Adaptive Cuckoo Search Algorithm for the Bin Packing Problem

Chapter · April 2016

DOI: 10.1007/978-3-319-33410-3_8

2 authors:

Zendaoui Zakaria
Université 8 mai 1945 - Guelma
3 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Abdesslem Layeb
Université Constantine 2
71 PUBLICATIONS **1,000** CITATIONS
SEE PROFILE

Adaptive Cuckoo Search Algorithm for the Bin Packing Problem

Zakaria Zendaoui and Abdesslem Layeb

Abstract Bin Packing Problem (BPP) is one of the most difficult NP-hard combinatorial optimization problems. For that, an adaptive version of Cuckoo Search (CS) is used to deal with this problem. This algorithm has proved to be effective in solving many optimization problems. The idea of the adaptive CS (ACS) is based on integer permutations based levy flight and a decoding mechanism to obtain discrete solutions. The ranked order value (ROV) rule is the key to any passage from a continuous space to a combinatorial one. The experimental results show that ACS can be superior to some metaheuristics for a number of BPP instances.

Keywords Combinatorial optimization · Cuckoo search · First fit algorithm · Bin packing problem

1 Introduction

The combinatorial optimization plays very important role in operational research, discrete mathematics and computer science. The objective of this field is to solve several combinatorial optimization problems that are NP-hard because the number of combinations increases exponentially with the size of the problem. Consequently, searching for every possible combination is computationally expansive and not practical. Bin packing problem (BPP) is known to be NP-Hard optimization problem. There are three main variants of BPP: one, two and three dimensional BPP. They have several real applications such as container loading, cutting stock, packaging design and resource allocation, etc. In this paper, we deal with the

Z. Zendaoui (✉) · A. Layeb
MISC Laboratory, Department of Computer Science and Its Applications,
University of Constantine 2, Constantine, Algeria
e-mail: zakaria.zendaoui@gmail.com

A. Layeb
e-mail: Layeb.univ@gmail.com

S. Chikhi et al. (eds.), *Modelling and Implementation of Complex Systems*, Lecture Notes in Networks and Systems 1,
DOI 10.1007/978-3-319-33410-3_8

one-dimensional Bin Packing Problem (BPP-1) [1–3]. As NP-hard optimization problem, not all existing algorithms are able to find the optimum solution within a reasonable time. In fact, several approximate methods have been proposed to solve this problem, which are generally based on heuristics or metaheuristics.

Metaheuristic algorithms are among the efficient optimization tools used for solving hard optimization problems. Metaheuristics present two important advantages which are simplicity and flexibility [4, 5]. In recent years, Most of meta-heuristic algorithms are nature inspired, mimicking the successful features of the underlying biological, physical, or sociological systems. Not all algorithms perform equally well, some may obtain better results than others for a given problem, and there is no universally efficient algorithm to deal with all problems. So many challenges remain, especially for solving tough, NP-hard optimization problems [6].

Many heuristics have been used to solve the BPP like the First Fit algorithm (FF), the Best Fit algorithm (BF), etc. [7–9]. Moreover, many kinds of meta-heuristics have been used too like genetic algorithms [10], Ant colony [11], etc.

CS is an optimization algorithm developed by Yang and Deb in 2009 [12]. It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). The cuckoo's behavior and the mechanism of Lévy flights [13, 14] have leading to design of an efficient inspired algorithm performing optimization search [15, 16]. In this paper, we present an adaptive CS algorithm, called ACS, by combining CS and decoding mechanism for solving bin packing problem. In the ACS, Ranked-Order-Value (ROV) technique is employed to convert continuous solutions into discrete ones. Results show that ACS obtains better performance than four other meta-heuristic algorithms on the bin packing problem.

This paper is organized as follows: Sect. 2 briefly introduces the CS. Section 3 present briefly the BPP. Section 4 describes the ACS. Section 5 shows the experimental results on a set of benchmarks of BPP-1. Finally, Sect. 6 concludes the paper.

2 Cuckoo Search Algorithm

In the standard CS, a cuckoo searches for a new nest via Lévy flights [17], rather than by simple isotropic random walks. Lévy flights essentially provide a random walk while their random steps are drawn from a Lévy distribution for large steps which has an infinite variance with an infinite mean. Here the consecutive jumps/steps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail [18]. CS is based on the following three idealized rules [19]:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.
- The best nests with high quality of eggs will carry over to the next generations.

- The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0, 1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

In the last rule, the parameter p_a is named switching probability. A fraction p_a of the n host nests are replaced by new nests (with new random solutions at new locations).

The pseudo-code of CS algorithm can be summarized in Algorithm 1 from which can see that the parameter p_a control the balance between local and global explorative random walks.

The local random walk can be written as

$$x_i^{t+1} = x_i^t + \alpha\, s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t) \tag{1}$$

where x_j^t and x_k^t are two different solutions selected randomly, $H(u)$ is a Heaviside function, ε is a random number drawn from a uniform distribution, and s is the step size.

On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda) \tag{2}$$

where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0) \tag{3}$$

here $\alpha > 0$ is the scaling factor, which should be related to the scales of the problem of interest, and $L(s, \lambda)$ determines the characteristic scale.

Algorithm 1 Cuckoo Search via Lévy Flights

```
Objective function f(x), x = (x_1, ..., x_d)^T
Generate initial population of n host nests x_i (i = 1, ..., n)
while (t < MaxGeneration) or (stop criterion) do
  Get a cuckoo (say, i) randomly by Lévy flights
  Evaluate its quality/fitness F_i
  Choose a nest among n (say, j) randomly
  if (F_i > F_j) then
    Replace j by the new solution
  end if
  A fraction (p_a) of worse nests are abandoned
          and new ones/solutions are built/generated
  Keep the best solutions (or nests with quality solu
  tions)
  Rank the solutions and find the current best;
end while
```

3 The Bin Packing Problem (BPP)

The BPP-1 [8, 20] consists to pack a set of items having different weights into a minimum number of bins which may have also different capacities. Among the most popular heuristics used to solve the bin packing problem, the First Fit algorithm (FF) which places each item into the first bin in which it will fit. The second popular heuristic algorithm is the Best Fit (BF) which puts each element into the filled bin in which it fits. Moreover, the FF and BF heuristics can be improved by applying a specific order of items like in First Fit Decreasing (FFD) and Best Fit Decreasing (BFD), etc. [9, 21].

Formally, the bin packing problem can be stated as follows:

$$Min\, z(y) = \sum_{j=1}^{n} y_j \tag{4}$$

Subject to constraints:

$$\sum_{i=1}^{n} w_i x_{ij} \leq C_{y_j}, j = 1..n \tag{5}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1..n \tag{6}$$

where,

$$y_j = \begin{cases} 1, & \text{if bin j is used} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$x_{ij} = \begin{cases} 1, & \text{if item i is placed in bin j} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

In the above model, the objective minimizes the total number of bins used to pack all items. We assumed here that all the bins have the same capacity C. The first constraint guarantees that the weights of items (w_i) filled in the bin j do not exceed the bin capacity. The second constraint ensures that each item is placed only in one bin.

It appears to be impossible to obtain exact solutions in polynomial time. The main reason is that the required computation grows exponentially with the size of the problem. Therefore, it is often required to find near optimal solutions to these problems in reasonable time. Efficient heuristic algorithms offer a good alternative to accomplish this goal. Within this perspective, we are interested in applying a CS to solve this problem.

4 Solving the BPP with CS

Now, the cuckoo search adaptation for solving the BPP is presented in detail.

4.1 BPP Solution Representation

The CS algorithm was designed for continuous optimization problems, while BPP is a discrete problem. Thus, standard CS cannot be directly adopted for the BPP. So, the most important issue to apply CS for the BPP is to find a suitable relationship between the real number solutions and the item permutations. For this issue, different techniques have been proposed, such as the largest ranked value (LRV) [22], the smallest position value (SPV) [23], the largest order value (LOV) [24], and ranked order value (ROV) [25]. By sorting the position values of a continuous solution in different order, different item permutations are obtained. Figure 1 presents an example of the four rules. It can be seen from Fig. 1 that ROV rule give the closest integer solution to continuous solution.

In this paper, the ROV rule is used [25]. The ROV is a simple method based on random key representation [26] with guarantees feasibility of new solutions without creating additional overhead.

We assume that a cuckoo lays a single egg in one nest, and each nest contains only one egg. This egg is a solution represented by an item permutation, and we use a decoding mechanism that reveals the actual assignment of the items to bins corresponding to this solution. Since the FF and BF methods are easily implemented, we have adapted them in our algorithm as a decoding mechanism.

For the sake of clarity, let's consider an instance of BBP-1. Table 1 contains 6 items to pack, numbered 1 through 6. The capacity of each bin is equal to 8.

To generate a BPP solution from a permutation of the items, we can take the items one by one in the order given by the permutation, and assigning them to the available bin according to the FF or BF method. In the current example, we use FF

Continuous solution	3.29	1.00	4.59	5.00	2.00	5.10
LRV	4	6	3	2	5	1
SPV	2	5	1	3	4	6
LOV	6	4	3	1	5	2
ROV	**3**	**1**	**4**	**5**	**2**	**6**

Fig. 1 An example for the four rules

Table 1 An instance of BBP

Items	1	2	3	4	5	6
w_i	2	2	2	3	3	4

Step 1: Permutation of numbers	4	3	1	5	6	2
Step 2: FF method	4	3	1	5	6	2
	bin 1			bin 2		bin 3

Fig. 2 Procedure of generation a random initial solution

method, so the result is a partition of items, with indicates that the item i is packed in the bin j as showed in Fig. 2.

4.2 Objective Function

The objective function is an important parameter in any optimization algorithm. Indeed, a good objective function helps the search process to find the optimal solution. In the context of the bin packing problem, using the number of bins as objective function can make the algorithm suffer from stagnation because there can be several arrangements which have the same number of bins. Though, this information will be better if it is integrated with other information like the fullness of the bins. That's why we have used the objective function defined by Falkenauer and Delchambre in [27], which contains both item's weight and bin capacities information. This objective function is given by the Eq. (9)

$$Max f = (sum_i/C)^k/nbin \tag{9}$$

where,

sum_i is the sum of all weight items packed in the bin i;
C is the capacity of bin;
$nbin$ is the number of used bins;

k is the parameter that defines equilibrium of the filling bins. By increasing k, we give a higher fitness to solutions that contain a mix of well-filled and less well-filled bins, rather than equally filled bins.

4.3 Moving in the Search Space

In this ACS to solve BPP, we have two moves that generate a new solution from an existing solution by changing the order of assigned items. The first move is the local random move (Fig. 3) that is given by the Eq. (1). The second move is the global random move that is carried out by using Lévy flights (Fig. 4) given by the Eq. (2). Both move give real values, so, the smallest value of a solution is firstly picked and assigned a rank value 1. Then, the second smallest position value will be assigned rank value 2. In a similar way, the new solution can be obtained.

Existing solution	4	**1**	6	**5**	**2**	3
Result of Eq. (1)	3.29	1.00	4.59	5.00	2.00	5.10
New solution	3	**1**	4	**5**	**2**	6

Fig. 3 Local random move

Existing solution	4	1	6	5	2	3
Result of Eq. (2)	3.74	5.41	5.63	5.08	2.46	2.98
New solution	3	5	6	4	1	2

Fig. 4 Global random move

Algorithm 2: Adaptive Cuckoo Search

Objective function $f(x), x = (x_1, \ldots, x_d)^T$
Generate initial population of n host nests $x_i\ (i = 1, \ldots, n)$
while (t < MaxGeneration) do
  X_i = Generate new solution by Lévy flights;
  P_i = Convert X_i to an item permutation based on ROV rule;
  Replace X_i by P_i;
  F_i = Evaluate P_i;
  P_j = Randomly choose solution from population;
  if $(F_i > F_j)$ then
    Replace P_j the new solution P_i;
  end if
  A fraction (p_a) of worse solutions are abandoned and new ones are generated according Eq. 1;
  Convert the new solutions to item permutations;
  Replace the new solutions by the item permutations;
  Evaluate their fitness;
  Rank the solutions and find the current best
end while

The pseudo-code of ACS algorithm can be seen in Algorithm 2.

5 Experimental Results

In order to test our approach and show the performance, The ACS to BPP is implemented and tested on some instances (benchmarks) of BPP-1 taken from the site http://www.wiwi.uni-jena.de/Entscheidung/bin-pp/. The benchmark data sets are divided into three classes: easy, medium and hard class instances. We have implemented our approach in Matlab 8.3 under the 32 bits Seven Operating System. Experiments are concluded on a laptop with the configurations of Intel(R) CoreTM 2 Duo 2.00 GHz, and 2 GB of RAM.

The parameters used in the experiments are shown in Table 2. The selected parameters are those values that gave the best results for both the solution quality and the computational time. Table 3 shows that the maximum number of iterations can be set to 200.

The obtained results are summarized in Tables 4, 5 and 6, where the first column is the name of the instance. The column 'Best Known' contains the best known results. The column 'FFD' contains the results of the first fit decreasing heuristic. The column 'AS' contains the results of the Ant System algorithm [28]. The column 'FCO' contains the results of the firefly algorithm based ant colony optimization [28]. The column 'QICS' contains the results of the quantum cuckoo search algorithm [29], and the column 'ACS' contains the results of our approach. Moreover, we have used the Friedman test for comparing statistically the obtained results.

Table 2 Parameter settings

Parameter	Value	Meaning
n	20	Population size
p_a	0.2	Portion of bad solutions
$N\_IterTotal$	200	Maximum number of iterations
α	0.01	Step size
Lb	1	Lower bounds
Ub		Number of items

Table 3 Average of best solutions of 10 runs for HARD0, N4W2B1R0 and N4C1W4_A

Instance	Best known	Iteration				
		1	2	90	185	500
HARD0	56	59	59	58	58	58
N4W2B1R0	101	107	106	106	105	105
N4C1W4_A	368	368	368	368	368	368

Table 4 Results of algorithms for the easy class

Instance	Best known	FFD	AS	FCO	QICS	ACS
N1C1W1_A	25	**25**	**25**	**25**	**25**	**25**
N1C1W1_D	28	**28**	**28**	**28**	**28**	**28**
N1C1W1_G	25	**25**	**25**	**25**	**25**	**25**
N1C1W1_B	31	**31**	**31**	**31**	**31**	**31**
N1C1W1_E	26	**26**	**26**	**26**	**26**	**26**
N1C1W1_F	27	**27**	**27**	**27**	**27**	**27**
N1C1W1_I	25	**25**	**25**	**25**	**25**	**25**
N2C1W2_P	68	**68**	**68**	**68**	**68**	**68**
N2C1W2_N	64	**64**	**64**	**64**	**64**	**64**
N2C1W2_O	64	**64**	**64**	**64**	**64**	**64**
N2C1W2_R	67	**67**	**67**	**67**	**67**	**67**
N4C1W2_T	323	**323**	**323**	**323**	**323**	**323**
N4C1W4_C	365	**365**	**365**	**365**	**365**	**365**
N4C1W4_A	368	**368**	**368**	**368**	**368**	**368**
N4C1W4_D	359	**359**	**359**	**359**	**359**	**359**
N4C1W4_B	349	**349**	**349**	**349**	**349**	**349**

Table 5 Results of algorithms for the medium class

Instance	Best known	FFD	AS	FCO	QICS	ACS
N1W1B2R1	17	18	18	**17**	**17**	**17**
N1W1B1R9	17	19	**17**	**17**	18	**17**
N1W1B1R2	19	20	20	**19**	20	**19**
N1W1B2R0	17	18	18	18	18	**17**
N1W1B2R3	16	17	17	17	17	17
N2W1B1R0	34	37	37	35	36	**34**
N2W1B1R3	34	38	36	36	37	35
N2W1B1R1	34	37	36	36	37	35
N2W1B1R4	34	37	35	35	37	**34**
N2W3B3R7	13	**13**	**13**	**13**	**13**	**13**
N2W4B1R0	12	**12**	**12**	**12**	**12**	**12**
N4W2B1R0	101	109	107	106	109	105
N4W2B1R3	100	109	106	105	108	104
N4W3B3R7	74	**74**	**74**	**74**	**74**	**74**
N4W4B1R0	56	58	58	57	58	57
N4W4B1R1	56	58	58	58	58	57

Table 6 Results of algorithms for the hard class

Instance	Best known	FFD	AS	FCO	QICS	ACS
HARD0	56	59	59	59	59	58
HARD1	57	60	60	59	60	59
HARD2	56	60	60	59	60	59
HARD3	55	59	59	59	59	58
HARD4	57	60	60	60	60	59
HARD5	56	59	59	59	59	58
HARD6	57	60	60	59	59	59
HARD7	55	59	59	58	59	58
HARD8	57	60	60	59	59	59
HARD9	56	60	59	59	59	59

In Table 4, the experimental results of the first series of easy instances are given. From this table we see that the results obtained by FFD, AS, FCO, QICS, and ACS are completely identical to the best know solutions.

In Table 5, the experimental results of the second series of medium instances are given. It can be seen from this table and Fig. 5 that ACS outperforms the other

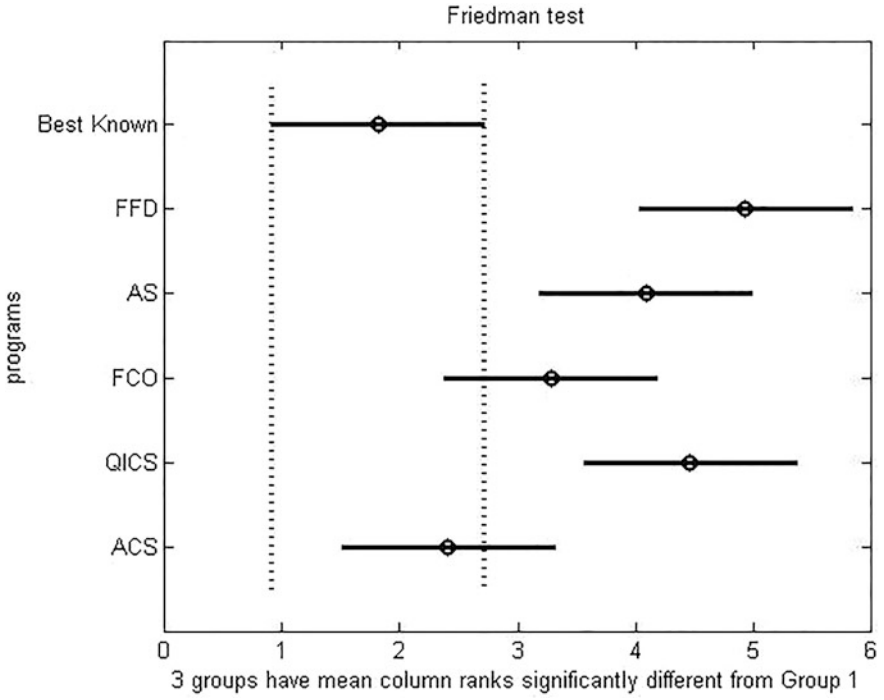

Fig. 5 Friedman test for medium instances

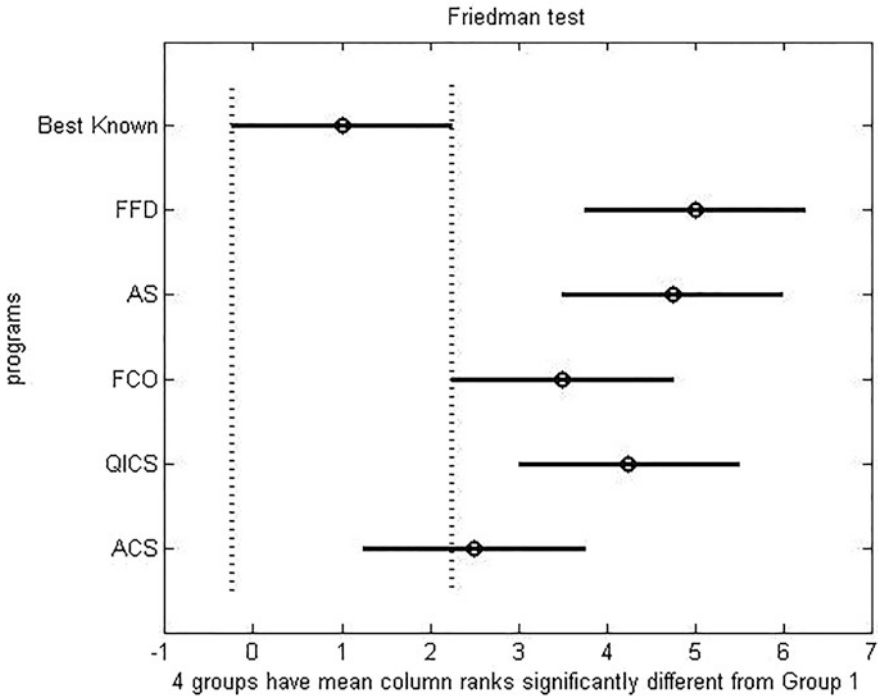

Fig. 6 Friedman test for hard instances

algorithms in solving all the sixteen tested instances. The ACS obtains nine best known solutions while FFD/AS/FCO/QICS only obtains three/four/six/four best known solutions among sixteen instances.

In Table 6, the experimental results of the third series of hard instances are given. It can be seen from this table and Fig. 6 that ACS outperforms the other algorithms in solving all the ten tested instances. The ACS is better, in term of solution quality, than FFD/AS/FCO/QICS in ten/nine/four/seven instances.

As it is well seen from the results presented in Fig. 7 of Friedman test for the three types of instances (Tables 4, 5 and 6), our ACS succeeds in finding the near optimal solutions compared to the other methods.

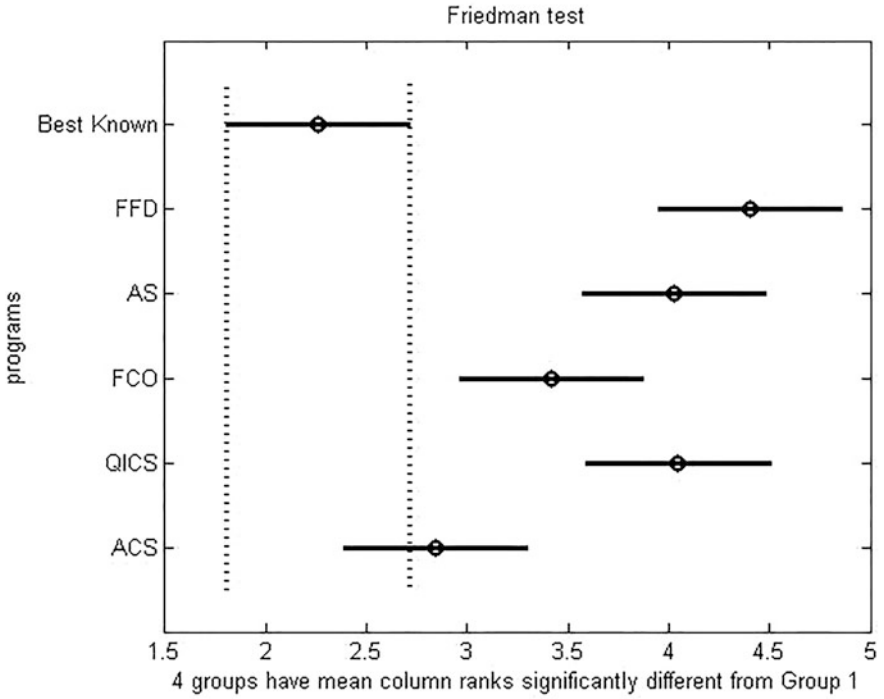

Fig. 7 Friedman test for all instances

6 Conclusion

In this paper, we have adapted Cuckoo Search algorithm to solve the one-dimensional bin packing problem (BPP-1). In our Adaptive Cuckoo Search (ACS) to BPP, the ROV technique is used to transform continuous solutions into discrete integer permutations. ACS has been implemented and its performance has been tested on three classes benchmark BPP-1 instances. Its performance has been compared with the FFD, AS, FCO and QICS algorithms. The results of the comparison have shown that the ACS outperforms all four methods for solving BPP-1 instances. ACS has few numbers of parameters and thus it is easier to implement and to tune these parameters. Furthermore, further studies can be fruitful when focusing on the improvement of ACS by introducing local search strategy, and the use of parallelism techniques.

References

1. Martello, S., Toth, P.: Bin-packing problem. In: Knapsack Problems: Algorithms and Computer Implementations (8), pp. 221–245. Wiley (1990)
2. Coffman, E.G. Jr., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-Hard Problems, pp. 46–93. PWS Publishing, Boston (1996)
3. Fleszar, K., Hindi, K.S.: New heuristics for one-dimensional bin-packing. Comput. Oper. Res. **29**(7), 821–839 (2002)
4. Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H.: Metaheuristic Applications in Structures and Infrastructures. Newnes (2013)
5. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, Bristol (2010)
6. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. Evolut. Comput. IEEE Trans. **1**(1), 67–82 (1997)
7. lvim, A.C.F., Ribeiro, C.C., Glover, F., Aloise, D.J.: A hybrid improvement heuristic for the one-dimensional bin packing problem. J. Heuristics **10**, 205–229 (2004)
8. Kao, C.-Y., Lin, F.-T.: A stochastic approach for the one-dimensional bin-packing problems. Syst. Man Cybern. **2**, 1545–1551 (1992)
9. Scholl, A., Klein, R., Juergens, C.: Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Comput. Oper. Res. **24**(7), 627–645 (1997)
10. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. J. Heuristics **2**, 5–30 (1996)
11. Wang, S., Shi, R., Wang, L., Ge, M.: Study on improved ant colony optimization for bin-packing problem. In: International Conference on Computer Desingn and Application (4), pp. 489–491 (2010)
12. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: World Congress on Nature and Biologically Inspired Computing. NaBIC 2009, pp. 210–214. IEEE, New York (2009)
13. Payne, R.B., Sorenson, M.D., Klitz, K.: The Cuckoos. Oxford University Press (2005)
14. Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing. J. Comput. Phys. **226**, 1830–1844 (2007)
15. Tein, L.H., Ramli, R.: Recent advancements of nurse scheduling models and a potential path. In: Proceedings 6th IMT-GT Conference on Mathematics, Statistics and its Applications (ICMSA 2010), pp. 395–409 (2010)
16. Dhivya, M.: Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach (CBPA). Int. J. Commun. Netw. Syst. Sci. **4**(4), 249–255 (2011)
17. Shlesinger, M.F., Zaslavsky, G.M., Frisch, U.: Lévy flights and related topics in physics. In: Levy Flights and Related Topics in Physics, vol. 450 (1995)
18. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
19. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms, pp. 105–107, 2nd edn. Luniver Press (2010)
20. Alvim, A.C., Ribeiro, C.C., Glover, F., Aloise, D.J.: A hybrid improvement heuristic for the one-dimensional bin packing problem. J. Heuristics **10**(2), 205–229 (2004)
21. Monaci, M.: Algorithms for packing and scheduling problems. Q. J. Belg. Fr. Ital. Oper. Res. Soc. **1**(1), 85–87 (2003)
22. Liang, J., Pan, Q.K., Tiejun, C., Wang, L.: Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. Int. J. Adv. Manuf. Technol. **55**(5–8), 755–762 (2011)
23. Tasgetiren, M.F., Liang, Y.C., Sevkli, M., Gencyilmaz, G.: Particle swarm optimization and differential evolution for the singlemachine total weighted tardiness problem. Int. J. Prod. Res. **44**(22), 4737–4754 (2006)

24. Qian, B., Wang, L., Rong, H., Wang, W.L., Huang, D.X., Wang, X.: A hybrid differential evolution method for permutation flow-shop scheduling. Int. J. Adv. Manuf. Technol. **38**(7–8), 757–777 (2008)
25. Liu, B., Wang, L., Qian, B., Jin, Y.H.: Hybrid Particle Swarm Optimization for Stochastic Flow Shop Scheduling with No-wait Constraint. International Federation of Automatic Control, Seoul (2008)
26. Bean., J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comput. **6**, 154–160 (1994)
27. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: Proceedings of the IEEE 1992 International Conference on Robotics and Automation, Nice, France (May 1992)
28. Layeb, A., Benayad, Z.: A novel firefly algorithm based ant colony optimization for solving combinatorial optimization problems. Int. J. Comput. Sci. Appl. **11**(2), 19–37 (2014)
29. Layeb, A., Boussalia, S.R.: A novel quantum inspired cuckoo search algorithm for bin packing problem. Int. J. Inf. Technol. Comput. Sci. (IJITCS) **4**(5), 58 (2012)