IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 19, 2018, accepted September 20, 2018. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2018.2872533

List-Based Simulated Annealing Algorithm With Hybrid Greedy Repair and Optimization Operator for 0-1 Knapsack Problem

SHI-HUA ZHAN, ZE-JUN ZHANG, LI-JIN WANG, AND YI-WEN ZHONG

College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou 350002, China
Key Laboratory of Smart Agriculture and Forestry (Fujian Agriculture and Forestry University), Fujian Province University, Fuzhou 350002, China

Corresponding author: Yi-Wen Zhong (yiwzhong@fafu.edu.cn)

This work was supported in part by the Nature Science Foundation of Fujian Province of China under Grant 2016J01280 and in part by the Special Fund for Scientific and Technological Innovation of Fujian Agriculture and Forestry University under Grant CXZX2016026 and Grant CXZX2016031.

ABSTRACT List-based simulated annealing (LBSA) algorithm, which uses list-based cooling scheme to control the change of parameter temperature, was first proposed for traveling salesman problem. This paper extends the application of LBSA algorithm for 0-1 knapsack problem (0-1 KP). A hybrid greedy repair and optimization operator, which combines density-based and value-based greedy repair and optimization operators, is designed to get better balance between intensification and diversification. Extensive experiments were performed to show the effectiveness and parameter robustness of a list-based cooling scheme and to verify the advantage of using hybrid greedy repair and optimization operator. Comparing experiments, which were conducted on small-scale, medium-scale, and large-scale 0-1 KP instances, have shown that LBSA algorithm is better than or competitive with other state-of-the-art metaheuristics.

INDEX TERMS Greedy repair and optimization, hybrid operator, list-based cooling scheme, list-based simulated annealing, 0-1 knapsack problem.

I. INTRODUCTION

Simulated annealing (SA) algorithm [1], [2] is a classical metaheuristic algorithm which has been widely used to address discrete and continuous optimization problems in many diverse fields. List-based SA (LBSA) algorithm [3] is a novel SA algorithm where list-based cooling scheme is designed to control the decrease of temperature. The applications of LBSA for traveling salesman problem in [3] and [4] have shown that list-based cooling scheme not only has less parameter, but also is more robust.

Knapsack problem (KP) is one of the classical NP-hard problems in combinatorial optimization. There are many variants of KP such as 0-1 KP, bounded KP, unbound KP, multiply-constrained KP, multidimensional KP, Set-Union KP, multiple KP, multiple-choice KP, quadratic KP, on-line KP, discounted KP and other types of KPs. Among all the variants, the simplest one is the 0-1 KP. Informally, the objective of 0-1 KP is to maximize the sum of the values of the items in the knapsack under the constraint that the weights is less than or equal to the knapsack's capacity. The 0-1 KP can be defined as follows. Given a set of n items numbered from 1 up to n, each with a weight w_i and a value v_i, along with a maximum weight capacity W of the knapsack, the objective of the 0-1 KP is to maximize

$$f(x) = \sum_{i=1}^{n} v_i x_i$$
$$s.t. \sum_{i=1}^{n} w_i x_i \leq W \wedge x_i \in \{0, 1\} \quad (1)$$

where x_i indicates whether item i is in the knapsack.

Although several exact algorithms are available to solve the 0-1 KP, based on dynamic programming approach, branch and bound approach or hybridizations of both approaches, those exact algorithms perform poorly for high-dimensional 0-1 KP instances because the searching space increases exponentially with respect to the number of items. As for most NP-hard problems, it may be enough to find workable solutions even if they are not optimal. As a result, many metaheuristics have been proposed to solve the 0-1 KP. Although SA algorithm is a simple and efficient metaheuristic for many combinatorial optimization problems, strangely,

little attention has been put on SA algorithm in the field of the 0-1 KP.

Aiming to extend the application of LBSA algorithm and to provide a simple and efficient metaheuristic for solving the 0-1 KP, this paper proposes a LBSA algorithm with hybrid greedy repair and optimization operator. In LBSA, list-based cooling scheme is used to control the change of parameter temperature. Furthermore, hybrid of two greedy repair and optimization operators is designed to prevent LBSA from premature convergence. Extensive experiments were performed to analyze the behaviors of the proposed algorithm. Comparing experiments, which were conducted on small-scale, medium-scale, and large-scale 0-1 KP instances, have shown that LBSA is better than or competitive with other state-of-the-art metaheuristics.

The rest of this paper is organized as follows. Section II provides a short description of SA algorithm, list-based cooling scheme, and metaheuristics for the 0-1 KP. Section III presents the proposed LBSA algorithm for the 0-1 KP. Section IV analyzes the behaviors of LBSA algorithm for the 0-1 KP. Section V compares the performance of LBSA algorithm with other state-of-the-art metaheuristics on a large number of 0-1 KP instances. Finally, in Section VI we summarize our study.

II. RELATED WORK

A. SIMULATED ANNEALING ALGORITHM

SA algorithm is a typical local search algorithm that has explicit strategy to avoid local minima. The fundamental idea is to accept moves resulting in solutions of worse quality than the current solution in order to escape from local minima. The probability of accepting such a move is decreased during the search through parameter temperature. In SA algorithm, The Metropolis acceptance criterion is used to decide whether to accept a candidate solution. Suppose x is the current solution and y is the candidate solution selected from x's neighbors. Their objective function values are $f(x)$ and $f(y)$ respectively. Suppose the optimization problem is a maximization problem, then the candidate solution y is accepted as the current solution x based on the acceptance probability:

$$p = \begin{cases} 1, & if\ f(y) \geqslant f(x) \\ e^{(f(y)-f(x))/t}, & otherwise \end{cases} \tag{2}$$

where $t > 0$ is the parameter temperature.

In order to apply the SA algorithm to a specific optimization problem, one must specify the cooling schedule. A cooling schedule consists of starting temperature, temperature decrement function, Markov chain length, and termination condition. Geometric cooling schedule, which can be described by the temperature-update formula $t_{k+1} = \alpha * t_k$, is probably the most commonly used in the SA literature. Alg. 1 is pseudo code of homogeneous SA algorithm with geometric cooling schedule. In alg. 1, G is total generation and M is Markov chain length of each temperature.

Algorithm 1 Simulated Annealing Algorithm

```
Set up initial temperature t and decrease rate α
Create an initial solution x randomly
for g = 1 to G do
  for m = 1 to M do
    Select a solution y from neighbors of x
    Use (2) to calculate acceptance probability p
    Produce a random number r ∈ [0, 1)
    if r < p then
      x = y
    end if
    Record best solution found
  end for
  t = α * t
end for
return Best solution found
```

B. LIST-BASED COOLING SCHEME

List-based cooling scheme [3] is a novel parameter control strategy for SA algorithm. In list-based cooling schedule, all temperatures are stored in a priority queue. In each iteration k, the maximum value t_{max} in the queue is used as current temperature to calculate acceptance probability for candidate solution. For homogeneous SA algorithm, if the Markov chain length (MCL) is M, then t_{max} will be used M times to calculate the acceptance probability. Suppose there are c times when worse solution is accepted, let $d_i = f(y) - f(x)$ and let p_i represents the acceptance probability, where $i \in \{1, \ldots, c\}$, and the relation between d_i and p_i is described by (2).

In SA algorithm, whenever a new solution is produced, a random number r is created and is compared with p_i to decide whether to accept this new solution. If r is less than p_i, then the new solution will be accepted. Therefore for each pair of d_i and p_i, there is a random number r_i and r_i is less than p_i. A new temperature t_i can be calculated by d_i and r_i as the following equation:

$$t_i = d_i / ln(r_i) \tag{3}$$

List-based cooling scheme uses the average of t_i to replace the t_{max} in the temperature list. Because t_i is always less than t_{max}. Thus the average of t_i is also less than t_{max}. As a result, the temperature values in the list will become lower and lower as the search proceed.

C. METAHEURISTICS FOR THE 0-1 KP

The 0-1 KP is a constrained optimization problem, and the solutions produced by metaheuristics may be infeasible. In general, both penalty function and repair strategy can be used to handle constrains. But in the field of the 0-1 KP, repair strategy is the most widely used strategy and is very effective. The most widely used repair strategy is a combination of greedy repair operator and greedy optimization operator. Suppose items are sorted in non-ascending order of some metric and are stored in an array H. Algs. 2-3 are greedy

repair operator and greedy optimization operator respectively. The most widely used metric is density which is defined as v_i/w_i for each item i, other metrics may include v_i or $1/w_i$. Suppose density is used as metric, then the greedy repair operator examines each item in non-decreasing order of v_i/w_i and changes x_i from one to zero if the constrain is violated, and the greedy optimization operator examines each item in non-increasing order of v_i/w_i and changes x_i from zero to one if selection of item i will not violate the constrain.

Algorithm 2 Greedy Repair Operator

Input: H The sorted items

1: $c = \sum_{i=1}^{n} w_i x_i$
2: **for** $i = n$ to 1 **do**
3: **if** $x_{H[i]} = 1$ and $c > W$ **then**
4: $x_{H[i]} = 0$
5: $c = c - w_{H[i]}$
6: **end if**
7: **end for**

Algorithm 3 Greedy Optimization Operator

Input: H The sorted items

1: $c = \sum_{i=1}^{n} w_i x_i$
2: **for** $i = 1$ to n **do**
3: **if** $x_{H[i]} = 0$ and $c + w_{H[i]} \leq W$ **then**
4: $x_{H[i]} = 1$
5: $c = c + w_{H[i]}$
6: **end if**
7: **end for**

Many metaheuristics have been proposed to solve the 0-1 KP. Abdel-Basset *et al.* [5] proposed a whale optimization algorithm (WOA) for the 0-1 KP and MKP. In WOA, a penalty function is added to the evaluation function so that the fitness of the feasible solutions can outperform the fitness of the infeasible ones. Bhattacharjee and Sarmah [6] proposed a modified discrete shuffled frog leaping algorithm where a mutation operator with a small probability is applied to modify the original population to avoid the premature convergence. Gherboudj *et al.* [7] proposed a discrete binary cuckoo search (DBCS) algorithm for the 0-1 KP and MKP. DBCS uses sigmoid function to transform a real number to a binary solution. Bhattacharjee and Sarmah [8] proposed a binary cuckoo search (BCS) algorithm and a binary firefly algorithm (BFA) for the 0-1 KP and MKP. In BCS, a balanced combination of local random walk and global explorative random walk is used along with the repair operator. In BFA, the variable distance move with the repair operator of the local search and opposition-based learning mechanism is applied. Nguyen *et al.* [9] proposed a novel binary social spider algorithm (BSSA) for the 0-1 KP. In BSSA, sigmoid function is used to convert real values to binary values and repair operator is used to repair infeasible solution and optimize the feasible solution. Kulkarni and Shabir [10] proposed a cohort intelligence (CI) algorithm for the 0-1 KP. Han and Liu [11] proposed an improved binary chicken swarm optimization (BCSO) algorithm for the 0-1 KP. BCSO uses sigmoid function to transform a real number to a binary solution and uses mutation operator to increase the population diversity. Zhou *et al.* [12] proposed a binary monkey algorithm (BMA) for the 0-1 KP. In BMA, somersault process is designed to avoid premature convergence, and cooperation process is designed to speed up the convergence speed. Gao *et al.* [13] proposed a quantum-inspired wolf pack algorithm which uses quantum rotation and quantum collapse to produce new solution. Yassien *et al.* [14] proposed a grey wolf optimization (GWO) where K-means clustering algorithm is used to group each 5-12 agents with each other at one cluster according to GWO constraint. Rizk-Allah and Hassanien [15] proposed novel binary bat algorithm (NBBA) where local search scheme is used to enhance the exploration capability. Yampolskiy and El-Barkouky [16] proposed a wisdom of artificial crowds (WoAC) algorithm where WoAC is used to refine the results obtained by GA. Wu *et al.* [17] proposed a hybrid symbiotic organisms search algorithm where greedy strategy is employed to repair the infeasible solution and optimize the feasible solution. Cao *et al.* [18] proposed a binary artificial bee colony algorithm with differential evolution approach for the 0-1 KP.

Abdel-Basset and Zhou [19] proposed an elite opposition-flower pollination algorithm (EFPA) for the 0-1 KP. In EFPA, global elite opposition based learning is adopted to enhance its exploration ability, and local self-adaptive greedy strategy is used to enhance its exploitation ability. Truong *et al.* [20] proposed a chemical reaction optimization with greedy strategy (CROG) for the 0-1 KP. In CROG, binary string is used to represent a solution, and repair operator with greedy add and random drop is used to repair infeasible solution and optimize the feasible solution. Similar solution representation and repair operator are adopted by Truong *et al.* [21] in artificial chemical reaction optimization algorithm with a greedy strategy for the 0-1 KP. Zhao *et al.* [22] proposed a genetic algorithm (GA) for the 0-1 KP where three greedy strategies are used to produce initial population. Umbarkar and Joshi [23] proposed a dual population GA (DPGA) for the 0-1 KP where the main goal of reserve population is to keep enough diversity and to prevent DPGA from premature convergence. Zhou *et al.* [24] proposed a complex-valued encoding wind driven optimization for the 0-1 KP where the complex number needs to be converted into a real number, and then the real number is transferred into 0 or 1.

Feng *et al.* [25] proposed an improved hybrid encoding cuckoo search algorithm (ICS) for the 0-1 KP. In ICS, position updating with adaptive step and genetic mutation are used to produce new position vector and sigmoid function is used to transform a real-coded vector in to a binary vector. Furthermore, greedy transform method is introduced to repair infeasible solution and optimize the feasible solution. Same encoding and greedy transform method are used in an effective hybrid cuckoo search algorithm with improved shuffled

frog leaping algorithm [26], in a novel hybrid cuckoo search algorithm with global harmony search (CSGHS) [27], in a novel binary monarch butterfly optimization (BMBO) [28], in chaotic monarch butterfly optimization (CMBO) algorithm with Gaussian mutation [29], and in opposition-based learning monarch butterfly optimization (OMBO) with Gaussian perturbation [30].

Several harmony search algorithms have been proposed for solving the 0-1 KP, such as global harmony search algorithm [31], hybrid quantum inspired harmony search algorithm [32], harmony search algorithm based on teaching-learning strategies [33], discrete global-best harmony search algorithm [34], and simplified binary harmony search algorithm [35].

III. LBSA FOR THE 0-1 KP

A. INITIAL TEMPERATURE LIST AND NEIGHBOR OPERATOR

To apply LBSA for the 0-1 KP, we must specify the way to create initial temperature list and the way to produce neighbor solution. In this paper, we use the values of items as the initial temperature values in temperature list. If the number of items is less than the length of temperature list, then the values will be used repeatedly. Owing to the robustness of LBSA algorithm, the produced temperature list is suitable enough for LBSA algorithm to have good performance. A simple flip operator is used to produce a neighbor solution, and then hybrid greedy repair and optimization operator is used to modify this new solution. Alg. 4 is the pseudo code of neighbor operator with hybrid greedy repair and optimization. In alg. 4, variable HD is an array which stores items sorted in non-ascending order of density v_i/w_i, variable HV is an array which stores items sorted in non-ascending order of value v_i, and variable PG is a parameter which represents the probability to use HD to call greedy repair and optimization operator. Apparently, big PG leads to high intensification, and small PG leads to high diversification. To obtain good balance between intensification and diversification, LBSA uses variable PG which is linearly increased from 0.9 to 1.

Algorithm 4 Neighbor Operator With Hybrid Greedy Repair and Optimization

Input: x A solution, PG Probability to use HD to call greedy repair and optimization operator
Output: a new solution

```
1:  y = x
2:  Randomly select an item i
3:  if y_i = 0 then
4:      y_i = 1
5:  else
6:      y_i = 0
7:  end if
8:  Produce a random number p ∈ [0, 1)
9:  if p < PG then
10:     Use HD to call alg. 2 to repair y
11:     Use HD to call alg. 3 to optimize y
12: else
13:     Use HV to call alg. 2 to repair y
14:     Use HV to call alg. 3 to optimize y
15: end if
16: Return y
```

B. PSEUDO CODE OF PROPOSED LBSA ALGORITHM

LBSA uses a simple SA framework based on fixed iteration times for outer loop and fixed Markov chain length in each temperature. The detailed pseudo code of the LBSA algorithm is listed in Alg. 5. In Alg. 5, variable L, G and M represent the length of temperature list, the maximum iteration times of LBSA algorithm and the Markov chain length in each temperature respectively. Variable c is used to record how many times a bad solution is accepted in each temperature and variable s is used to store the sum of temperatures calculated by (3). The average temperature s/c is used to update the temperature list in line 27. Variable pg represents the probability to use density-based greedy repair and optimization in Alg. 4, and it is linearly increased by $step$. As a result, pg increases from 0.9 to 1 along the search process. In Alg. 5, line 1 to line 4 are used to create initial temperature list. The main structure of Alg. 5 is a nested loop. The iteration times of outer loop is G, and the iteration times of inner loop is M for each outer loop. In each iteration of inner loop, Alg. 4, whose time complexity is $O(N)$, is called to produce a neighbor solution. As a result, the time complexity of LBSA algorithm is $O(G \times M \times N)$.

IV. BEHAVIORS ANALYSIS OF LBSA ALGORITHM

In order to analyze the behaviors of LBSA algorithm for the 0-1 KP, three experiments have been carried on six large-scale 0-1 KP instances. The first experiment is used to verify the necessity to use probability-based acceptance and to find a suitable length for initial temperature list. The second experiment is used to analyze the convergence behaviors of LBSA algorithm. The third experiment is used to analyze the effect of hybrid greedy repair and optimization operator. These three experiments were performed on three groups of instances used by [29]. The features of the three groups of instances are list in table 1. Each group contains five 0-1 KP instances and the number of items is 800, 1000, 1200, 1500 and 2000, respectively. These fifteen instances are called KP01, KP02, . . ., KP15, respectively. Among those fifteen instances, KP01 to KP05 are uncorrelated instances, KP06 to KP10 are weakly-correlated instances, and KP11 to KP15 are strongly-correlated instances. The six instances used in this section are KP01, KP04, KP06, KP09, KP11, and KP14. Except in the experiments for parameter tuning, the length of temperature list is 100, and the probability to use density-based greedy repair and optimization operator

TABLE 1. Features of three group 0-1 KP instances.

Correlation	Weight w_i	Value v_i	Capacity W
Uncorrealted	rand(10, 100)	rand(10, 100)	0.75 * sum of weights
Weakly-correlated	rand(10, 100)	rand($w_i - 10$, $w_i + 10$)	0.75 * sum of weights
Strongly-correlated	rand(10, 100)	$w_i + 10$	0.75 * sum of weights

Algorithm 5 List-Based SA Algorithm for the 0-1 Knapsack Problem

```
Create an empty temperature list list
for i = 1 to L do
    Put the value v_i of item i into list
end for
Randomly produce a solution x.
Set best solution bx to x
pg = 0.9 //Initial probability to use density-based greedy repair and optimization
step = 0.1/G //Step used to increase pg linearly to 1
for g = 1 to G do
    t = The maximum value in list
    s = 0, c = 0.
    for m = 1 to M do
        Use x and pg to call Alg. 4 to produce a candidate solution y
        Use (2) to calculate acceptance probability p of y
        Produce a random number r ∈ [0, 1)
        if r < p then
            if f(y) < f(x) then
                s = s + (f(y) − f(x))/ln(r)
                c = c + 1
            else if f(y) > f(bx) then
                bx = y
            end if
            x = y
        end if
    end for //end of inner loop
    if c > 0 then
        Replace the maximum value in list with s/c
    end if
    pg = pg + step //Increase pg by step
end for //end of for each agent
Return bx
```

linearly increases from 0.9 to 1. In all the experiments, the termination condition is 200 generations and the Markov chain length M is set to 200. Difference between solution found and the best known solution is used to compared LBSAs with different parameter.

A. PARAMETER TUNING FOR LENGTH OF INITIAL TEMPERATURE LIST

To verify the necessity to use probability-based acceptance and to find a suitable list length, we test 6 different list lengths, 1, 50, 100, 150, 200, and ∞ for each instance. If the list length is 1, then the maximum value of all items is used as initial temperature. If the list length is ∞, then LBSA will always use the maximum value of all items as current temperature. For each list length, we run LBSA algorithm 100 times and calculate the average profit. Fig. 1 to Fig. 6 are the box-plots of difference between solution found and the best known solution with different list length on KP01, KP04, KP06, KP09, KP11, and KP14 respectively. It is shown that: (1) the performance of LBSA with list length equal to 1 is always the worst. If the list length is 1, the temperature will decrease quickly, the search behavior of LBSA is similar to hill-climbing algorithm. It means LBSA will fall in local optima easily; (2) there is a wide range of list length for LBSA to have good performance; (3) if the list length is ∞, the performance of LBSA on KP01, KP04, KP06, and KP09 becomes worse slightly. According to the simulation results, the temperature list length is set to 100 in the following simulations.

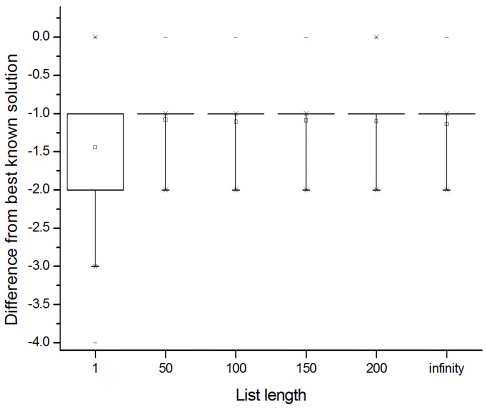

FIGURE 1. Effect of list length on KP01.

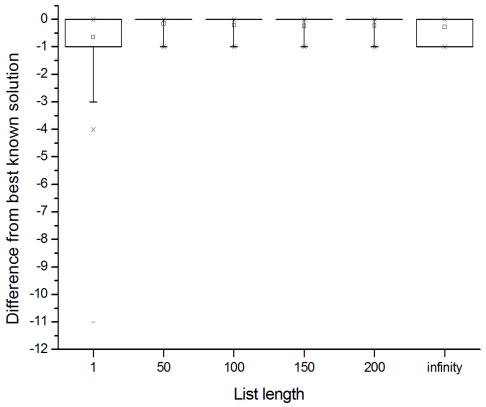

FIGURE 2. Effect of list length on KP04.

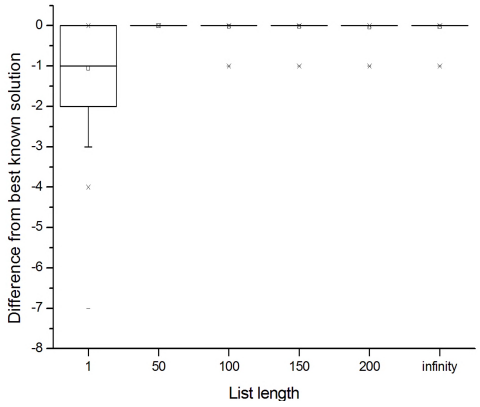

FIGURE 3. Effect of list length on KP06.

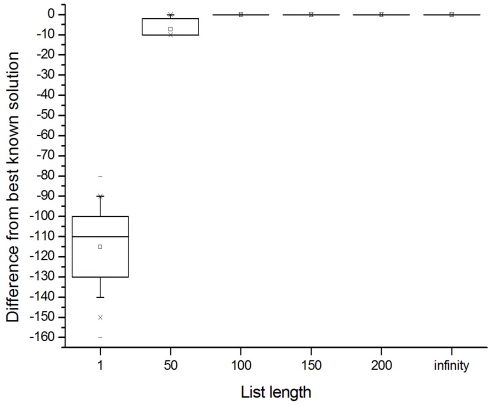

FIGURE 6. Effect of list length on KP14.

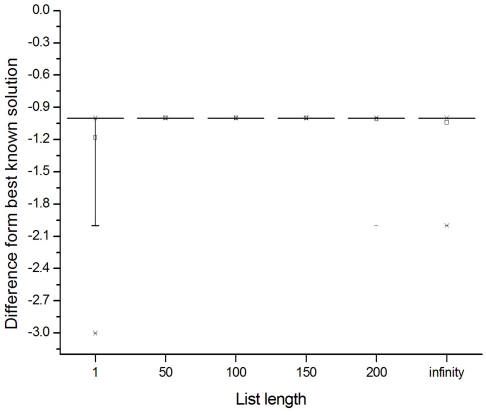

FIGURE 4. Effect of list length on KP09.

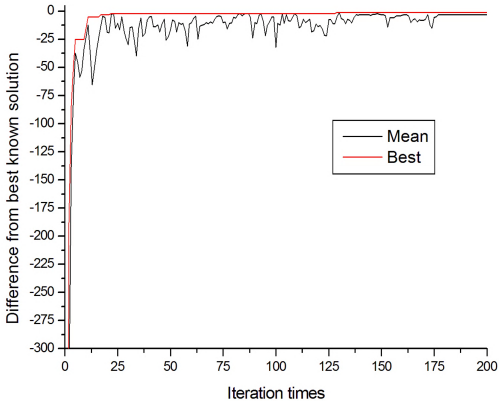

FIGURE 7. Convergence process on KP01.

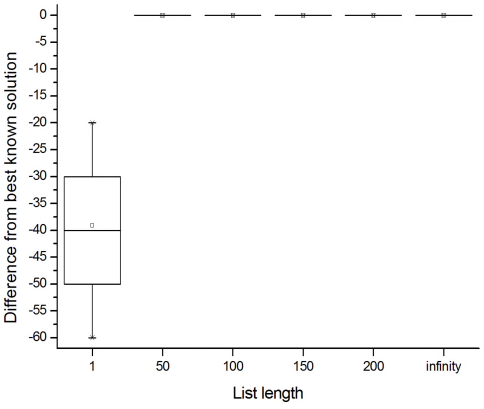

FIGURE 5. Effect of list length on KP11.

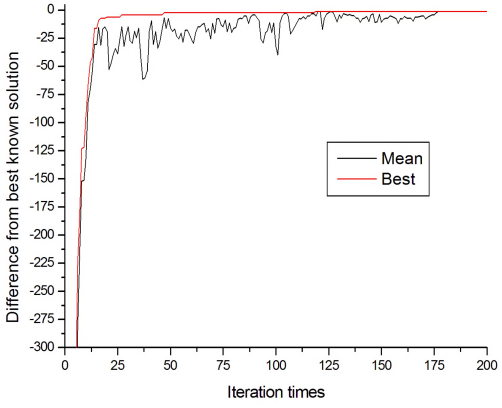

FIGURE 8. Convergence process on KP04.

B. CONVERGENCE ANALYSIS

To analyze the convergence behaviors of LBSA algorithm, we draw the convergence process of best solution and average solution at each temperature. The used temperature list lengths includes 120, 160, and 220. Fig. 7 to Fig. 12 are the convergence process of LBSA algorithm on KP01, KP04, KP06, KP09, KP11, and KP14 respectively. Those figures clearly show that: (1) the best solution converges quickly on all three types of 0-1 KP instances; (2) LBSA algorithm maintains better diversity on strongly-correlated instances (KP11 and KP14), this may be the reason why LBSA is most robust on strongly-correlated instances.

C. EFFECT OF HYBRID GREEDY REPAIR AND OPTIMIZATION

To analyze the effectiveness of hybrid greedy repair and optimization operator, we compare the performance of LBSA algorithm with different probability of density-based

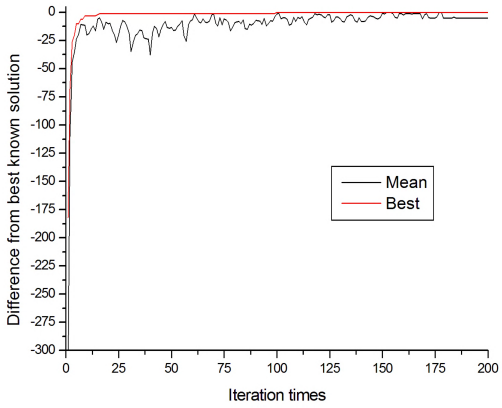

FIGURE 9. Convergence process on KP06.

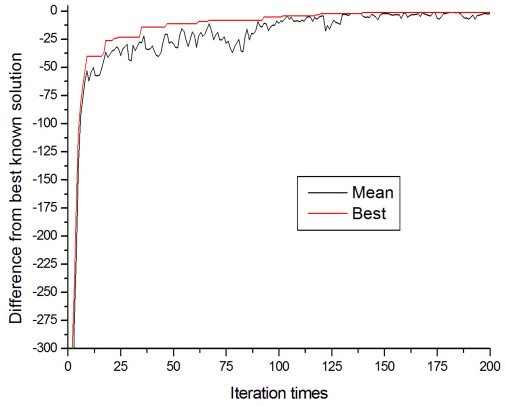

FIGURE 10. Convergence process on KP09.

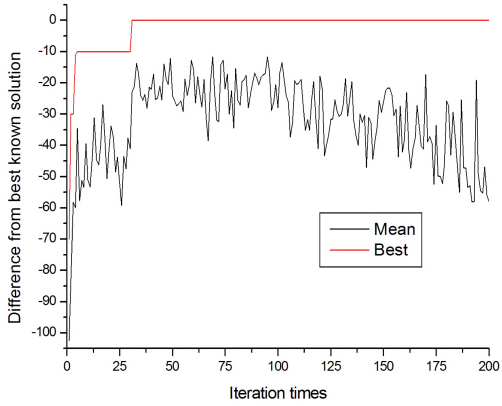

FIGURE 11. Convergence process on KP11.

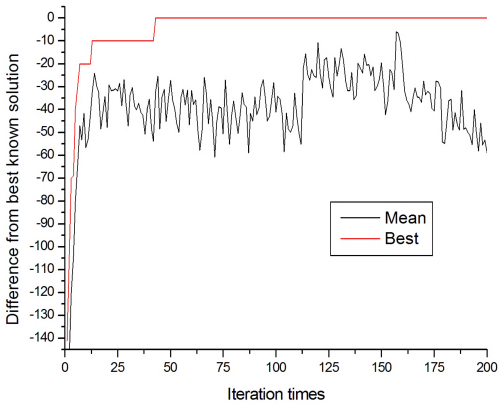

FIGURE 12. Convergence process on KP14.

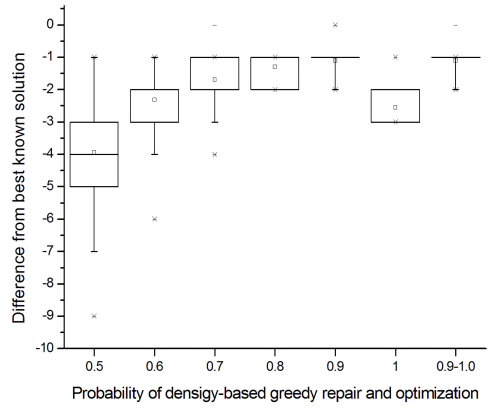

FIGURE 13. Effect of hybrid greedy repair and optimization operator on KP01.

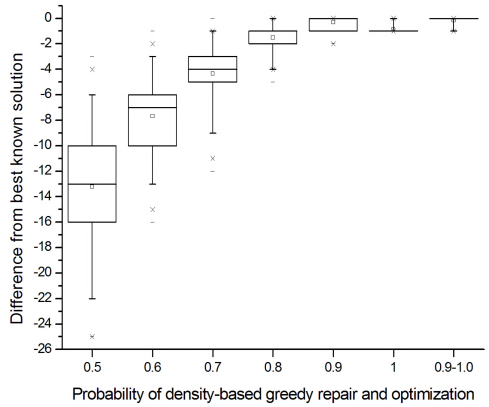

FIGURE 14. Effect of hybrid greedy repair and optimization operator on KP04.

greedy repair and optimization operator. Those used probabilities include 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, and linearly increasing from 0.9 to 1.0. Fig. 13 to Fig. 18 are the boxplots of difference between solution found and the best known solution with different probability on KP01, KP04, KP06, KP09, KP11, and KP14 respectively. From those figures we have: (1) the probability of density-based greedy repair and optimization should not be too small; (2) except on KP09 and KP14 instances, to always use density-based greedy repair and optimization is worse than to use it with 0.9 probability; (3) linearly increasing probability pg from 0.9 to 1.0 has best results in all cases.

V. COMPETITIVENESS OF LBSA ALGORITHM

In order to observe the competitiveness of LBSA algorithm, LBSA's performance was compared with some of

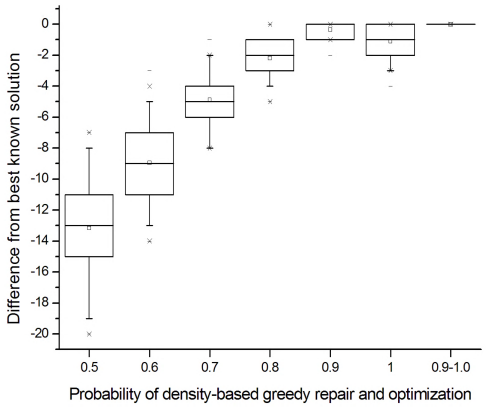

FIGURE 15. Effect of hybrid greedy repair and optimization operator on KP06.

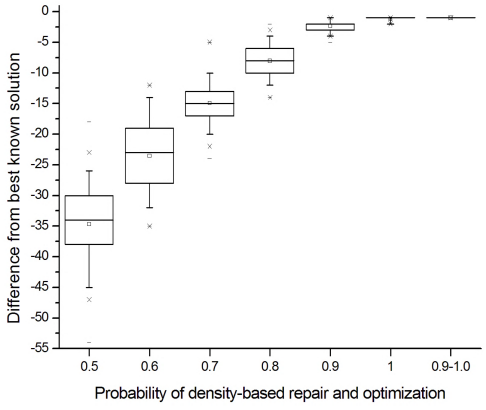

FIGURE 16. Effect of hybrid greedy repair and optimization operator on KP09.

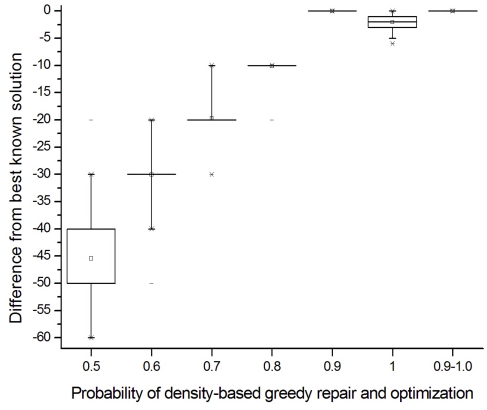

FIGURE 17. Effect of hybrid greedy repair and optimization operator on KP11.

other state-of-the-art algorithms on small-scale, medium-scale, and large-scale 0-1 KP instances. In all the following experiments, the length of temperature list is 100, the maximum generation is 200, and the sampling times in each temperature is 200. The probability to use

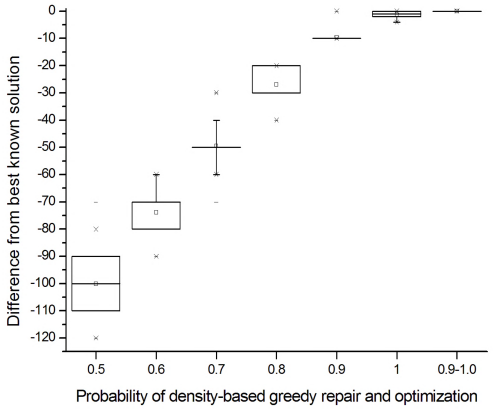

FIGURE 18. Effect of hybrid greedy repair and optimization operator on KP14.

density-based greedy repair and optimization operator linearly increases from 0.9 to 1. The following experiments were run on an Intel Core i5-4590 CPU, with 3.3 GHz and a RAM of 4 GB. Java was used as the programming language.

A. SIMULATION ON SMALL-SCALE 0-1 KP INSTANCES

LBSA was tested on ten standard small-scale 0-1 KP instances [31] where the number of items is 10, 20, 4, 4, 15, 10, 7, 23, 5, and 20 respectively. The performance of LBSA was compared with five state-of-the-art metaheuristics which are published in recent two years. Those five metaheuristics include binary cuckoo search algorithm (BCSA) [8], binary monarch butterfly optimization (BMBO) algorithm [28], novel binary bat algorithm (NBBA) [15], binary artificial bee colony algorithm with differential evolution (BABC) [18], and complex-valued encoding wind driven optimization (CWDO) [24]. The performance comparison is listed in table 2. Table 2 shows that almost all algorithms can always find the best known solutions on all instances. It means small-scale 0-1 KP instances are easy for metaheuristics to find the optima.

B. SIMULATION ON MEDIUM-SCALE 0-1 KP INSTANCES

The performance of LBSA was tested on ten medium-scale 0-1 KP instances and compared with cohort intelligence (CI) algorithm [10] and novel binary bat algorithm (NBBA) [15]. The ten medium instances are taken from [10] and [15] where the number of items is 30, 35, 40, 45, 50, 55, 60, 65, 70, and 75 respectively. It must be pointed out that the F15 instance used in [10] and [15] is different. In F15 instance used in [10], the values of 55 items is {78 69 87 59 63 12 22 4 45 33 29 50 19 94 95 60 1 91 69 8 100 **32 81 47 59 48 56 18 59 16 45 54 47** ***84 100*** **98 75 20 4 19 58 63 37 64 90 26 29 13 53 83**}. In F15 instance used in [15], the values of 55 items is {78 69 87 59 63 12 22 4 45 33 29 50 19 94 95 60 1 91 69 8 100 ***84 100*** **32 81 47 59 48 56 18 59 16 45 54 47 98 75 20 4 19 58 63 37 64 90 26 29 13 53 83**}. The values in bold

TABLE 2. Performance comparison on ten small-scale 0-1 KP instances.

Ins	Method	Best	Worst	Mean	Ins	Method	Best	Worst	Mean
F1	BCSA	295	295	295	F6	BCSA	52	52	52
	BMBO	295	295	295		BMBO	52	52	52
	NBBA	295	295	295		NBBA	52	52	52
	BABC	295	295	295		BABC	52	52	52
	CWDO	295	295	295		CWDO	52	52	52
	LBSA	295	295	295		LBSA	52	52	52
F2	BCSA	1024	1024	1024	F7	BCSA	107	107	107
	BMBO	1024	1024	1024		BMBO	107	107	107
	NBBA	1024	1024	1024		NBBA	107	107	107
	BABC	1024	1024	1024		BABC	107	107	107
	CWDO	1024	1024	1024		CWDO	107	107	107
	LBSA	1024	1024	1024		LBSA	107	107	107
F3	BCSA	35	35	35	F8	BCSA	9767	9767	9767
	BMBO	35	35	35		BMBO	9767	9765	9766.12
	NBBA	35	35	35		NBBA	9767	9767	9767
	BABC	35	35	35		BABC	9767	9767	9767
	CWDO	35	35	35		CWDO	9767	9767	9767
	LBSA	35	35	35		LBSA	9767	9767	9767
F4	BCSA	23	23	23	F9	BCSA	130	130	130
	BMBO	23	23	23		BMBO	130	130	130
	NBBA	23	23	23		NBBA	130	130	130
	BABC	23	23	23		BABC	130	130	130
	CWDO	23	23	23		CWDO	130	130	130
	LBSA	23	23	23		LBSA	130	130	130
F5	BCSA	481.07	481.07	481.07	F10	BCSA	1025	1025	1025
	BMBO	481.07	481.07	481.07		BMBO	1025	1025	1025
	NBBA	481.07	481.07	481.07		NBBA	1025	1025	1025
	BABC	481.07	481.07	481.07		BABC	1025	1025	1025
	CWDO	481.07	481.07	481.07		CWDO	1025	1025	1025
	LBSA	481.07	481.07	481.07		LBSA	1025	1025	1025

TABLE 3. Performance comparison on ten medium-scale 0-1 KP instances.

Ins	Method	Best	Worst	Mean	Ins	Method	Best	Worst	Mean
F11	CI	1437	1398	1418	F16	CI	2643	2581	2605
	NBBA	1437	1437	**1437**		NBBA	2643	2632	2642.60
	LBSA	1437	1437	**1437**		LBSA	**2651**	**2651**	**2651**
F12	CI	1689	1679	1686.5	F17	CI	2817	2905	2915
	NBBA	1689	1689	**1689**		NBBA	2917	2917	**2917**
	LBSA	1689	1689	**1689**		LBSA	2917	2917	**2917**
F13	CI	1816	1791	1807.5	F18	CI	2814	2716	2773.66
	NBBA	1821	1821	**1821**		NBBA	2818	2814	2817.63
	LBSA	1821	1821	**1821**		LBSA	2818	2818	**2818**
F14	CI	2020	2007	2017	F19	CI	3221	3211	3216
	NBBA	2033	2033	**2033**		NBBA	3223	3219	3212.9
	LBSA	2033	2033	**2033**		LBSA	3223	3223	**3223**
F15	CI	2440	2421	2436.17	F20	CI	3614	3591	3603.8
	NBBA	2448	2448	2448		NBBA	3614	3605	3613.23
	LBSA	**2449**	**2449**	**2449**		LBSA	3614	3614	**3614**

are different. The difference is caused by the misplacement of values 84 and 100. LBSA uses the F15 instance from [15]. The performance comparison is listed in table 3. The best Mean and the new Best are highlighted in bold. LBSA can always find best solution on all instances. Table 3 shows that NBBA and LBSA always have better performance than CI in terms of mean solution, and LBSA is better than NBBA on 5 instances. It is worth noting that LBSA found new best solution on F15 instance, the new best solution is {111110001111011101101111111110101111110111111110 1 1}. The best solution of F16 instance is {1110011111011 111 0111111010011111111110011011011101010111}, and its profit is 2651.

C. SIMULATION ON LARGE-SCALE 0-1 KP INSTANCES

The fifteen large-scale 0-1 KP instances introduced in section IV are used to compare the performance of LBSA with hybrid cuckoo search algorithm with global harmony search (CSGHS) [27], binary monarch butterfly optimization (BMBO) algorithm [28], chaotic monarch butterfly optimization (CMBO) algorithm [29], and opposition-based learning monarch butterfly optimization (OMBO) algorithm [30]. The performance comparison is listed in tables 4-6, where the best values are highlighted in bold. CSGHS and BMBO were not tested on instances with 2000 items. LBSA costs less time than other metaheuristics. The running time of LBSA is less than 0.4 second. The maximum running time

TABLE 4. Performance comparison on large-scale uncorrelated 0-1 KP instances.

Ins	Num	Opt	Method	Best	Worst	Mean	Median	Std	Time
KP01	800	40686	CSGHS	40342	40056	40182	40190	68.87	≤ 8
			BMBO	40232	39765	40035	40036	105.8	≤ 8
			CMBO	**40686**	40683	40683	40683	0.71	≤ 8
			OMBO	**40686**	40683	40684	40683	0.86	≤ 8
			LBSA	**40686**	**40684**	**40684.9**	**40685**	**0.18**	0.15
KP02	1000	50592	CSGHS	50027	49717	49846	49835	84.36	≤ 8
			BMBO	50024	49336	49699	49689	135.3	≤ 8
			CMBO	**50592**	50590	50590	50590	0.49	≤ 8
			OMBO	**50592**	50590	50590	50590	0.70	≤ 8
			LBSA	**50592**	**50591**	**50591.98**	**50592**	**0.04**	0.18
KP03	1200	61845	CSGHS	60951	60616	60788	60791	79.79	≤ 8
			BMBO	61109	60214	60677	60660	165.8	≤ 8
			CMBO	61845	61840	61841	61840	1.38	≤ 8
			OMBO	61845	61840	61842	61843	1.82	≤ 8
			LBSA	**61846**	**61845**	**61845.27**	**61845**	**0.40**	0.23
KP04	1500	77033	CSGHS	75889	75452	75639	75631	112.3	≤ 10
			BMBO	75761	75062	75464	75482	193.3	≤ 10
			CMBO	**77033**	77031	77031	77031	**0.31**	≤ 10
			OMBO	**77033**	77031	77031	77031	0.56	≤ 10
			LBSA	**77033**	**77032**	**77032.76**	**77033**	0.37	0.28
KP05	2000	102316	CSGHS	-	-	-	-	-	-
			BMBO	-	-	-	-	-	-
			CMBO	**102316**	102313	102314	102313	0.93	≤ 10
			OMBO	**102316**	102313	102314	102313	1.11	≤ 10
			LBSA	**102316**	**102315**	**102315.9**	**102316**	**0.20**	0.38

TABLE 5. Performance comparison on large-scale weakly-correlated 0-1 KP instances.

Ins	Num	Opt	Method	Best	Worst	Mean	Median	Std	Time
KP06	800	35069	CSGHS	34850	34795	34824	34825	14.00	≤ 8
			BMBO	34860	34681	34786	34784	35.01	≤ 8
			CMBO	**35069**	35064	35067	35067	1.45	≤ 8
			OMBO	**35069**	35064	35067	35068	1.47	≤ 8
			LBSA	**35069**	**35068**	**35068.99**	**35069**	**0.02**	0.16
KP07	1000	43786	CSGHS	43484	43386	43440	43442	22.39	≤ 8
			BMBO	43491	43359	43412	43413	31.36	≤ 8
			CMBO	**43786**	43781	43784	43784	1.34	≤ 8
			OMBO	**43786**	43782	43785	43785	1.03	≤ 8
			LBSA	**43786**	**43785**	**43785.97**	**43786**	**0.06**	0.20
KP08	1200	53552	CSGHS	52711	52354	52556	52565	76.89	≤ 8
			BMBO	52774	52110	52425	52390	158.2	≤ 8
			CMBO	53552	**53552**	53552	**53552**	**0.00**	≤ 8
			OMBO	53552	**53552**	53552	**53552**	1.82	≤ 8
			LBSA	**53553**	**53552**	**53552.03**	**53552**	0.06	0.21
KP09	1500	65710	CSGHS	65116	64980	65045	65044	38.14	≤ 10
			BMBO	65123	64916	65022	65012	56.38	≤ 10
			CMBO	**65710**	65708	**65709**	65708	0.58	≤ 10
			OMBO	**65710**	65708	**65709**	**65709**	0.52	≤ 10
			LBSA	65709	**65709**	**65709**	**65709**	**0.00**	0.29
KP10	2000	108200	CSGHS	-	-	-	-	-	-
			BMBO	-	-	-	-	-	-
			CMBO	**108200**	**108200**	**108200**	**108200**	**0.00**	≤ 10
			OMBO	**108200**	**108200**	**108200**	**108200**	**0.00**	≤ 10
			LBSA	**108200**	**108200**	**108200**	**108200**	**0.00**	0.36

of other metaheuristics is 8 seconds for the instances with number of items less than 1500, and 10 seconds for other instances.

Table 4 is the performance comparison on five large-scale uncorrelated 0-1 KP instances. Table 4 shows that CMBO, OMBO, and LBSA have better performance than CSGHS and BMBO. In terms of best value, there is no significant difference among CMBO, OMBO, and LBSA. LBSA found new best solution on KP03 instance. In terms of worst solution, mean solution, and median solution, LBSA outperforms CMBO and OMBO on all five instances.

Table 5 is the performance comparison on five large-scale weakly-correlated 0-1 KP instances. Table 5 shows that CMBO, OMBO, and LBSA have better performance than CSGHS and BMBO. In terms of best value, there is no significant difference among CMBO, OMBO, and LBSA. LBSA found new best solution on KP08 instance, but it didn't find the best solution on KP09 instance. In terms of worst

TABLE 6. Performance comparison on large-scale strongly-correlated 0-1 KP instances.

Ins	Num	Opt	Method	Best	Worst	Mean	Median	Std	Time
KP11	800	40167	CSGHS	40147	40126	40132	40130	5.54	≤ 8
			BMBO	40127	40107	40116	40117	4.52	≤ 8
			CMBO	**40167**	40166	**40167**	**40167**	0.14	≤ 8
			OMBO	**40167**	**40167**	**40167**	**40167**	**0.00**	≤ 8
			LBSA	**40167**	**40167**	**40167**	**40167**	**0.00**	0.15
KP12	1000	49443	CSGHS	49403	49383	49393	49393	6.52	≤ 8
			BMBO	49393	49353	49378	49382	10.12	≤ 8
			CMBO	49433	49433	49422	49433	2.49	≤ 8
			OMBO	**49443**	49441	**49443**	**49443**	0.34	≤ 8
			LBSA	**49443**	**49443**	**49443**	**49443**	**0.00**	0.18
KP13	1200	60640	CSGHS	60587	60567	60573	60570	5.32	≤ 8
			BMBO	60588	60530	60562	60560	11.98	≤ 8
			CMBO	**60640**	60639	**60640**	**60640**	0.14	≤ 8
			OMBO	**60640**	**60640**	**60640**	**60640**	**0.00**	≤ 8
			LBSA	**60640**	**60640**	**60640**	**60640**	**0.00**	0.22
KP14	1500	74932	CSGHS	74858	74817	74835	74832	9.31	≤ 10
			BMBO	74842	74772	74818	74821	15.80	≤ 10
			CMBO	**74932**	74931	**74932**	**74932**	0.27	≤ 10
			OMBO	**74932**	74931	**74932**	**74932**	0.14	≤ 10
			LBSA	**74932**	**74932**	**74932**	**74932**	**0.00**	0.28
KP15	2000	99683	CSGHS	-	-	-	-	-	-
			BMBO	-	-	-	-	-	-
			CMBO	**99683**	99672	99682	**99683**	2.23	≤ 10
			OMBO	**99683**	99679	**99683**	**99683**	0.58	≤ 10
			LBSA	**99683**	**99683**	**99683**	**99683**	**0.00**	0.37

solution, mean solution, and median solution, LBSA is not worse than CMBO and OMBO on all five instances.

Table 6 is the performance comparison on five large-scale strongly-correlated 0-1 KP instances. Table 6 shows that CMBO, OMBO, and LBSA have better performance than CSGHS and BMBO. Among CMBO, OMBO, and LBSA, only LBSA can always find best solutions on all five instances.

VI. CONCLUSIONS

This paper presented a LBSA algorithm with hybrid greedy repair and optimization operator for the 0-1 KP. In LBSA algorithm, hybrid of density-based and value-based greedy repair and optimization operator is designed to obtain better balance between exploitation and exploration. The performance of LBSA algorithm was extensively tested on small-scale, medium-scale, and large-scale 0-1 KP instances. The performance of LBSA algorithm was compared with several state-of-the-art metaheuristics from the literature. Simulation results reveal that LBSA algorithm is very effective and efficient for the 0-1 KP. A shortage of LBSA algorithm is that it is intrinsically serial. As a result, it is meaningful to study the parallelism implementation of LBSA algorithm for the 0-1 KP. Another meaningful research direction is to study whether LBSA algorithm is still effective and efficient for other variants of KP.

REFERENCES

[1] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *J. Optim. Theory Appl.*, vol. 45, no. 1, pp. 41–51, 1985.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] S.-H. Zhan, J. Lin, Z.-J. Zhang, and Y.-W. Zhong, "List-based simulated annealing algorithm for traveling salesman problem," *Comput. Intell. Neurosci.*, vol. 2016, Feb. 2016, Art. no. 1712630.

[4] Y. Zhong, J. Lin, L. Wang, and H. Zhang, "Discrete comprehensive learning particle swarm optimization algorithm with metropolis acceptance criterion for traveling salesman problem," *Swarm Evol. Comput.*, vol. 42, pp. 77–88, Oct. 2018.

[5] M. Abdel-Basset, D. El-Shahat, and A. K. Sangaiah, "A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem," *Int. J. Mach. Learn.*, pp. 1–20, Oct. 2017, doi: 10.1007/s13042-017-0731-3.

[6] K. K. Bhattacharjee and S. P. Sarmah, "Shuffled frog leaping algorithm and its application to 0/1 knapsack problem," *Appl. Soft Comput.*, vol. 19, pp. 252–263, Jun. 2014.

[7] A. Gherboudj, A. Layeb, and S. Chikhi, "Solving 0–1 knapsack problems by a discrete binary version of cuckoo search algorithm," *Int. J. Bio-Inspired Comput.*, vol. 4, no. 4, pp. 229–236, 2012.

[8] K. K. Bhattacharjee and S. P. Sarmah, "Modified swarm intelligence based techniques for the knapsack problem," *Appl. Intell.*, vol. 46, no. 1, pp. 158–179, 2017.

[9] P. H. Nguyen, D. Wang, and T. K. Truong, "A novel binary social spider algorithm for 0–1 knapsack problem," *Int. J. Innov. Comput. Inf. Control*, vol. 13, no. 6, pp. 2039–2049, 2017.

[10] A. J. Kulkarni and H. Shabir, "Solving 0–1 knapsack problem using cohort intelligence algorithm," *Int. J. Mach. Learn.*, vol. 7, no. 3, pp. 427–441, 2016.

[11] M. Han and S. Liu, "An improved binary chicken swarm optimization algorithm for solving 0–1 knapsack problem," in *Proc. IEEE 13th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2017, pp. 207–210.

[12] Y. Zhou, X. Chen, and G. Zhou, "An improved monkey algorithm for a 0–1 knapsack problem," *Appl. Soft Comput.*, vol. 38, pp. 817–830, Jan. 2016.

[13] Y. Gao, F. Zhang, Y. Zhao, and C. Li, "Quantum-inspired wolf pack algorithm to solve the 0–1 knapsack problem," *Math. Problems Eng.*, vol. 2018, Jun. 2018, Art. no. 5327056.

[14] E. Yassien, R. Masadeh, A. Alzaqebah, and A. Shaheen, "Grey wolf optimization applied to the 0/1 knapsack problem," *Int. J. Comput. Appl.*, vol. 169, no. 5, pp. 11–15, 2017.

[15] R. M. Rizk-Allah and A. E. Hassanien, "New binary bat algorithm for solving 0–1 knapsack problem," *Complex Intell. Syst.*, vol. 4, no. 1, pp. 31–53, 2018.

[16] R. V. Yampolskiy and A. El-Barkouky, "Wisdom of artificial crowds algorithm for solving NP-hard problems," *Int. J. Bio-Inspired Comput.*, vol. 3, no. 6, pp. 358–369, 2011.

[17] H. Wu, Y. Zhou, and Q. Luo, "Hybrid symbiotic organisms search algorithm for solving 0–1 knapsack problem," *Int. J. Bio-Inspired Comput.*, vol. 12, no. 1, pp. 23–53, 2018.
[18] J. Cao, B. Yin, X. Lu, Y. Kang, and X. Chen, "A modified artificial bee colony approach for the 0–1 knapsack problem," *Appl. Intell.*, vol. 48, no. 6, pp. 1582–1595, 2017.
[19] M. Abdel-Basset and Y. Zhou, "An elite opposition-flower pollination algorithm for a 0–1 knapsack problem," *Int. J. Bio-Inspired Comput.*, vol. 11, no. 1, pp. 46–53, 2016.
[20] T. K. Truong, K. Li, and Y. Xu, "Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 1774–1780, 2013.
[21] T. K. Truong, K. Li, Y. Xu, A. Ouyang, and T. T. Nguyen, "Solving 0–1 knapsack problem by artificial chemical reaction optimization algorithm with a greedy strategy," *J. Intell. Fuzzy Syst.*, vol. 28, no. 5, pp. 2179–2186, 2015.
[22] J. Zhao, T. Huang, F. Pang, and Y. Liu, "Genetic algorithm based on greedy strategy in the 0–1 knapsack problem," in *Proc. IEEE 3rd Int. Conf. Genetic Evol. Comput. (WGEC)*, Oct. 2009, pp. 105–107.
[23] A. J. Umbarkar and M. S. Joshi, "0/1 knapsack problem using diversity based dual population genetic algorithm," *Int. J. Intell. Syst. Appl.*, vol. 6, no. 10, pp. 34–40, 2014.
[24] Y. Zhou, Z. Bao, Q. Luo, and S. Zhang, "A complex-valued encoding wind driven optimization for the 0–1 knapsack problem," *Appl. Intell.*, vol. 46, no. 3, pp. 684–702, 2017.
[25] Y. Feng, K. Jia, and Y. He, "An improved hybrid encoding cuckoo search algorithm for 0–1 knapsack problems," *Comput. Intell. Neurosci.*, vol. 2014, Dec. 2014, Art. no. 970456.
[26] Y. Feng, G. Wang, Q. Feng, and X. Zhao, "An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0–1 knapsack problems," *Comput. Intell. Neurosci.*, vol. 2014, Oct. 2014, Art. no. 857254.
[27] Y. Feng, G.-G. Wang, and X.-Z. Gao, "A novel hybrid cuckoo search algorithm with global harmony search for 0–1 knapsack problems," *Int. J. Comput. Intell. Syst.*, vol. 9, no. 6, pp. 1174–1190, 2016.
[28] Y. Feng, G.-G. Wang, S. Deb, M. Lu, and X.-J. Zhao, "Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1619–1634, 2017.
[29] Y. Feng, J. Yang, C. Wu, M. Lu, and X. Zhao, "Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation," *Memetic Comput.*, vol. 10, no. 2, pp. 135–150, 2018.
[30] Y. Feng, G.-G. Wang, J. Dong, and L. Wang, "Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0–1 knapsack problem," *Comput. Elect. Eng.*, vol. 67, pp. 454–468, Apr. 2017.
[31] D. Zou, L. Gao, S. Li, and J. Wu, "Solving 0–1 knapsack problem by a novel global harmony search algorithm," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1556–1564, 2011.
[32] A. Layeb, "A hybrid quantum inspired harmony search algorithm for 0–1 optimization problems," *J. Comput. Appl. Math.*, vol. 253, pp. 14–25, Dec. 2013.
[33] S. Tuo, L. Yong, and F. Deng, "A novel harmony search algorithm based on teaching-learning strategies for 0–1 knapsack problems," *Sci. World J.*, vol. 2014, Jan. 2014, Art. no. 637412.
[34] W.-L. Xiang, M.-Q. An, Y.-Z. Li, R.-C. He, and J.-F. Zhang, "A novel discrete global-best harmony search algorithm for solving 0–1 knapsack problems," *Discrete Dyn. Nature Soc.*, vol. 2014, Mar. 2014, Art. no. 573731.
[35] X. Kong, L. Gao, H. Ouyang, and S. Li, "A simplified binary harmony search algorithm for large scale 0–1 knapsack problems," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5337–5355, 2015.

SHI-HUA ZHAN received the B.S. degree in industrial electric automation from Fuzhou University, Fuzhou, China, in 1990. He is currently an Associate Professor with the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou. His current research interests include computational intelligence and network security.

ZE-JUN ZHANG received the B.S. and M.S. degrees in computer science from Guizhou University, Guiyang, China, in 2007 and 2010, respectively, and the Ph.D. degree in electronic engineering from Xidian University, Xi'an, China, in 2014. He is currently a Lecturer with the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, China. His current research interests include pattern recognition and synthetic aperture radar image processing.

LI-JIN WANG received the Ph.D. degree from Beijing Forestry University, Beijing, China, in 2008. He is currently a Professor with the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, China. His current research interests include nature-inspired algorithm and intelligent information processing.

YI-WEN ZHONG received the M.S. and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China, in 2002 and 2005, respectively. He is currently a Professor with the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, China. His current research interests include computational intelligence, data visualization, and bioinformatics.