Journal of Computer Engineering
ISSN 10003428

Original Research Article
Vol 9, Issue 2, 2020 | ISSN: 1000-3428

Optimizing Bin Packing with Biologically Inspired Approaches: A Study on Bee Colony Optimization

Dr. Sofia Patel, Dr. Ethan Thompson

Department of Computer Science, University of California, Los Angeles (UCLA); Department of Industrial and Systems Engineering, Georgia Institute of Technology

***Abstract*—**We treat the two-dimensional bin packing problem which involves packing a given set of rectangles into a minimum number of larger identical rectangles called bins. This combinatorial problem is NP-hard. We propose a pretreatment for the oriented version of the problem that allows the valorization of the lost areas in the bins and the reduction of the size problem. A heuristic method based on the strategy first-fit adapted to this problem is presented. We present an approach of resolution by bee colony optimization. Computational results express a comparison of the number of bins used with and without pretreatment.

I. Introduction

THE bin packing problem has many industrial applications, including the cutting of standardized stock units in the wood, steel and glass industries, packing on shelves or truck beds in transportation and warehousing, and the paging of articles in newspapers. Many real problems can be modeled as bin packing problems. However, each real problem has its own specificities. The one-dimensional bin packing problem is strongly NP-hard. This also applies in BPP-2D, since it is a generalization of the one-dimensional version [1] and defined as: Given a set of n rectangular items A = $\{a_1,...,a_n\}$ and a number of larger identical rectangles called bins, the problem is finding the minimum number of bins that must be used to pack this set of items without overlapping.

Many papers deal with BPP-2D and most of them are concerned with the case where the items have a fixed orientation. Lodi et al. [3] have proposed an extensive survey. We quote Berkey and Wang [4], who have presented a heuristic version for the problem. Meta-heuristic procedures have been presented in Lodi et al. [2] and Faroe et al. [5]. Lower bounds have been presented in Carlier et al. [6]. They dominate the bounds introduced by Boschetti and Mingozzi [7], Martello and Vigo [8] and Fekete and Schepers [9].

Few papers deal with the version of BPP-2D where items may be rotated by 90◦. A heuristic method and new lower bounds have been introduced by Boschetti and Mingozzi [10]. Lodi et al. [2] have presented a meta-heuristic algorithm based on a tabu search method. Another tabu search method has been proposed in Harwig and Barnes [11] where items are packed in bottom left stable positions using the method presented in
Chazelle [12].

In this paper, we are interested in the case where the items have a fixed orientation. This version is known as 2BP|O|F according to the classification proposed by Lodi et al. [2], where the field denoted O means that the items are not rotated by 90◦ and the field denoted F means that guillotine packing are not imposed. We consider that each item $a_i = (L_i, l_i)$ in A has a width L_i and a height l_i. We denote as L and l the width and the height of the bin B, B= (L, l).An instance I of a BPP2D|O|F is then defined by the pair (A, B). The minimum number of bins required to pack all the items of an instance I is denoted as OPT (I). L_i is the greatest dimension of item a_i, and l_i the other dimension. The largest and smallest dimensions of the bin are, respectively, L and l.

This paper is organized as follows: In Section II, we develop mathematically a pretreatment with fixed orientation. The strategy used to represent a solution based on a first-fit algorithm adapted to this problem is described in Section III. In Section IV, we are interested in modeling the problem using the bee colony optimization. The implementation is to test the effectiveness of pretreatment; indeed it is to compare the number of bins used on the same instance, with and without pretreatment. The principle of creating the sets of tests is given. A conclusion follows in Section V.

II. Pretreatment

The presented pretreatment is based on the concept of identically feasible function (IFF) defined in Carlier et al. [6] as: Let I = (A, B) be a BPP-2D instance; a function *f* is an IFF associated with instance I iff instance *f* (I) = (A', B) obtained by applying *f* to all items of A is such that OPT (I) = OPT (*f*
(I)).

The pretreatment consists in reclaiming lost areas in the bins. Applying pretreatment to the initial instance allows some items to be enlarged. If the size of an item is updated to (L, l) the size of a bin, its optimal packing is then known and it can be treated separately.

The symbols and abbreviations used to define the pretreatment are: □ **m:** number of objects

- **n:** number of bins
- **A:** all objects as A = $\{a_i = (L_i, l_i), i=1... m\}$ □ **B:** all the bins of the same size as B = (L, l)

How to cite: Dr. Sofia Patel, Dr. Ethan Thompson (2020). Optimizing Bin Packing with Biologically Inspired Approaches: A Study on Bee Colony Optimization. Journal of Computer Engineering, 9(2), 22–30. DOI: https://doi.org/10.5281/zenodo.19341378

- **L:** width of each bin
- **l:** height of each bin
- **L_i:** width of the item a_i
- **l_i:** height of the item a_i
- For a_i in A: L_i l_i (i=1, 2... m)
- $_i$: equivalence class representative identical objects

Definition 1. The dimensions of the objects and bins are integers. An instance I of BPP-2D oriented is then defined by the pair (A, B). The optimal value of the number of bins needed to store all the objects of an instance I is denoted OPT (I).

Each object was stored in a bin and positioned such that the edge of L_i dimension is parallel to the width of the bin (L) and that the edge of l_i dimension is parallel to the height of the bin (l) (Fig. 1).

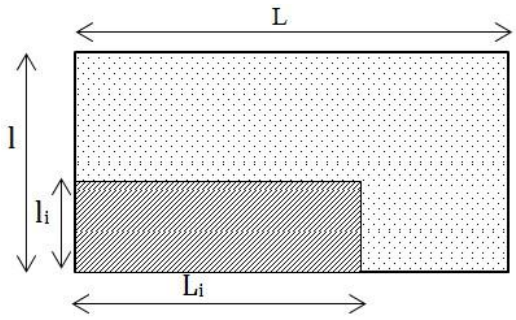

Fig. 1 Orientation of an object in a bin

Definition 2. Each item $a_i = (L_i, l_i)$ in A has:

- $L_i \leq L$ and $l_i \leq l$, otherwise the feasibility is impossible.
- $L_i \geq l_i$ (i = 1, 2... m), this constraint allows the storage of objects depending on the width.

A consists of the following subsets of objects: A_G, A_I, A_P, such as:

$$A = A_I \cup A_G \cup A_P$$

A_G: all large objects, A_I: set of identical objects defined by the equivalence class. A_P: all sufficiently small objects.

Definition 3. $_i$ represents the equivalence class contains the objects having the height equal to l_i

- $_i = \{a_i \in A_I$: with small dimension equal to $l_i\}$
- $_i = cl(a_i) = \{a_j \square A_I : l_j = l_i\}$
- $A_I = \cup$ $_i$ such as $\square$ $i \neq j$: $_i \cap$ $_j = \emptyset$
- $l(_i)$: height of the objects of the equivalence class $_I$ with : $\forall i = 1,...,k : l(_i) > l(_{i+1})$

Each equivalence class contains objects having the same height. The objects of the same equivalence class are sorted in order of descending width.

Mathematical Formulation

$$\sum \in \{0,1 \quad 1, \dots, \quad (1)$$

$$\sum 1 \quad \in \{0,1\} \quad (\forall a_i \in I, z = 1, \dots, n) \quad (2)$$

$$\sum \in L \quad L \quad (3a)$$

$$z \quad 1, \dots, n$$

$$\sum \in l \quad l \quad (3b)$$

$$z1, \dots, n$$

$$\sum \in \quad 1 \quad (4)$$

Decision Variables

- $_z$: If the bin z is used then $_z = 1$, otherwise $_z = 0$
- $_{iz}$: If the item a_i is assigned to the bin z then $_{iz} = 1$, otherwise $_{iz} = 0$
- $_{izt}$: If the item a_i is assigned to the floor t in the bin z then
- $_{izt} = 1$, otherwise $_{izt} = 0$

In this model, the objective is to minimize the total number of bins used. Constraints (2) ensure that each item is placed in one bin. Constraints (3a) ensure that the sum of widths of the items placed in the bin z, do not exceed the width of the bin. Constraints (3b) ensure that the sum of heights of the items placed in the bin z, do not exceed the height of the bin. Finally, constraint (4) controls whether each object is placed in one and only one level within a single bin.

Feasibility Constraints

This pretreatment aimed at storing oriented objects in their large size. Objects, sorted in order of decreasing width are classified into three categories:

- Large objects, denoted A_G, are the objects that occupy more than the half of the bin in the width and in the height.
- Identical objects, denoted A_I, are defined by the equivalence classes.
- Sufficiently small objects, denoted A_P, are objects whose dimension is less than a threshold established.

All A_I objects are in conflict with those A_G hence the expression:

$$\forall a_i \in A_I, \forall a_j \in A_G : \quad _{iz} + \quad _{jz} \leq _z$$

$$(i, j=1, \dots, m, z=(1,\dots,n)) \quad (5)$$

A conflict between two objects is a restriction prohibiting storing these objects in the same bin [13]. A conflict also

exists between the objects belonging to different equivalence classes in the same level, such as:

$$\square\ a_{i'} \square\ _i , \square\ a_{j'} \square\ _j : f_{i'zt} + f_{j'zt} \leq 1$$

$$(a_{i'} , a_{j'} \in A_I, i \neq j \text{ and } i , j \in \{ 1,...,k\} ,\forall z,t) \quad (6)$$

So the objects $a_{i'}$ and aj' cannot be stored in the same level of a given bin.

Category 1

A_G is the set of large objects, such as:

$$A_G : l_i = \{ a_i \square\ L \text{ and } l \}$$

If the large object takes the size of a bin, its optimal storage becomes trivial and can then be separated from the body to solve. Otherwise, items belonging A_P are stored with the large object in the bin.

Note: The number of bins needed to store the objects of A_G is equal to $| A_G |$. $| A_G |$ determines the number of bins required to store large objects. Thus, the total number of bins is n such that $n \ | A_G |$.

Category 2

Let A_I all identical objects defined as:

$$A_I = \{ a_i \in A : \frac{1}{4} L < L_i \leq L \}$$

Category 3

Let A_P the set of sufficiently small objects, such as:

$$A_P = \{ a_i \in A : L_i \leq L \}$$

These are items that have an enough small size so they can be stored with the objects of A_G and / or A_I.

Storage of Objects

The storage of the objects must be preceded by a sorting which is done according to the following strategy: Large objects are sorted according to decreasing width, followed by the intermediate objects (the majority) who themselves are divided into equivalence classes sorted according to decreasing height (recall that in each equivalence class the objects are sorted in order of descending width). And finally, the objects sufficiently small are, such as large objects, sorted in order of decreasing width.

Example of an Instance Illustrating This Sort

A= (12,10); (11,7); (10,6); (9,7); (7,6); (6,6); (6,6); (6,5); (6,5); (5,5); (5,5); (6,4); (6,4); (5,4); (5,4); (4,4); (4,4); (6,3); (6,3); (5,3); (5,3); (4,3); (4,3); (6,2); (6,2); (5,2); (4,2); (4,2); (6,1); (6,1);(5,1); (4,1); (4,1); (3,3); (3,2); (2,2); (1,1).

B= (12, 10)

For this instance, we have:

- **5** large objects: {(12, 10) ; (11,7) ; (10,6) ; (9,7) ;(7,6)}
- **6** equivalence classes:

$_1$ = {(6, 6); (6, 6)}
$_2$ = {(6, 5); (6, 5); (5, 5); (5, 5); (5, 5)}
$_3$= {(6, 4); (6, 4); (5, 4); (5,4); (4,4); (4,4)}
$_4$= {(6,3); (6,3); (5,3); (5,3); (4,3); (4,3)}
$_5$ = {(6, 2); (6, 2); (5, 2); (4, 2) ;(4, 2)}
$_6$ = {(6, 1); (6, 1); (5, 1); (4, 1); (4, 1)}

- **4** sufficiently small objects: {(3,3); (3,2); (2,2); (1,1)}

Storage of Large Objects

For each object $a_i \in A_G$, two new bins B_i^1 and B_i^2 are considered as: $B_i^1 = (L - L_i, l_i)$ and $B_i^2 = (L, l - l_i)$. This is to calculate the dimensions of the two bins $B_i^1$and B_i^2, and fill them with a subset $A_{P'} \subset A_P$ (Fig. 2).

$Pr_1 : a_i \rightarrow a_i'$

$L_i' = L$ and $l_i' = l$ if $a_i \in A_G$
$L_i' = 0$ and $l_i' = 0$ if $a_i \in A_{P'}$
$L_i' = L_i$ and $l_i' = l_i$ otherwise

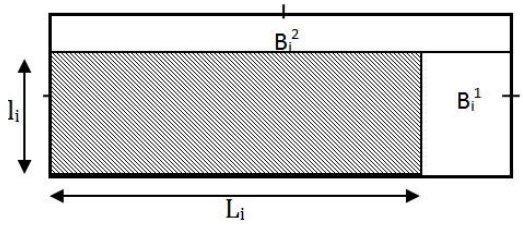

Fig. 2 Storage of a large object in a bin

The function Pr_1 is a procedure pretreatment valid if and only if $| A_G |$ bins are sufficient to store all objects of $A_G \cup A_{P'}$ such that $A_{P'} \subset A_P$.

Storage of Identical Objects

We consider different classes of equivalence in the order of descending height. A procedure for conflict detection is applied. It involves separating the equivalence classes in the same level of a given bin. The aim is to store items belonging to the same equivalence class to form a first layer in the bin. If the lost width obtained after placement of a set of objects belonging to $_i$ (i...k), is less $L_i \forall L_i \in _i$, objects $A_p \setminus A_{p'}$ are added.

For each equivalence class $_i$, we seek to build the subset A_i ^{r,q},, such that:

$$A_i^{r,q} = \{ a_i \in \mathcal{A}_i : \sum L_i = r \text{ and } l_i = q \text{ : } L = r \text{ and } l \}$$

Construction of the Subsets $A_i^{r,q}$

Initially, the equivalence classes are organized in descending order of height. The first object of the first equivalence class is deposited at the bottom left in a bin empty and the remaining objects are placed to the right of it until the current object is not be placed according its width. First level is formed whose the height is equal to the height of the equivalence classes considered.

Another subset belonging to the same equivalence class or to an equivalence class of greater height can be superposed while l-q l($_i$) ∀ i∈ {1,...,k}. This procedure is repeated until the optimal packing is obtained in the bin.

Remark: To ensure optimality, it is necessary to put a maximum number of objects of the same equivalence classes in the same level (if the cardinality of the equivalence class permitting). Otherwise select the next class.

For each subset $A_i^{r,q}$ creates two new bins B_i^3 and B_i^4 are considered: B_i^3 = (L – r, q) such as: L_i^4 = (L – r and B_i – q)

$(\forall L_i \in \mathcal{A}_i : \forall i=1,...k)$ l

Either the two decision problems: Is it possible to store objects $A_P \setminus A_{P'}$ in the bin B_i^3? Is it possible to consider the bin B_i^4 with the principle of storing identical objects? Two cases:

- $1^\circ : l - q \geq l($ $_i)$ $(i=1,\cdots,k)$, build a new level above the previous
- $2^\circ : l - q < l($ $_i)$ (i=1,...,k), objects small enough to

$A_P \setminus A_{P'}$ are stored in the bin B_i^4.

If the answers are yes, then the following function is an IFF associated to the considered instance I: $Pr_2 : A_i^{r,q} \rightarrow A_i^{r,q'}$

$r' = L$ and $q' = q$ if ($a_i \in A_I$ and $L - r < L_i$ and $(l - q \geq l($ $_i)$, i=1,...,k)) $r' = L$ and $q' = l$ if ($a_i \in A_I$ and $L - r < L_i$ and $(l - q < l($ $_i)$ i=1,...,k))

$L_i' = 0$ and $l_i' = 0$ if ($a_i \in A_P \setminus A_{P'}$)

$L_i' = L_i$ and $l_i' = l_i$ otherwise

III. Adaptation of the Heuristic First-Fit

The objects are sorted by decreasing width and oriented as follows: Each object stored in a bin is positioned such that the edge of L_i dimension is parallel to the width of the bin (L) and the edge dimension l_i is parallel to the height of the bin (l). The objects are placed successively according to their widths in the bins for forming different levels. More formally, at first, one bin is considered and objects are sorted in descending order of width. The first object (the largest) is placed in the bottom left in the bin, and other objects are placed (depending on the sort order) to the right of the first object until the current object cannot be placed by its width. A first stage is formed and its height is determined by the highest object.

If the remaining height in the bin (after creating the first floor) is greater than or equal to the height of the candidate object, a new floor is created above the previous one. Otherwise, a new bin is considered without closing the first.

In an intermediate step where we have opened k bins numbered from 1 to k in the order of their first use, a current object is stored in the level of the bin of lower number having enough space to contain it. If no bin can contain the object a_i, a new bin (k + 1) is opened without closing the others. The general procedure is repeated until all the objects of the instance considered will be stored.

IV. Optimization Using the Bee Colony Algorithm

The algorithm is based on the natural bee behavior:

a. The bee colony can spread over long distances in multiple directions (over 10 km). The nectar or pollen should be visited by more bees.
b. The scout bees look for the food source randomly from one plot to another.
c. The bees returning to the hive evaluate different food sources in function of a certain quality threshold (measured by a combination of some elements, such as the sugar content).
d. The bees deposit the nectar or pollen and go to the dance floor to perform a waggle dance.
e. The bees communicate through the waggle dance for giving the following information: The direction of the source (angle between the sun and the source), the distance from the hive (duration of the dance) and the quality of the food source (frequency of the dance).
f. Thus, the bees can collect food effectively and quickly.
g. This source will be announced in a new waggle dance to determine the current food level. If this level is suitable more bees will be recruited for this source.
h. Depending on the physical condition, sources with copious amounts of nectar or pollen can be visited by more bees or may be abandoned.

Basic Algorithm of the Bee Method

The pseudo code of the basic bee's algorithm is presented in the simplified form:

a. Initialize the population randomly
b. Evaluate the fitness of the population
c. While stopping criterionis not met
d. Select m solutions amongst n to perform local search
e. Recruit bees for selected solutions and evaluate their fitness
f. Select the best bee in each neighborhood
g. Assign the remaining bees randomly and evaluate their fitness End while

The Parameters of the Simulation

Objects parameters:

- Number of objects to be stored
- Maximum dimensions (L_i and l_i) of each objects $a_i \in$ A Bins parameters:
- Number of bins
- Size of each bin (L, l) Bee algorithm parameters:
- Number of bees
- Percentage of each type of bees (Scouts, active and inactive)
- Number of visits
- Number of iterations

To select and set the parameters of the bee algorithm, various tests were performed. The best combination providing an optimal solution was maintained. Indeed, we varied the number of:

- Active bees (between 30% and 60%)
- Scout bees (between 20% and 40%)
- Inactive bees (between 10% and 30%)
- Iterations from 5 to 50

The Coding Parameters

The nectar (or pollen) represents the object to be placed in the bin. The hive represents the bin to fill with objects. The Quality, proximity and ease of extraction of the food source are the fitness of the solution.

TABLE I
CATEGORIES AND PROBABILITIES OF THE OBJECTS

Category 1	Category 2	Category 3
-Probability 10% - The Width L_i is randomly generated following a uniform distribution whose numerical values are between 7 and 12 - l_i height is less than or equal to L_i	-Probability 50% - The Width L_i is randomly generated following a uniform distribution whose numerical values are between 4 and 6 - l_i height is less than or equal to L_i	-Probability 40% - The Width L_i is randomly generated following a uniform distribution whose numerical values are between 1 and 3 - l_i height is less than or equal to L_i

We varied the number of objects in each instance: 40, 80, 160, and 240. The instance is characterized by three categories of objects with a probability of existence (Table I). We consider the bins that have a width equal to 12 and a height equal to 10.

Computational Results

The algorithm is implemented in JAVA programming language. The choice of integrated development environment NetBeans 7.4 was guided by the benefits of object oriented programming. The results for a few samples with 40, 80, 160 and 240 objects are given in Table II.

TABLE II
COMPARISON OF THE NUMBER OF BINS TO PACK THE OBJECTS

Set	Number of objects	n bin without pretreatment	n bin with pretreatment	Optimum
1	40	08	08	08
2	80	14	13	13
3	160	30	28	28
4	240	46	38	38

The minimum number of bins used to store the objects of each sample is obtained by applying our pretreatment. This was specially the case when the number of objects increases.

V. CONCLUSION

[13] K. Jansen and S. Öhring. Approximation algorithms for time constrained scheduling. Information and Computation, 1997; 132(2); 85–108.

In this paper, we have presented a mathematical formulation for BPP-2D the oriented case. We have presented an adaptation of the heuristic first fit for the resolution of the problem where we stop only when the empty space in the bin is less than the dimensions of the remaining objects. We have compared the numbers of bins used to pack all objects of the instances with and without the pretreatment. The minimum number of bins used to store the objects of each instance is obtained by applying our pretreatment. This good result is mainly due to the introduction of the concept of equivalence classes in the pretreatment.

REFERENCES

[1] Lee J, kim B,. Johnson A L. A two-dimensional bin packing problem with size changeable items for the production of wind turbine flanges in the open die forging industry. IIE Transactions, 2013; 45, 12, 1332-1344

[2] Lodi A, Martello S, Vigo D. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS Journal on Computing 1999;11:345–57.

[3] Lodi A, Martello S, Vigo D. Recent advances on two-dimensional binpacking problems. Discrete Applied Mathematics 2002;123:379–96.

[4] Berkey JO,Wang PY. Two-dimensional finite bin-packing algorithms.
Journal of Operational Research Society 1987;38:423–9.

[5] Faroe O, Pisinger D, Zachariasen M. Guided local search for the three-dimensional bin-packing problem. INFORMS Journal on Computing 2003;15:267–83.

[6] Carlier J, Clautiaux F, Moukrim A. New reduction procedures and lower bounds for the two-dimensional bin-packing problem with fixed orientation. Computers & Operations Research.2006.

[7] Boschetti MA, Mingozzi A. The two-dimensional finite bin-packing problem. Part I: new lower bounds for the oriented case. 4OR 2003; 1:27–42.

[8] Martello S, Vigo D. Exact solution of the two-dimensional finite binpacking problem. Management Science 1998;44:388–99.

[9] Fekete S, Schepers J. A general framework for bounds for higher-dimensional orthogonal packing problems. Mathematical Methods of Operations Research 2004;60:311–29

[10] Boschetti MA, Mingozzi A. The two-dimensional finite bin-packing problem. Part I: new lower and upper bounds. 4OR 2003;1:135–47

[11] Harwig J, Barnes J. An adaptive tabu search approach for 2-dimensional orthogonal packing problems. Military Operations Research, 2006,in press.

[12] Chazelle B. The bottom-left bin-packing heuristic: an efficient implementation. IEEE Transactions on Computers 1983;C-32:697–707.