solving the transmission expansion planning problem, *Electric Power Systems Research*, vol.143, pp.235-243, 2017.
[13] A. A. Ewees, M. A. El Aziz and M. Elhoseny, Social-spider optimization algorithm for improving ANFIS to predict biochar yield, *The 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2017.
[14] 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.