Decision Science Letters 9 (2020) 581–594

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

# Harmony search algorithm with adaptive parameter setting for solving large bin packing problems

**Amol C. Adamuthe[a*] and Tushar Nitave[b]**

[a]*Department of Computer Science & Information Technology, Rajarambapu Institute of Technology, India*
[b]*Department of Computer Science, Illinois Institute of Technology, Chicago, United States*

**C H R O N I C L E**

*Article history*:
Received January 14, 2020
Received in revised format:
March 23, 2020
Accepted June 5 2020
Available online
June 5, 2020

*Keywords*:
*Harmony search algorithm*
*Bin packing problem*
*Combinatorial optimization*
*Constraint satisfaction problem*
*Heuristics*

**A B S T R A C T**

Bin packing problem is a constrained optimization problem with a huge search space due to large combinations. Bin packing problem has a wide range of applications in multiple fields. This paper presents harmony search algorithm with different initialization and adaptive PAR strategies for solving bin packing problem. The proposed Harmony search (HS) variations tests two partial feasible initialization strategies for bin packing problem. The paper presents adaptive PAR strategies for better exploration and exploitation of HS algorithm. The PAR values are tuned in every iteration. Improved initialization strategy, population initialization after premature convergence and adaptive PAR leads to the better exploration of harmony search algorithm for bin packing problem. The performance of variations are tested over 120 benchmark instances with 100 and 200 objects with varying complexities. The results show that improved HS performs better than basic HS with respect to best, mean, convergence rate. The performance of algorithms is tested with varying harmony memory size and harmony memory considering rate. Results show that variation in these two parameter values has less effect on performance of improved versions.

© 2020 by the authors; licensee Growing Science, Canada.

## 1. Introduction

Combinatorial optimization problems are difficult to solve due to huge search space and constraints. Bin packing problem (BPP) is one of the important combinatorial optimization problems. Bin packing problem has applications is multiple domains such as transportation (Perboli et al., 2014), logistics (Aggoun et al., 2016), resource placement (Song et al., 2013; Adamuthe & Patil, 2018). The BPP problem formulation is well known and presented in many papers (Fleszar & Hindi, 2002; Schoenfield, 2002; Alvim et al., 2004; Abdel-Basset et al., 2018). In one-dimensional bin packing problem, items with varying weights are to be packed using homogeneous/heterogeneous bins. Consider a set of '$k$' bins $Bin_1, Bin_2 \ldots Bin_k$ with same or different capacity and '$n$' objects with different weights $W_1, W_2 \ldots W_n$. The objective is to pack all the items into the bins in such a way that a minimum number of bins are required. A solution is said to be optimal if it uses a minimal number of bins.

$$\min \ \sum_{i=1}^{k} z_i \tag{1}$$

subject to

$$\sum_{j=1}^{n} W_j y_{ij} \ \leq \ B_i \quad \text{for all i} \tag{2}$$

$y_{ij}$ = 1 if object $j$ is put in bin $i$, zero, otherwise

$z_i$ =1 if bin $i$ is used, 0, otherwise $\qquad i \in k \text{ and } j \in n$

* Corresponding author.
E-mail address: amol.admuthe@gmail.com amol.adamuthe@ritindia.edu (A. C. Adamuthe)

© 2020 by the authors; licensee Growing Science, Canada.
doi: 10.5267/j.dsl.2020.6.001

In literature, bin packing problem is investigated with different deterministic and heuristic algorithms. First-Fit (FF) and Best-Fit (BF) algorithms have an absolute worst-case ratio of 7/4 for the BPP (Simchi-Levi, 1994). Worst-case performance ratio of first fit algorithm (Dósa & Sgall, 2013) and best fit algorithm is 17/10 (Dósa & Sgall, 2014). A large investigation is done to solve optimization problems using approximation algorithms. Approximation algorithms are investigated when polynomial-time of an algorithm is expensive. A linear approximation algorithm was presented in (Berghammer & Reuter, 2003) for BPP with absolute approximation factor 3/2. Martel (1985) presented a linear off-line algorithm for bin packing with performance ratio of 4/3. In literature, different heuristic algorithms are experimented to solve bin packing problem such as genetic algorithms (Falkenauer, 1996), cuckoo search (Layeb & Boussalia, 2012; Zendaoui & Layeb, 2016), firefly algorithm (Layeb & Benayad, 2014), ant colony optimization (Levine & Ducatelle, 2004), Lévy-based whale optimization algorithm (Abdel-Basset et al., 2018).

In 2001, Geem et al. (2001) presented a new population based heuristic algorithm called as Harmony Search (HS) algorithm. The improvisation technique used by HS algorithm is similar to that of music players. HS is analogous to GA and PSO. In recent years harmony search algorithm is investigated to solve different optimization problems (Lee & Geem, 2005; Mahdavi et al., 2007; Kattan & Abdullah, 2013; Valian et al., 2014). From 2001, various modification have been proposed by researchers to improve efficiency of basic HS algorithm. These modified HS algorithms are better than basic HS in terms of convergence, optimization capability and solution accuracy for applied problem instances. An opposition-based initialization to accelerate HS algorithm was proposed by Banerjee et al. (2014) and results conclude robustness of this approach. Taherinejad (2009) presented modified harmony search (MHS) with pitch-adjustment step in such a way that harmony is affected by the best harmony in the HM. Different investigation are,

- Pitch adjustment rule (Omran & Mahdavi, 2008)
- Balanced intensification and diversification with suitable pitch adjustment rules (Yadav et al., 2012)
- Adaptive parameter setting (Mahdavi et al., 2007; Wang & Huang, 2010; Khalili et al., 2014; Park et al., 2017)

This paper presents harmony search algorithm for solving bin packing problem. The basic harmony search algorithm shows good convergence towards the optimal solution. However, stuck in performing local search in the high performance region in the search space. Paper presents partially feasible initialization, re-initialization strategies and adaptive PAR for bin packing problem. Paper presents two strategies for changing PAR values

- Increasing PAR presented by (Mahdavi et al., 2007)
- Decreasing PAR

Performance of basic HS and two variations are presented using 120 benchmark instances of bin packing problem with 100 and 200 objects. Experiments are conducted to check the effect of Harmony Memory Size (HMS) and Harmony Memory Considering Rate (HMCR).

In next section proposed harmony search algorithm for solving bin packing problem is presented. Section 3 is about dataset, results and performance evaluations. At last, conclusions is stated in section 4.

## 2. Harmony search for Bin packing problem

Harmony search is inspired from musicians' behaviour. Musicians' try to improvise pitches of their instruments. The fundamental steps of harmony search algorithm are (Geem et al., 2001):

Step 1. Harmony search algorithm parameters initialization
Step 2. Harmony memory initialization
Step 3. Improvise harmony
Step 4. Update the harmony memory
Step 5. Repeat steps 3 and 4 until the termination condition is reached

This section presents optimizing bin packing problem with harmony search algorithm. Application of basic steps of harmony search algorithm for bin packing are explained in detail. Bin packing problem is formulated as minimization problem in which we have to minimize the number of bins used. Penalty function is applied if constraint described by equation 3 is violated. The total penalty for a solution is calculated as follows:

$$P = |\sum_{i=1}^{k} p_i| \tag{3}$$

Penalty value of the solution is considered as objective value.

Step 1: This is basic step of HS. Need to initialize the HS parameters namely harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjusting rate (PAR). Paper presents basic and two version of adaptive HS

algorithm. For basic HS algorithm, parameters are initialized randomly. For adaptive PAR HS algorithm the initialization values for PAR are presented below.

Step 2: Representation and initialization of harmony memory (HM) considering the problem specific knowledge.

Real coded solution representation is used to solve the BPP problem. Size of the array depends on the number of objects to be packed in bin. The values in the array indicate the bin number to which the object is assigned. Fig. 1 shows the sample solution with 10 objects and 6 bins. For example, $Obj_5$ is assigned to $Bin_1$. Objects $Obj_2$ and $Obj_5$ are packed in bin $Bin_1$. This representation guarantees the placement of all objects.

| $Obj_1$ | $Obj_2$ | $Obj_3$ | $Obj_4$ | $Obj_5$ | $Obj_6$ | $Obj_7$ | $Obj_8$ | $Obj_9$ | $Obj_{10}$ |
|---|---|---|---|---|---|---|---|---|---|
| $Bin_3$ | $Bin_1$ | $Bin_5$ | $Bin_6$ | $Bin_1$ | $Bin_2$ | $Bin_5$ | $Bin_9$ | $Bin_3$ | $Bin_2$ |

**Fig. 1.** Solution Representation

In HS a matrix known as harmony memory (HM) which stores best solution vectors is considered as core data structure. HM can be represented as a matrix where each row corresponds to solution vector. Sample HM is shown in Fig 2.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $Bin_3$ | $Bin_1$ | $Bin_9$ | $Bin_6$ | $Bin_1$ | $Bin_2$ | $Bin_5$ | $Bin_9$ | $Bin_3$ | $Bin_8$ |
| $Bin_9$ | $Bin_3$ | $Bin_2$ | $Bin_1$ | $Bin_5$ | $Bin_9$ | $Bin_6$ | $Bin_1$ | $Bin_2$ | $Bin_2$ |
| $Bin_3$ | $Bin_1$ | $Bin_5$ | $Bin_6$ | $Bin_1$ | $Bin_2$ | $Bin_5$ | $Bin_9$ | $Bin_3$ | $Bin_8$ |
| $Bin_8$ | $Bin_5$ | $Bin_9$ | $Bin_6$ | $Bin_1$ | $Bin_2$ | $Bin_5$ | $Bin_9$ | $Bin_3$ | $Bin_3$ |
| $Bin_5$ | $Bin_2$ | $Bin_9$ | $Bin_6$ | $Bin_3$ | $Bin_3$ | $Bin_9$ | $Bin_1$ | $Bin_7$ | $Bin_1$ |

**Fig. 2.** Sample harmony memory

This paper presents two problem specific harmony search initialization methods for solving bin packing problem.

HM initialization procedure 1:
In this initialization strategy, we randomly select a bin for all object sequentially. The randomly selected bin is assigned to object if it has capacity to occupy the object i.e. (current_bin_capacity – obj_size) > 0. If the assignment violates the constraint then next fit bin is identified.

HM initialization procedure 2:
In this initialization strategy we first iterate over all the bins ($\forall i.\ 0 \le ii \le num\_bins$) until we find a bin such that bin_size[ii] – obj_size >= 0 and then we assign that bin to current object. If after iterating all bins, we fail to find a bin which can accommodate a object then, we randomly assign a bin to that object.

```
reference_bin[ii][jj] ← bin_cap, ∀ii. ii ≤ HMS ∧ ∀jj. jj ≤ number_of_bins
while ii ≤ HMS do
    while jj ≤ number_of_obj do
        assigned_bin ← rand[0, max_bins]
        if obj_weight[jj] < reference_bin[ii][assigned_bin] then
            HM[ii][jj]← assigned_bin
            reference_bin[ii][jj]← reference_bin[ii][jj] – obj_weight[ii]
        else
            while kk ≤ number_of_bins do
                if obj_weight[jj] < reference_bin[ii][kk] then
                    HM[ii][jj]← kk
                    reference_bin[ii][kk]← reference_bin[ii][kk] – obj_weight[ii]
                fi
            od
        fi
    od
od
```

**Fig. 3.** Pseudocode for harmony memory initialization procedure 1

```
reference_bin[ii][jj] ← bin_cap, ∀ii. ii ≤ HMS ∧ ∀jj. jj ≤ number_of_bins
while ii ≤ HMS do
    while jj ≤ number_of_obj do
        obj_assigned ← False
        while kk ≤ number_of_bins do
            if obj_weight[jj] < reference_bin[ii][assigned_bin] then
                HM[ii][jj]← assigned_bin
                reference_bin[ii][jj]← reference_bin[ii][jj] – obj_weight[ii]
                obj_assigned ← True
                break
            fi
        od
        if obj_assigned not True:
            bin ← randomly assign a bin
            HM[ii][jj] ← bin
            reference_bin[ii][bin]← reference_bin[ii][bin] – obj_weight[ii]
        fi
    od
od
```

**Fig. 4.** Pseudocode for harmony memory initialization procedure 2

Step 3: Improvise a new solution from HM.
Many heuristic algorithms use parameters to control the exploration and exploitation capabilities of algorithm. In this step new solution is generated using two parameters namely harmony memory consideration rate (HMCR) and pitch adjusting rate (PAR). The value of the parameter HMCR and PAR range from 0 to 1. The HMCR is probability of selecting a value from previous HM. Every selected value by the memory consideration is examined is modified considering pitch adjusting rate. HMCR and PAR are important for adjusting the convergence of algorithm and fine tune the results to get the optimized solution. Basic harmony search uses static values in all iterations until stopping criteria is satisfied. This paper presents two ways for adaptive PAR values based on the convergence rate of algorithm. The value of PAR is changed after every iteration depending on HM diversity.

Adaptive PAR strategy 1:

The value of PAR is tuned as per following equation (Mahdavi et al., 2007):

$$PAR = PAR + \left( PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{NI} \times gn \right) \tag{4}$$

*gn* is the current generation (current iteration) and NI represents maximum iterations.
$PAR_{max}$ and $PAR_{min}$ are initialized to 1.0 and 0.45 respectively.

Adaptive PAR strategy 2:

$$PAR = PAR + \left( PAR_{min} - \frac{(PAR_{max} - PAR_{min})}{NI} \times gn \right) \tag{5}$$

$PAR_{max}$ and $PAR_{min}$ are initialized to 1.0 and 0.85 respectively.

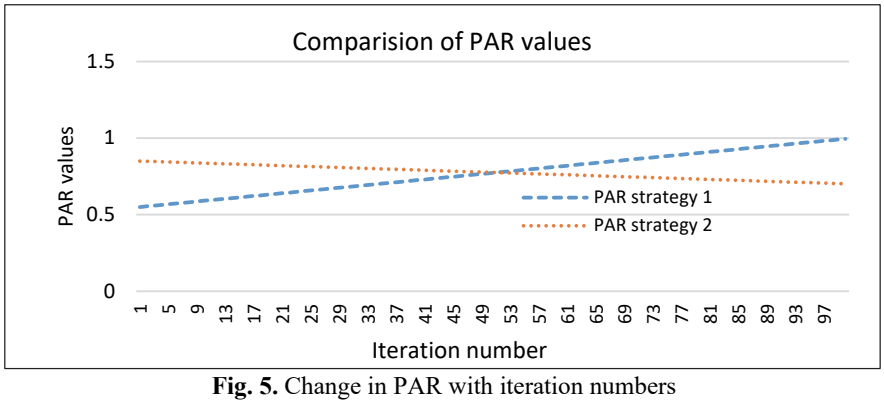



**Fig. 5.** Change in PAR with iteration numbers

Fig. 6 presents pseudocode for basic harmony search algorithm for solving bin packing problem. Fig. 7 presents pseudocode for adaptive harmony search algorithm with dynamic PAR. It covers two versions of adaptive HS algorithm with different PAR initiation and changing strategy.

Basic Algorithm (V1): Harmony Search with static PAR

```
Input: Number of objects with weight, number of bins with capacity
Variables: number_of_bins, bin_cap, HS, number_of_obj, assigned_bin, obj_weight
Output: minimum bins required

Initialize HMS, HMCR, PAR and max iterations
Initialize harmony memory with procedure 1

begin
 while ii ≤ max_iterations do
    while jj ≤ HMS do
       while kk ≤ number_of_objects do
          if rand[0,1] < HMCR then
            memory_consideration(kk)
            if rand[0,1] < PAR then pitch_adjustment(jj, kk)fi
            else
              random_solution(jj, kk)
          fi
       od
       memory_updation(jj)
    od
 od
end
```

**Fig. 6.** Pseudocode Harmony Search with static PAR (V1)

Adaptive Harmony Search with Dynamic PAR

```
Input, Variables, Output: ← same as Algorithm 1

Initialize HMS, HMCR and max iterations
Initialize PAR with strategy 1 (for version 2 – V2)
Initialize PAR with strategy 2 (for version 3 – V3)
Initialize harmony memory with procedure 2

begin
while ii ≤ max_iterations do
  while jj ≤ HMS do
    while kk ≤ number_of_objects do
      if rand[0,1] < HMCR then
        memory_consideration(kk)
        if rand[0,1] < PAR then
          if reference_bin[jj][kk] > bin_cap then
            pitch_adjustment(jj, kk)
          fi
        fi
      else
        random_solution(jj, kk)
      fi
    od
    memory_updation(jj)
  od

  Adaptive PAR strategy 1 (for variation 2- V2)
  Adaptive PAR strategy 2 (for variation 3- V3)

  if current_iteration > threshold then
    re_initialize_HM()
    threshold = threshold + 100
  fi
od
end
```

**Fig. 7.** Pseudocode Adaptive Harmony Search with dynamic PAR (V2 and V3)

## 3. Results and Performance Evaluation of HS

This section presents dataset details, obtained results and discussion. The proposed harmony search algorithm variations for solving bin packing problem are implemented using 'C' programmming language. The programs are executed on machine with Intel core i5-3210M CPU 2.5 GHz and 4 GB RAM. Benchmark instances of bin packing problem are taken to test the performance of proposed variations. Benchmark datasets for bin packing problem taken from https://www2.wiwi.uni-jena.de/Entscheidung/binpp/index.htm. These data sets are prepared as per Martello and Toth (Martello & Toth, 1990)

and best known solutions are available. The naming conventions of the data sets are as follows.

Parameters used,

- o → number of objects
- c_bin → capacity of bin
- $W_j$ → size of object j (j=1,…,o)
- min → minimum number of required bins

The name of the instances are given by NpCqWr_i where,

- p=1 (for o=50), p=2 (for o=100), p=3 (for o=200), p=4 (for o=500)
- q=1 (for c_bin=100), q=2 (for c_bin=120), q=3 (for c_bin=150)
- r=1 (for $W_j$ from [1,100]), r=2 ([20,100]), r=4 ([30,100], for (j=1,…,o)
- i=A...T for the 20 instances of each class

Total 120 instances with p=2 and 3, q=1 and r=1,2, and 4 are used for experimentation. The input file for the following instances consists of number of objects, bin capacity and size of each object separated by new line. The structure of input data sets is as given below.

| Sample input file | |
|---|---|
| 3 | ← number of objects |
| 100 | ← capacity of bin |
| 30 | } size of items |
| 40 | |
| 30 | |

These problem instances are experimented by different authors. The optimal solutions are presented by (Fleszar & Hindi, 2002; Schoenfield, 2002; Alvim et al., 2004; Schwerin & Wäscher, 1997). Paper presents the performance evaluation of proposed variations of harmony search algorithms using obtained best, mean, standard deviation, convergence graph and success rate. The success rate is calculated using equation 6. For every dataset harmony search algorithm was executed 10 times.

$$\textit{Success rate} = \textit{Number of times best obtained / number of times program executed} \quad (6)$$

Table 1 and 2 shows the success rate of experimented three harmony search algorithm for 120 instances of bin packing problem. The variation V1 fails to give the best known results for all the instances with 100 and 200 objects. Results indicate that proposed initialization strategy and HS algorithm are not tuned to the problem for obtaining best results. Results show that the performance of HS version 2 and version 3 is better than the basic HS algorithm. Both the variations provide best-known solution for given instances. Initialization strategy 2, adaptive PAR and re-initialization works well for bin packing problem. The result shows that adaptive PAR harmony search algorithm versions work better than static PAR HS. Harmony search algorithm with increasing PAR (version 2) is found better than decreasing PAR (version 3). The variation 2 gives 100 percent success rate for more than 66 percent instances with 100 objects. Variation V2 gives good success rate to majority of instances with 100 objects. Approximately 1 percent of instances shows less success rate than 50 percent. The 100 percent success rate is drastically reduced to approximately 25 percent for instances with 200 objects. Approximately 10 percent of instances shows less success rate than 50 percent. There is a large variation on the success rate.

The variation 3 gives 100 percent success rate for more than 66 percent instances with 100 objects. Version 3 fails to give an optimal solution to approximately 25% instances with 100 objects. The performance of version 3 reduces drastically for instances with 200 objects. Only 50% instances are solved by version 3.

**Table 1**
Success rate of proposed HS variations for 60 instances with 100 objects

| Dataset with 100 objects | HS variation | | | Dataset with 100 objects | HS variation | | |
|---|---|---|---|---|---|---|---|
| | V3 | V2 | V1 | | V3 | V2 | V1 |
| N2C1W1_A | 100 | 100 | 0 | N2C1W2_K | 100 | 100 | 0 |
| N2C1W1_B | 100 | 100 | 0 | N2C1W2_L | 100 | 50 | 0 |
| N2C1W1_C | 100 | 80 | 0 | N2C1W2_M | 100 | 100 | 0 |
| N2C1W1_D | 100 | 100 | 0 | N2C1W2_N | 100 | 90 | 0 |
| N2C1W1_E | 100 | 100 | 0 | N2C1W2_O | 0 | 100 | 0 |
| N2C1W1_F | 100 | 100 | 0 | N2C1W2_P | 100 | 100 | 0 |
| N2C1W1_G | 100 | 90 | 0 | N2C1W2_Q | 0 | 100 | 0 |
| N2C1W1_H | 100 | 100 | 0 | N2C1W2_R | 100 | 80 | 0 |
| N2C1W1_I | 100 | 50 | 0 | N2C1W2_S | 0 | 100 | 0 |
| N2C1W1_J | 100 | 100 | 0 | N2C1W2_T | 100 | 100 | 0 |
| N2C1W1_K | 0 | 70 | 0 | N2C1W4_A | 100 | 70 | 0 |
| N2C1W1_L | 10 | 100 | 0 | N2C1W4_B | 100 | 100 | 0 |
| N2C1W1_M | 0 | 100 | 0 | N2C1W4_C | 0 | 100 | 0 |
| N2C1W1_N | 40 | 90 | 0 | N2C1W4_D | 100 | 40 | 0 |
| N2C1W1_O | 100 | 100 | 0 | N2C1W4_E | 100 | 100 | 0 |
| N2C1W1_P | 100 | 100 | 0 | N2C1W4_F | 0 | 100 | 0 |
| N2C1W1_Q | 0 | 100 | 0 | N2C1W4_G | 0 | 80 | 0 |
| N2C1W1_R | 0 | 80 | 0 | N2C1W4_H | 60 | 100 | 0 |
| N2C1W1_S | 100 | 100 | 0 | N2C1W4_I | 100 | 100 | 0 |
| N2C1W1_T | 100 | 100 | 0 | N2C1W4_J | 100 | 90 | 0 |
| N2C1W2_A | 100 | 100 | 0 | N2C1W4_K | 0 | 100 | 0 |
| N2C1W2_B | 0 | 100 | 0 | N2C1W4_L | 100 | 100 | 0 |
| N2C1W2_C | 100 | 90 | 0 | N2C1W4_M | 100 | 80 | 0 |
| N2C1W2_D | 100 | 100 | 0 | N2C1W4_N | 100 | 50 | 0 |
| N2C1W2_E | 0 | 100 | 0 | N2C1W4_O | 100 | 100 | 0 |
| N2C1W2_F | 100 | 80 | 0 | N2C1W4_P | 80 | 70 | 0 |
| N2C1W2_G | 100 | 100 | 0 | N2C1W4_Q | 0 | 100 | 0 |
| N2C1W2_H | 100 | 100 | 0 | N2C1W4_R | 100 | 100 | 0 |
| N2C1W2_I | 0 | 100 | 0 | N2C1W4_S | 100 | 100 | 0 |
| N2C1W2_J | 100 | 60 | 0 | N2C1W4_T | 100 | 100 | 0 |

**Table 2**
Success rate of proposed HS variations for 60 instances with 200 objects

| Dataset with 200 objects | HS variation | | | Dataset with 200 objects | HS variation | | |
|---|---|---|---|---|---|---|---|
| | V3 | V2 | V1 | | V3 | V2 | V1 |
| N3C1W1_A | 0 | 80 | 0 | N3C1W2_K | 0 | 100 | 0 |
| N3C1W1_B | 0 | 60 | 0 | N3C1W2_L | 0 | 50 | 0 |
| N3C1W1_C | 100 | 70 | 0 | N3C1W2_M | 40 | 100 | 0 |
| N3C1W1_D | 0 | 80 | 0 | N3C1W2_N | 0 | 90 | 0 |
| N3C1W1_E | 0 | 50 | 0 | N3C1W2_O | 0 | 80 | 0 |
| N3C1W1_F | 0 | 40 | 0 | N3C1W2_P | 0 | 100 | 0 |
| N3C1W1_G | 100 | 90 | 0 | N3C1W2_Q | 20 | 60 | 0 |
| N3C1W1_H | 0 | 70 | 0 | N3C1W2_R | 0 | 50 | 0 |
| N3C1W1_I | 0 | 80 | 0 | N3C1W2_S | 100 | 20 | 0 |
| N3C1W1_J | 100 | 30 | 0 | N3C1W2_T | 0 | 50 | 0 |
| N3C1W1_K | 0 | 70 | 0 | N3C1W4_A | 0 | 70 | 0 |
| N3C1W1_L | 0 | 100 | 0 | N3C1W4_B | 0 | 100 | 0 |
| N3C1W1_M | 100 | 70 | 0 | N3C1W4_C | 0 | 100 | 0 |
| N3C1W1_N | 0 | 90 | 0 | N3C1W4_D | 0 | 40 | 0 |
| N3C1W1_O | 0 | 100 | 0 | N3C1W4_E | 0 | 50 | 0 |
| N3C1W1_P | 0 | 50 | 0 | N3C1W4_F | 0 | 80 | 0 |
| N3C1W1_Q | 0 | 100 | 0 | N3C1W4_G | 0 | 80 | 0 |
| N3C1W1_R | 0 | 80 | 0 | N3C1W4_H | 0 | 60 | 0 |
| N3C1W1_S | 0 | 40 | 0 | N3C1W4_I | 0 | 60 | 0 |
| N3C1W1_T | 0 | 80 | 0 | N3C1W4_J | 0 | 80 | 0 |
| N3C1W2_A | 0 | 40 | 0 | N3C1W4_K | 100 | 70 | 0 |
| N3C1W2_B | 100 | 100 | 0 | N3C1W4_L | 20 | 60 | 0 |
| N3C1W2_C | 0 | 90 | 0 | N3C1W4_M | 0 | 80 | 0 |
| N3C1W2_D | 100 | 100 | 0 | N3C1W4_N | 0 | 50 | 0 |
| N3C1W2_E | 0 | 100 | 0 | N3C1W4_O | 0 | 100 | 0 |
| N3C1W2_F | 0 | 80 | 0 | N3C1W4_P | 0 | 70 | 0 |
| N3C1W2_G | 0 | 100 | 0 | N3C1W4_Q | 0 | 100 | 0 |
| N3C1W2_H | 0 | 70 | 0 | N3C1W4_R | 100 | 30 | 0 |
| N3C1W2_I | 0 | 100 | 0 | N3C1W4_S | 0 | 100 | 0 |
| N3C1W2_J | 0 | 60 | 0 | N3C1W4_T | 0 | 90 | 0 |

Table 3 and 4 show the obtained best, mean and standard deviation on the objective value of experimented three harmony search algorithm variations for 120 instances of bin packing problem. The basic harmony search algorithm fails to give best known solution for all the instances. The negative values show the penalty applied for constraint violations. Zero value indicates no violation of constraints. HS variations 2 and 3 shows best performance for more than 95% instances of bin packing problem with 100 objects. The performance of variation 2 and 3 are approximately same.

**Table 3**
Performance of proposed HS variations for 60 datasets with 100 objects

| Dataset | Minimal bins | | HS Variation V3 | V2 | V1 | Dataset | Minimal bins | | HS Variation V3 | V2 | V1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N2C1W1_A | 48 | Best | 0 | 0 | -285 | N2C1W2_K | 72 | Best | 0 | 0 | -313 |
| | | SD | 0 | 0 | 13.56 | | | SD | 0 | 0 | 14.25 |
| | | Mean | 0 | 0 | -367.66 | | | Mean | 0 | 0 | -398.36 |
| N2C1W1_B | 49 | Best | 0 | 0 | -233 | N2C1W2_L | 62 | Best | 0 | 0 | -269 |
| | | SD | 0 | 0 | 18.19 | | | SD | 0 | 0 | 15.52 |
| | | Mean | 0 | 0 | -342.45 | | | Mean | 0 | 0 | -366 |
| N2C1W1_C | 46 | Best | 0 | 0 | -293 | N2C1W2_M | 65 | Best | 0 | 0 | -293 |
| | | SD | 0 | 0 | 13.26 | | | SD | 0 | 0 | 13.26 |
| | | Mean | 0 | 0 | -360.45 | | | Mean | 0 | 0 | -360.56 |
| N2C1W1_D | 50 | Best | 0 | 0 | -290 | N2C1W2_N | 64 | Best | 0 | 0 | -288 |
| | | SD | 0 | 0 | 18.02 | | | SD | 0 | 0 | 20.92 |
| | | Mean | 0 | 0 | -370.98 | | | Mean | 0 | 0 | -374.38 |
| N2C1W1_E | 58 | Best | 0 | 0 | -318 | N2C1W2_O | 64 | Best | -1 | 0 | -368 |
| | | SD | 0 | 0 | 18.9 | | | SD | 2.08 | 0 | 14.56 |
| | | Mean | 0 | 0 | -380.1 | | | Mean | -4 | 0 | -426.36 |
| N2C1W1_F | 60 | Best | 0 | 0 | -328 | N2C1W2_P | 68 | Best | 0 | 0 | -384 |
| | | SD | 0 | 0 | 15.42 | | | SD | 0 | 0 | 21.59 |
| | | Mean | 0 | 0 | -384.86 | | | Mean | 0 | 0 | -498.36 |
| N2C1W1_G | 60 | Best | 0 | 0 | -277 | N2C1W2_Q | 65 | Best | -2 | 0 | -288 |
| | | SD | 0 | 0 | 20.92 | | | SD | 1.01 | 0 | 20.92 |
| | | Mean | 0 | 0 | -374.36 | | | Mean | -3 | 0 | -374.36 |
| N2C1W1_H | 52 | Best | 0 | 0 | -369 | N2C1W2_R | 67 | Best | 0 | 0 | -532 |
| | | SD | 0 | 0 | 14.86 | | | SD | 0 | 0 | 21.26 |
| | | Mean | 0 | 0 | -426.32 | | | Mean | 0 | 0 | -609.05 |
| N2C1W1_I | 62 | Best | 0 | 0 | -3842 | N2C1W2_S | 66 | Best | -1 | 0 | -325 |
| | | SD | 0 | 0 | 21.49 | | | SD | 0 | 0 | 15.36 |
| | | Mean | 0 | 0 | -467.24 | | | Mean | -1 | 0 | -385.36 |
| N2C1W1_J | 59 | Best | 0 | 0 | -308 | N2C1W2_T | 66 | Best | 0 | 0 | -318 |
| | | SD | 0 | 0 | 21.72 | | | SD | 0 | 0 | 18.9 |
| | | Mean | 0 | 0 | -417.16 | | | Mean | 0 | 0 | -380.1 |
| N2C1W1_K | 55 | Best | -1 | 0 | -349 | N2C1W4_A | 73 | Best | 0 | 0 | -290 |
| | | SD | 0.29 | 0 | 20.9 | | | SD | 0 | 0 | 18.02 |
| | | Mean | -1 | 0 | -450.29 | | | Mean | 0 | 0 | -370.98 |
| N2C1W1_L | 55 | Best | 0 | 0 | -278 | N2C1W4_B | 71 | Best | 0 | 0 | -285 |
| | | SD | 0 | 0 | 20.54 | | | SD | 0 | 0 | 13.56 |
| | | Mean | 0 | 0 | -390.78 | | | Mean | 0 | 0 | -367.66 |
| N2C1W1_M | 48 | Best | -15 | 0 | -313 | N2C1W4_C | 77 | Best | -1 | 0 | -285 |
| | | SD | 0.62 | 0 | 14.45 | | | SD | 1.10 | 0 | 13.56 |
| | | Mean | -17 | 0 | -378.27 | | | Mean | -2 | 0 | -367.66 |
| N2C1W1_N | 48 | Best | 0 | 0 | -289 | N2C1W4_D | 82 | Best | 0 | 0 | -233 |
| | | SD | 0 | 0 | 15.36 | | | SD | 0 | 0 | 18.25 |
| | | Mean | 0 | 0 | -366.61 | | | Mean | 0 | 0 | -342.56 |
| N2C1W1_O | 54 | Best | 0 | 0 | -268 | N2C1W4_E | 83 | Best | 0 | 0 | -349 |
| | | SD | 0 | 0 | 15.42 | | | SD | 0 | 0 | 20.9 |
| | | Mean | 0 | 0 | -366 | | | Mean | 0 | 0 | -450.29 |
| N2C1W1_P | 54 | Best | 0 | 0 | -293 | N2C1W4_F | 77 | Best | -4 | 0 | -278 |
| | | SD | 0 | 0 | 13.26 | | | SD | 0.65 | 0 | 20.54 |
| | | Mean | 0 | 0 | -360.45 | | | Mean | -4 | 0 | -390.78 |
| N2C1W1_Q | 46 | Best | -22 | 0 | -315 | N2C1W4_G | 71 | Best | -1 | 0 | -326 |
| | | SD | 2.03 | 0 | 14.14 | | | SD | 0.67 | 0 | 14.256 |
| | | Mean | -28 | 0 | -379.81 | | | Mean | -1 | 0 | -398.36 |
| N2C1W1_R | 45 | Best | -1 | 0 | -233 | N2C1W4_H | 75 | Best | 0 | 0 | -268 |
| | | SD | 0.20 | 0 | 18.19 | | | SD | 0.77 | 0 | 15.25 |
| | | Mean | -1 | 0 | -342.45 | | | Mean | -1 | 0 | -312 |
| N2C1W1_S | 45 | Best | 0 | 0 | -349 | N2C1W4_I | 73 | Best | 0 | 0 | -289 |
| | | SD | 0 | 0 | 20.9 | | | SD | 0 | 0 | 12.36 |
| | | Mean | 0 | 0 | -450.29 | | | Mean | 0 | 0 | -378 |
| N2C1W1_T | 52 | Best | 0 | 0 | -313 | N2C1W4_J | 74 | Best | 0 | 0 | -313 |
| | | SD | 0 | 0 | 14.45 | | | SD | 0 | 0 | 14.256 |
| | | Mean | 0 | 0 | -378.27 | | | Mean | 0 | 0 | -369.36 |

**Table 3**
Performance of proposed HS variations for 60 datasets with 100 objects (continued)

| Dataset | Minimal bins | | HS Variation V3 | V2 | V1 | Dataset | Minimal bins | | HS Variation V3 | V2 | V1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N2C1W2_A | 64 | Best | 0 | 0 | -278 | N2C1W4_K | 70 | Best | -11 | 0 | -349 |
| | | SD | 0 | 0 | 20.36 | | | SD | 3.80 | 0 | 20.9 |
| | | Mean | 0 | 0 | -398.78 | | | Mean | -19 | 0 | -450.29 |
| N2C1W2_B | 61 | Best | -1 | 0 | -369 | N2C1W4_L | 75 | Best | 0 | 0 | -323 |
| | | SD | 1.37 | 0 | 14.85 | | | SD | 0 | 0 | 14.25 |
| | | Mean | -3 | 0 | -426.36 | | | Mean | 0 | 0 | -378.26 |
| N2C1W2_C | 68 | Best | 0 | 0 | -328 | N2C1W4_M | 72 | Best | 0 | 0 | -307 |
| | | SD | 0 | 0 | 15.46 | | | SD | 0 | 0 | 21.25 |
| | | Mean | 0 | 0 | -384.56 | | | Mean | 0 | 0 | -416.36 |
| N2C1W2_D | 74 | Best | 0 | 0 | -277 | N2C1W4_N | 71 | Best | 0 | 0 | -298 |
| | | SD | 0 | 0 | 20.98 | | | SD | 0 | 0 | 13.25 |
| | | Mean | 0 | 0 | -374.38 | | | Mean | 0 | 0 | -369.36 |
| N2C1W2_E | 65 | Best | -12 | 0 | -532 | N2C1W4_O | 80 | Best | 0 | 0 | -289 |
| | | SD | 3.22 | 0 | 21.06 | | | SD | 0 | 0 | 13.366 |
| | | Mean | -22 | 0 | -609.36 | | | Mean | 0 | 0 | -369.36 |
| N2C1W2_F | 65 | Best | 0 | 0 | -308 | N2C1W4_P | 67 | Best | 0 | 0 | -295 |
| | | SD | 0 | 0 | 21.72 | | | SD | 0 | 0 | 13.56 |
| | | Mean | 0 | 0 | -417.16 | | | Mean | 0 | 0 | -369.66 |
| N2C1W2_G | 73 | Best | 0 | 0 | -349 | N2C1W4_Q | 75 | Best | -1 | 0 | -287 |
| | | SD | 0 | 0 | 20.9 | | | SD | 0 | 0 | 20.54 |
| | | Mean | 0 | 0 | -450.29 | | | Mean | -1 | 0 | -396.36 |
| N2C1W2_H | 70 | Best | 0 | 0 | -289 | N2C1W4_R | 70 | Best | 0 | 0 | -268 |
| | | SD | 0 | 0 | 15.36 | | | SD | 2.36 | 0 | 15.25 |
| | | Mean | 0 | 0 | -366.61 | | | Mean | -2 | 0 | -369 |
| N2C1W2_I | 67 | Best | -2 | 0 | -313 | N2C1W4_S | 80 | Best | 0 | 0 | -308 |
| | | SD | 0.07 | 0 | 14.45 | | | SD | 0 | 0 | 21.72 |
| | | Mean | -2 | 0 | -378.26 | | | Mean | 0 | 0 | -417.26 |
| N2C1W2_J | 67 | Best | 0 | 0 | -313 | N2C1W4_T | 70 | Best | 0 | 0 | -349 |
| | | SD | 0 | 0 | 28.46 | | | SD | 0 | 0 | 20.9 |
| | | Mean | 0 | 0 | -379.26 | | | Mean | 0 | 0 | -450.29 |

Results from table 4 show that the performance of all three algorithms reduced with an increase in problem complexity with rise in problem size. Best, mean and standard deviations of all three algorithms reduced for instances with 200 objects. For few instances the performance of variation 2 and 3 are similar. There are many instances where variation 2 is found better than variation 3.

Table 4
Best, mean and standard deviation values of proposed HS variations for 60 datasets with 200 objects

| Dataset | Minimal bins | | HS Variation V3 | V2 | V1 | Dataset | Minimal bins | | HS Variation V3 | V2 | V1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N3C1W1_A | 105 | Best | -34 | -2 | -1374 | N3C1W2_K | 120 | Best | -43 | -14 | -1392 |
| | | SD | 17.58 | 49.84 | 22.58 | | | SD | 18.30 | 43.37 | 32.26 |
| | | Mean | -84 | -41 | -1353 | | | Mean | -93 | -41 | -1494.85 |
| N3C1W1_B | 114 | Best | -9 | 0 | -1523 | N3C1W2_L | 136 | Best | -4 | 0 | -1523 |
| | | SD | 20.66 | 46.92 | 35.79 | | | SD | 18.78 | 45.69 | 35.79 |
| | | Mean | -65 | -12.49 | -1667.69 | | | Mean | -49 | -12.59 | -1667.69 |
| N3C1W1_C | 99 | Best | 0 | 0 | -1213 | N3C1W2_M | 136 | Best | 0 | 0 | -1404 |
| | | SD | 0 | 56.71 | 30.93 | | | SD | 22.01 | 46.06 | 34.4 |
| | | Mean | 0 | -17.51 | -1354.46 | | | Mean | -48 | -17.69 | -1567.06 |
| N3C1W1_D | 108 | Best | -3 | 0 | -1404 | N3C1W2_N | 136 | Best | -18 | 0 | -1428 |
| | | SD | 16.14 | 57.83 | 34.4 | | | SD | 30.03 | 46.92 | 26.4 |
| | | Mean | -47 | -19.04 | -1567.06 | | | Mean | -105 | -19.04 | -1534.86 |
| N3C1W1_E | 98 | Best | -3 | 0 | -1297 | N3C1W2_O | 127 | Best | -6 | 0 | -1374 |
| | | SD | 6.46 | 70.4 | 29.74 | | | SD | 17.89 | 49.84 | 28.46 |
| | | Mean | -15 | -28.75 | -1392.08 | | | Mean | -52 | -28.75 | -1471.27 |
| N3C1W1_F | 113 | Best | -2 | 0 | -1443 | N3C1W2_P | 126 | Best | -1 | 0 | -1235 |
| | | SD | 14.03 | 66.12 | 28.16 | | | SD | 20.03 | 54.12 | 32.63 |
| | | Mean | -22 | -21.84 | -1535.08 | | | Mean | -45 | -21.65 | -1397.12 |
| N3C1W1_G | 111 | Best | 0 | 0 | -1392 | N3C1W2_Q | 135 | Best | 0 | 0 | -1320 |
| | | SD | 0 | 43.37 | 32.26 | | | SD | 23.08 | 56.71 | 32.35 |
| | | Mean | 0 | -10.76 | -1494.85 | | | Mean | -48 | -1025 | -1448.88 |
| N3C1W1_H | 104 | Best | -10 | 0 | -1416 | N3C1W2_R | 123 | Best | -52 | -8 | -1689 |
| | | SD | 6.67 | 45.69 | 33.33 | | | SD | 22.14 | 57.83 | 37.44 |
| | | Mean | -26 | -12.32 | -1530.17 | | | Mean | -120 | -12.32 | -1867.86 |
| N3C1W1_I | 100 | Best | -1 | 0 | -1310 | N3C1W2_S | 130 | Best | 0 | 0 | -1404 |
| | | SD | 14.52 | 46.05 | 29.35 | | | SD | 12.49 | 66.12 | 34.4 |
| | | Mean | -20 | -10.47 | -1411.53 | | | Mean | -20 | -10.48 | -1567.06 |
| N3C1W1_J | 108 | Best | 0 | 0 | -1374 | N3C1W2_T | 136 | Best | -1 | 0 | -1218 |
| | | SD | 15.24 | 55.172 | 28.46 | | | SD | 14.12 | 72.4 | 30.68 |
| | | Mean | -21 | -15.6 | -1471.27 | | | Mean | -22 | -15.2 | -1256.23 |

**Table 4**

Best, mean and standard deviation values of proposed HS variations for 60 datasets with 200 objects (Continued)

| Dataset | Minimal bins | | HS Variation V3 | HS Variation V2 | HS Variation V1 | Dataset | Minimal bins | | HS Variation V3 | HS Variation V2 | HS Variation V1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N3C1W1_K | 102 | Best | -4 | 0 | -1428 | N3C1W4_A | 149 | Best | -8 | -2 | -1374 |
| | | SD | 11.35 | 52.65 | 26.4 | | | SD | 17.56 | 43.37 | 22.58 |
| | | Mean | -41 | -15.43 | -1534.86 | | | Mean | -51 | -41 | -1455.52 |
| N3C1W1_L | 97 | Best | -50 | 0 | -1235 | N3C1W4_B | 149 | Best | -1 | 0 | -1423 |
| | | SD | 6.91 | 43.37 | 32.63 | | | SD | 14.11 | 45.69 | 33.58 |
| | | Mean | -69 | -10.76 | -1397.12 | | | Mean | -22 | -12.49 | -1557.59 |
| N3C1W1_M | 106 | Best | 0 | 0 | -1320 | N3C1W4_C | 146 | Best | -12 | 0 | -1425 |
| | | SD | 14.38 | 45.69 | 32.35 | | | SD | 23.17 | 46.05 | 32.22 |
| | | Mean | -21 | -12.32 | -1448.88 | | | Mean | -73 | -17.51 | -1597.19 |
| N3C1W1_N | 93 | Best | -38 | 0 | -1191 | N3C1W4_D | 148 | Best | -13 | 0 | -1566 |
| | | SD | 8.35 | 46.05 | 28.35 | | | SD | 20.16 | 46.92 | 28.54 |
| | | Mean | -53 | -10.47 | -1311.8 | | | Mean | -65 | -19.04 | -1452.33 |
| N3C1W1_O | 98 | Best | -79 | -12 | -1311 | N3C1W4_E | 142 | Best | -2 | 0 | -1198 |
| | | SD | 12.77 | 52.65 | 28.54 | | | SD | 11.42 | 49.84 | 25.36 |
| | | Mean | -117 | -15.43 | -1452.22 | | | Mean | -25 | -28.75 | -1422.6 |
| N3C1W1_P | 108 | Best | -10 | 0 | -1432 | N3C1W4_F | 140 | Best | -8 | 0 | -1297 |
| | | SD | 20.53 | 55.172 | 36.22 | | | SD | 14.68 | 55.172 | 29.74 |
| | | Mean | -42 | -15.6 | -1594.19 | | | Mean | -44 | -21.84 | -1392.08 |
| N3C1W1_Q | 100 | Best | -13 | 0 | -1404 | N3C1W4_G | 148 | Best | -10 | 0 | -1215 |
| | | SD | 6.23 | 45.69 | 34.4 | | | SD | 19.16 | 56.71 | 30.93 |
| | | Mean | -36 | -12.32 | -1567.06 | | | Mean | -69 | -10.76 | -1354.26 |
| N3C1W1_R | 99 | Best | -2 | 0 | -1286 | N3C1W4_H | 141 | Best | -4 | 0 | -1678 |
| | | SD | 12.67 | 0 | 28.26 | | | SD | 16.53 | 57.83 | 36.8 |
| | | Mean | -31 | 0 | -1422.7 | | | Mean | -62 | -12.32 | -1886.2 |
| N3C1W1_S | 100 | Best | -27 | 0 | -1269 | N3C1W4_I | 140 | Best | -24 | 0 | -1687 |
| | | SD | 15.52 | 0 | 31.38 | | | SD | 20.29 | 66.12 | 37.55 |
| | | Mean | -69 | 0 | -1425.03 | | | Mean | -86 | -10.47 | -1879.36 |
| N3C1W1_T | 102 | Best | -15 | 0 | -1361 | N3C1W4_J | 142 | Best | -21 | 0 | -1523 |
| | | SD | 13.62 | 0 | 29.24 | | | SD | 20.81 | 70.4 | 35.79 |
| | | Mean | -62 | 0 | -1496.87 | | | Mean | -76 | -15.6 | -1667.59 |
| N3C1W2_A | 125 | Best | -2 | -111 | -1523 | N3C1W4_K | 147 | Best | 0 | 0 | -1143 |
| | | SD | 11.59 | 169.469 | 35.79 | | | SD | 0 | 66.12 | 28.25 |
| | | Mean | -28 | -157.02 | -1667.69 | | | Mean | 0 | -80.58 | -1535.09 |
| N3C1W2_B | 126 | Best | 0 | -113 | -1374 | N3C1W4_L | 148 | Best | 0 | 0 | -1416 |
| | | SD | 12.81 | 196.42 | 22.58 | | | SD | 22.48 | 56.71 | 33.33 |
| | | Mean | -19 | -182.3 | -1466.59 | | | Mean | -45 | -10.76 | -1530.18 |
| N3C1W2_C | 125 | Best | -42 | -98 | -1679 | N3C1W4_M | 149 | Best | -8 | 0 | -1404 |
| | | SD | 14.47 | 162.689 | 26.7 | | | SD | 29.17 | 57.83 | 34.4 |
| | | Mean | -86 | -140.53 | -1886.22 | | | Mean | -98 | -12.32 | -1567.06 |
| N3C1W2_D | 139 | Best | 0 | -111 | -1311 | N3C1W4_N | 148 | Best | -4 | 0 | -1392 |
| | | SD | 15.81 | 169.469 | 28.54 | | | SD | 24.88 | 45.69 | 32.26 |
| | | Mean | -22 | -157.02 | -1452.22 | | | Mean | -61 | -12.49 | -1494.85 |
| N3C1W2_E | 132 | Best | -1 | -1 | -1191 | N3C1W4_O | 143 | Best | -1 | 0 | -1374 |
| | | SD | 18.62 | 163.369 | 28.35 | | | SD | 15.78 | 70.4 | 22.58 |
| | | Mean | -47 | -74.6 | -1311.8 | | | Mean | -34 | -15.6 | -1465.69 |
| N3C1W2_F | 123 | Best | -30 | -37 | -1297 | N3C1W4_P | 143 | Best | -3 | 0 | -1298 |
| | | SD | 19.30 | 217.29 | 29.74 | | | SD | 22.92 | 46.05 | 29.85 |
| | | Mean | -84 | -106.14 | -1392.08 | | | Mean | -46 | -17.51 | -1326.35 |
| N3C1W2_G | 132 | Best | -14 | -30 | -1213 | N3C1W4_Q | 146 | Best | -23 | 0 | -1215 |
| | | SD | 13.85 | 174.79 | 30.93 | | | SD | 18.96 | 46.92 | 30.36 |
| | | Mean | -72 | -87.75 | -1354.46 | | | Mean | -78 | -19.04 | -1365.26 |
| N3C1W2_H | 129 | Best | -39 | -79 | -1416 | N3C1W4_R | 145 | Best | 0 | 0 | -1326.35 |
| | | SD | 16.14 | 165.43 | 33.33 | | | SD | 18.04 | 55.172 | 32.36 |
| | | Mean | -86 | -125.78 | -1530.17 | | | Mean | -22 | -21.84 | -1445.36 |
| N3C1W2_I | 126 | Best | -25 | -48 | -1443 | N3C1W4_S | 145 | Best | -21 | 0 | -1452 |
| | | SD | 13.19 | 164.277 | 28.16 | | | SD | 23.75 | 49.84 | 36.3 |
| | | Mean | -72 | -132.43 | -1535.08 | | | Mean | -75 | -28.75 | -1598.36 |
| N3C1W2_J | 126 | Best | -20 | -1 | -1374 | N3C1W4_T | 146 | Best | -1 | 0 | -1258 |
| | | SD | 14.61 | 192.38 | 22.58 | | | SD | 12.36 | 43.37 | 30.36 |
| | | Mean | -59 | -81.66 | -1466.59 | | | Mean | -21 | -41 | -1258.3 |

Fig. 8 shows the performance of the basic harmony search algorithm for sample datasets with 100 and 200 objects. Results show reduced performance with an increase in number of objects in bin packing problem. The basic HS algorithm is subjected to premature convergence leading to local optimal solution. Fig. 9 shows fast convergence of version 2 algorithm for sample datasets with 100 objects and 200 objects.

The performance of all three algorithms is tested by varying the HMCR values. The HMCR values are varied from 0.1 to 0.9. Fig. 10, 11(a) and 11(b) present the performance of version 1, 2 and 3 for changing HMCR values respectively. Version 1 algorithm shows better convergence with higher values of HMCR. Performance increases with HMCR values. Best performance is achieved with HMCR values 0.9 and 0.8. The basic harmony search (V1) algorithm fails to give optimal solution for bin packing problem instances.

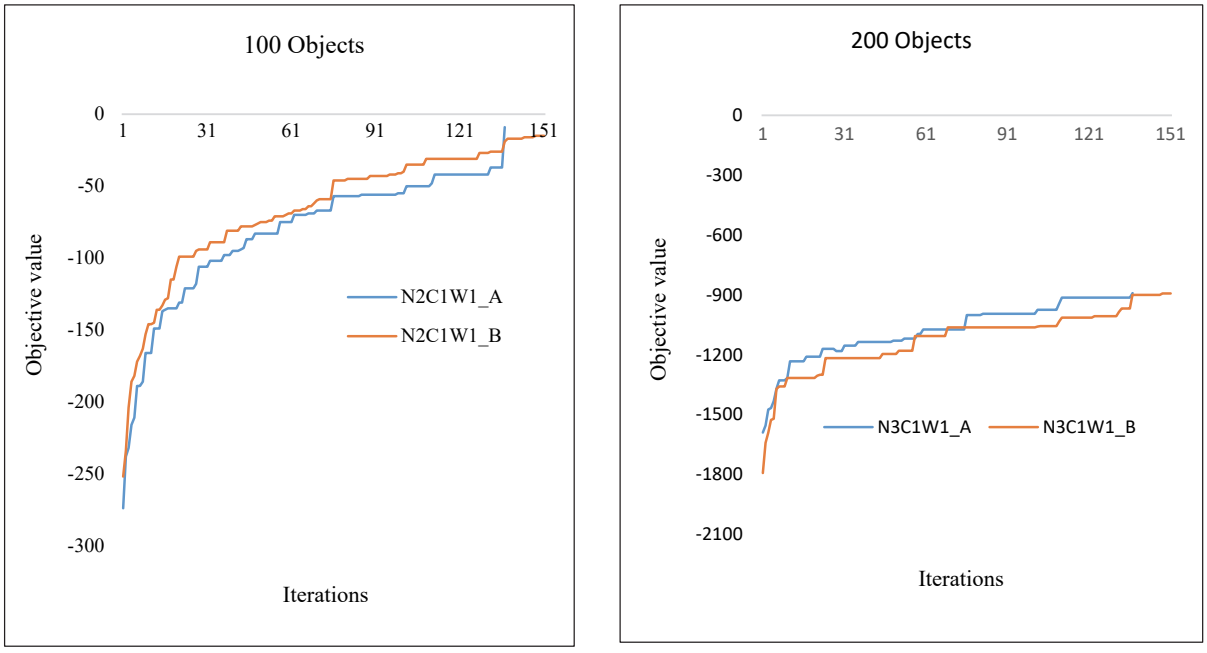



**Fig. 8.** Performance of version 1 algorithm for sample datasets with 100 and 200 objects

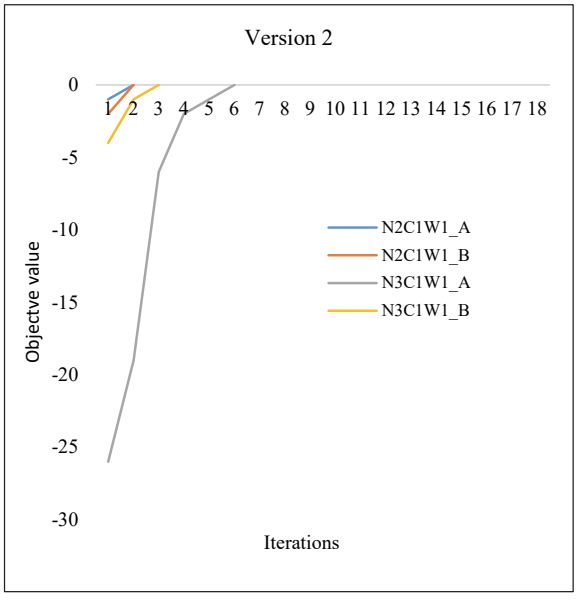



**Fig. 9.** Performance of version 2 algorithm for sample datasets with 100 and 200 objects

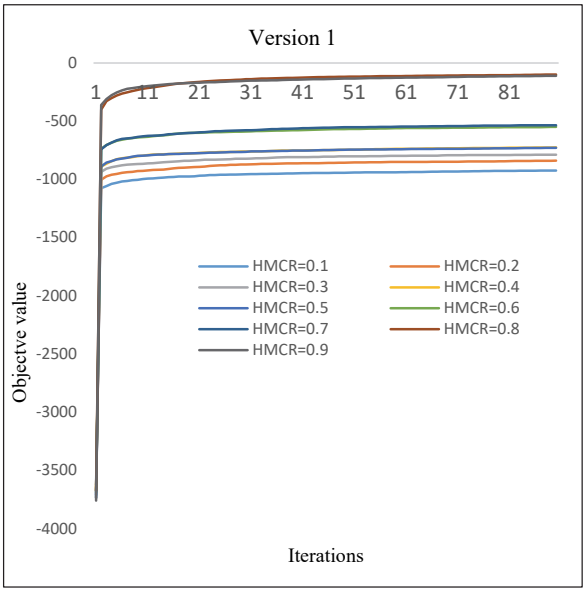
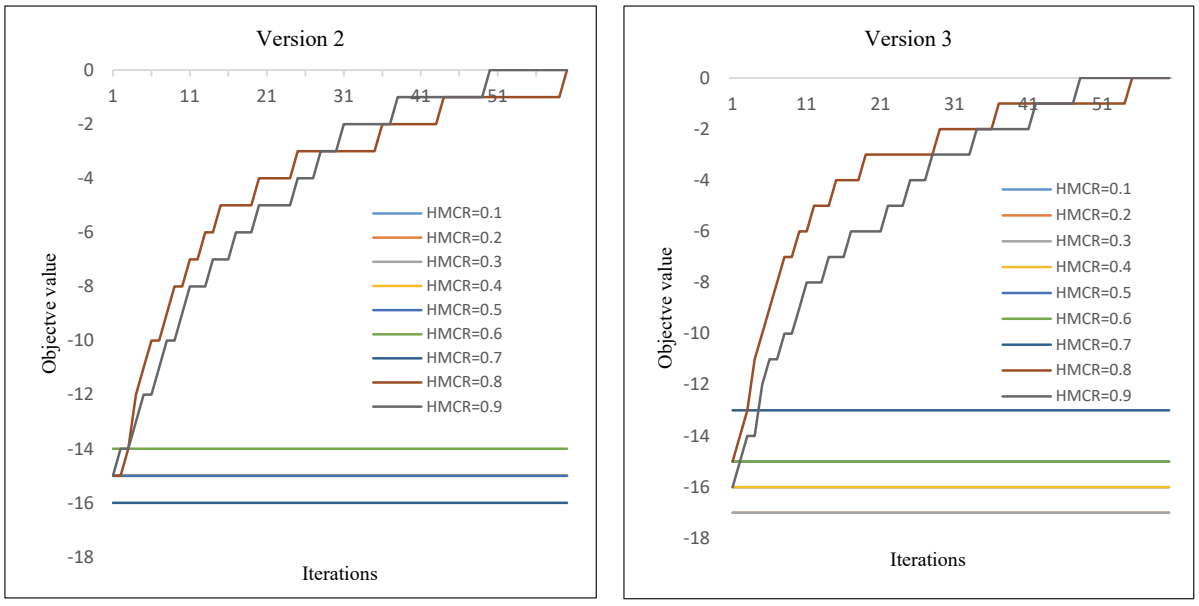



**Fig. 10.** Performance of version 1 algorithm for sample datasets with 100 and 200 objects

The performance of version 2 and version 3 is approximately same with changing HMCR values. Version 2 and 3 shows better convergence towards optimal solution. HMCR values 0.8 and 0.9 shows better performance as compared to other values. Harmony search algorithm with adaptive PAR shows better performance subjected to HMCR values.

Fig. 12 shows the effect of harmony memory size on performance of HS with adaptive PAR applied to solving bin packing problem. Impact of harmony memory size is very small on performance of algorithm.




(a) (b)

**Fig. 11.** Performance of algorithm for sample datasets with 100 and 200 objects (a) version 2 (b) version 3

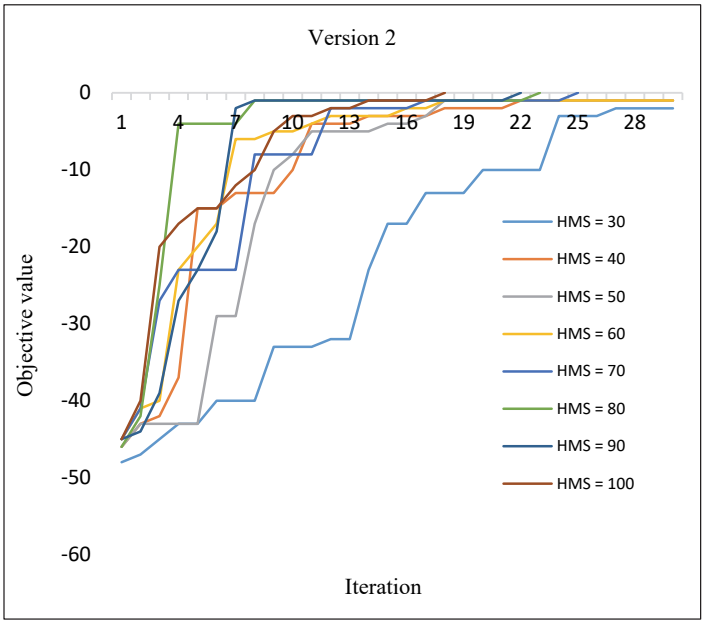



**Fig. 12.** Performance of version 2 algorithm for sample datasets with 100 and 200 objects

## 4. Conclusions

Bin packing problem is an important constraint satisfaction problem with huge combinations. In literature, many deterministic algorithms and heuristics algorithms are investigated to optimize the solutions. This paper presents the application of harmony search algorithm for solving bin packing problem.

Harmony search is a powerful heuristic to solve combinatorial optimization problem. It shows good convergence towards an optimal solution. Experiments show that HS stuck into the best region failing to reach optimal solution. To improve the performance of HS, paper presents two problem specific initialization strategies and re-initialization strategy. Results show that algorithm specific parameters have impact on the performance of HS to reach to optimal solution. Paper investigates two adaptive PAR strategies for fine tuning algorithm to optimize the solutions. The increasing PAR strategy is taken from (Mahdavi et al., 2007). Paper presents the results of basic and two variations of HS algorithm for solving 120 instances of

bin packing problem. Results show that the fine tuning of HS algorithm with adaptive PAR (increasing and decreasing) works better than static PAR. Performance of version 2 (increasing PAR) and version 3 (decreasing PAR) are similar with respect to best, mean and success rate for 60 bin packing problem instances with 100 objects. Version 2 is found better for larger problem instances with 200 objects. The performance of both HS variations are reduced with increase in problem complexity with size. Further experiments are conducted to analyze the impact of HMCR and HMS values of performance of algorithms. High HMCR values show better results. There is scope to improve the strategy for adaptive PAR for solving large instances on bin packing problem with minimal impact of HMCR values.

## References


Abdel-Basset, M., Manogaran, G., Abdel-Fatah, L., & Mirjalili, S. (2018). An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. *Personal and Ubiquitous Computing*, *22*(5-6), 1117-1132.

Adamuthe, A. C., & Patil, J. T. (2018). Differential evolution algorithm for optimizing virtual machine placement problem in cloud computing. *International Journal of Intelligent Systems and Applications*, *11*(7), 58.

Aggoun, A., Rhiat, A., & Fages, F. (2016, May). Panorama of real-life applications in logistics embedding bin packing optimization algorithms, robotics and cloud computing technologies. In *2016 3rd International Conference on Logistics Operations Management (GOL)* (pp. 1-4). IEEE.

Alvim, A. C., Ribeiro, C. C., Glover, F., & Aloise, D. J. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, *10*(2), 205-229.

Banerjee, A., Mukherjee, V., & Ghoshal, S. P. (2014). An opposition-based harmony search algorithm for engineering optimization problems. *Ain Shams Engineering Journal*, *5*(1), 85-101.

Berghammer, R., & Reuter, F. (2003). A linear approximation algorithm for bin packing with absolute approximation factor 32. *Science of Computer Programming*, *48*(1), 67-80.

Dósa, G., & Sgall, J. (2013). First Fit bin packing: A tight analysis. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Dósa, G., & Sgall, J. (2014, July). Optimal analysis of Best Fit bin packing. In *International Colloquium on Automata, Languages, and Programming* (pp. 429-441). Springer, Berlin, Heidelberg.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, *2*(1), 5-30.

Fleszar, K., & Hindi, K. S. (2002). New heuristics for one-dimensional bin-packing. *Computers & operations research*, *29*(7), 821-839.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *simulation*, *76*(2), 60-68.

Kattan, A., & Abdullah, R. (2013). A dynamic self-adaptive harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, *219*(16), 8542-8567.

Khalili, M., Kharrat, R., Salahshoor, K., & Sefat, M. H. (2014). Global dynamic harmony search algorithm: GDHS. *Applied Mathematics and Computation*, *228*, 195-219.

Layeb, A., & Benayad, Z. (2014). A novel firefly algorithm based ant colony optimization for solving combinatorial optimization problems. *IJCSA*, *11*(2), 19-37.

Layeb, A., & Boussalia, S. R. (2012). A novel quantum inspired cuckoo search algorithm for bin packing problem. *International Journal of Information Technology and Computer Science*, *4*(5), 58-67.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, *194*(36-38), 3902-3933.

Levine, J., & Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research society*, *55*(7), 705-716.

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied mathematics and computation*, *188*(2), 1567-1579.

Martel, C. U. (1985). A linear time bin-packing algorithm. *Operations research letters*, *4*(4), 189-192.

Martello, S., & Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete applied mathematics*, *28*(1), 59-70.

Omran, M. G., & Mahdavi, M. (2008). Global-best harmony search. *Applied mathematics and computation*, *198*(2), 643-656.

Park, J., Kwon, S., Kim, M., & Han, S. (2017). A cascaded improved harmony search for line impedance estimation in a radial power system. *IFAC-PapersOnLine*, *50*(1), 3368-3375.

Perboli, G., Gobbato, L., & Perfetti, F. (2014). Packing problems in transportation and supply chain: new problems and trends. *Procedia-Social and Behavioral Sciences*, *111*(0), 672-681.

Schoenfield, J. E. (2002). Fast, exact solution of open bin packing problems without linear programming. Draft. *US Army Space & Missile Defence Command, Huntsville*, *20*, 72.

Schwerin, P., & Wäscher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, *4*(5-6), 377-389.

Simchi-Levi, D. (1994). New worst-case results for the bin-packing problem. *Naval Research Logistics (NRL)*, *41*(4), 579-585.

Song, W., Xiao, Z., Chen, Q., & Luo, H. (2013). Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, *63*(11), 2647-2660.

Taherinejad, N. (2009, August). Highly reliable harmony search algorithm. In *2009 European Conference on Circuit Theory and Design* (pp. 818-822). IEEE.

Valian, E., Tavakoli, S., & Mohanna, S. (2014). An intelligent global harmony search approach to continuous optimization problems. *Applied Mathematics and Computation*, *232*, 670-684.

Wang, C. M., & Huang, Y. F. (2010). Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, *37*(4), 2826-2837.

Yadav, P., Kumar, R., Panda, S. K., & Chang, C. S. (2012). An intelligent tuned harmony search algorithm for optimisation. *Information Sciences*, *196*, 47-72.

Zendaoui, Z., & Layeb, A. (2016). Adaptive cuckoo search algorithm for the bin packing problem. In *Modelling and Implementation of Complex Systems* (pp. 107-120). Springer, Cham.


© 2020 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).