An Analysis of Solutions to the 2D Bin Packing Problem and Additional Complexities

Amitesh Anand Pandey

August 19, 2021

Abstract

The Bin Packing problem in 2 space is an NP-Hard combinatorial problem in optimization of packing and arrangement of objects in a given space. It has a wide variety of applications ranging from logistics in retail industries to resource allocation in cloud computing. In this paper, we discuss the mathematical formulation of this problem. Furthermore, we analyse its time complexity, its NP-Hardness and some of its stochastic solutions with their efficiencies. We then propose additional complexities that would make the problem more fit for industrial use and discuss in depth the domains in which it might prove to be useful. We conclude while suggesting areas of improvement in operations research on this subject.

Contents

1	Introduction	2
2	Mathematical Formulation	2
3	Proof of NP-Hardness	5
4	Optimization Techniques	6
4.1	Approach 1	6
4.2	Approach 2	7
5	Additional Complexities	9
6	Conclusion	10
7	Scope for Further Research	10

1 Introduction

The 2D Bin Packing problem represents an array of industrial applications. It has usage in a variety of fields, like industrial logistics, and cloud resource allocation. In its most nascent form, the problem asks of the most optimised (and least resource intensive) way to arrange a given set of rectangles into a larger square/rectangle. The 2D Bin-Packing Problem (2D-BPP) is strongly NP-Hard since it is but a generalization of the NP-Complete 1D Bin Packing Problem. [1]

Figure 1: Two Representations of a Bin Packing Arrangement [2]

The 2D Bin Packing problem can be defined as a simple optimisation of the Height (H) of an arrangement, given a container of infinite height and fixed width. The lower this height (H), the better the arrangement. An optimization as such in the case of the single dimensional Bin Packing problem is achieved through algorithms like k-Harmonic and RFF Bin Packing. However, for the 2D BPP, more nuanced solutions need to be acquired. Solutions using a genetic algorithm are "heuristic", meaning that they are not the best solutions out there, but are 'good-enough' for industrial use. [3]

2 Mathematical Formulation

An ILP (Integer Linear Programming) equation is used to define optimization problems in a mathematically rigorous way. Though the 1D Bin Packing Problem can simply be defined as in equation (1).[4]

Definition 2.1 *Instance: Finite set I of items, a size $s(i) \in \mathbb{Z}^+$ for each $i \in I$, a positive integer bin capacity B , and a positive integer K .*

Definition 2.2 *Question: Is there a partition of I into disjoint sets $I_1, \dots, I_K$ such that the sum of the sizes of the items in each I_j is B or less?*

$$\begin{aligned}
& \text{minimize } K = \sum_{j=1}^n y_j & (1) \\
& \text{subject to } K \geq 1, \sum_{i \in I} s(i)x_{ij} \leq By_j, \forall j \in \{1, \dots, n\} \\
& \sum_{j=1}^n x_{ij} = 1, \quad \forall i \in I \\
& y_j \in \{0, 1\}, \quad \forall j \in \{1, \dots, n\} \\
& x_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in \{1, \dots, n\}
\end{aligned}$$

where $y_j = 1$ if bin j is used and $x_{ij} = 1$ if item i is put into bin j . Here, K is subject to multiple constraints. However, a definition of the 2D Bin Packing Problem retains complexity, and a generalization uses more nuanced Linear Programming techniques. Moreover, the formulation is subject to many more constraints, some of which we have detailed in the equations that follow. The 2BP problem consists of a set of n rectangular items, each having a height h_i and a width w_i , $i = 1, \dots, n$. The objective is to pack them into a minimum number of rectangular bins, each having height H and width W . Items may not overlap and we do not consider rotation.

A feasible layout for 3-stage 2BP consists of a set of bins, each bin consists of a set of stripes, each stripe consists of a set of stacks, and each stack consists of items having equal width. Every such pattern can be reduced into its so-called normal form by moving each item to its uppermost and leftmost position, so that void space appears only at the bottom of stacks, to the right of the last stack in each stripe and below the last stripe.

In the sequel we consider only patterns in normal form. In [4] a polynomial-sized ILP model for 2-stage 2BP has been proposed. We extend this model in order to get a polynomial-sized ILP formulation for 3-stage 2BP. The items are sorted so that $h_1 \geq h_2 \geq \dots \geq h_n$. The order of the items within each stack is not relevant, so they can always be ordered according to their indices. A solution may contain at most n stacks. We label each stack with the index of the highest item it contains, i.e., the smallest item index.

Similarly, a solution has at most n stripes, and a stripe's label is the label of its highest stack. Finally at most n bins are needed and we label each of them with the smallest index of the stripes it contains. The model uses the following variables

- $\alpha_{j,i}, j = 1, \dots, n, i = j, \dots, n$: rectangle i is contained in stack j
- $\beta_{k,j}, k = 1, \dots, n, j = 1, \dots, n$: stack j is contained in stripe k ;
- $\gamma_{l,k}, l = 1, \dots, n, k = l, \dots, n$: stripe k is contained in bin l ;
- $\delta_{l,i,j}, l = 1, \dots, n-1, i = l+1, \dots, n$, and $j = l, \dots, i-1$: item i contributes to the total height of all stripes in bin l ;

i.e., item i appears in stack j , stack j appears in stripe j , and stripe j appears in bin l . All algorithms that we will review later will be an attempt to minimize equation (2) while satisfying the constraints as shown. [5]

$$\text{minimize } \sum_{l=1}^n \gamma_{l,l} \quad (2)$$

$$\begin{aligned} \text{subject to } & \sum_{j=1}^i \alpha_{j,i} = 1, \forall i = 1, \dots, n \\ & \sum_{i=j+1}^n \alpha_{j,i} \leq (n-j)\alpha_{j,j}, \quad \forall j = 1, \dots, n-1 \\ & \alpha_{j,i} = 0, \forall j = 1, \dots, n-1 \mid \forall i > j \mid w_i \neq w_j \wedge h_i + h_j > H \\ & \sum_{k=1}^n \beta_{k,j} = \alpha_{j,j} \quad \forall j = 1, \dots, n \\ & \sum_{i=j}^n h_i \alpha_{j,i} < \sum_{i=k}^n h_i \alpha_{k,i} + (H+1)(1-\beta_{k,j}), \\ & \forall k = 2, \dots, n, \quad \forall j = 1, \dots, k-1 \\ & \sum_{i=j}^n h_i \alpha_{j,i} \leq \sum_{i=k}^n h_i \alpha_{k,i} + H(1-\beta_{k,j}), \\ & \forall k = 1, \dots, n-1, \quad \forall j = k+1, \dots, n \\ & \sum_{j=1}^n w_j \beta_{k,j} \leq W \beta_{k,k}, \quad \forall k = 1, \dots, n \\ & \sum_{i=l}^n h_i \left(\gamma_{l,i} + \sum_{j=l}^{i-1} \delta_{l,i,j} \right) \leq H \gamma_{l,l}, \quad \forall l = 1, \dots, n-1 \\ & \alpha_{j,i} + \gamma_{l,j} - 1 \leq \delta_{l,i,j} \leq (\alpha_{j,i} + \gamma_{l,j}) / 2, \\ & \quad \forall l = 1, \dots, n-1, \forall i = l+1, \dots, n, \forall j = l, \dots, i-1 \\ & \sum_{k=l+1}^n \gamma_{l,k} \leq (n-l)\gamma_{l,l}, \quad \forall l = 1, \dots, n-1 \end{aligned}$$

3 Proof of NP-Hardness

NP or Non-Deterministic Polynomial Time is a complexity class used to arrange decision problems on the basis of their complexity. Problems in the NP class have proofs of solutions verifiable in polynomial time. NP-Hard is also a complexity class, and it is used to arrange problems that are at least as hard as the hardest problems in the NP class. For proving that a problem is NP-Hard, the most common way is to deduce it from another previously known NP-Hard Problem. Here, we use the NP-Hard Partition Problem. We follow the proof described in [6]

Definition 3.1 (Partition Problem) *The Partition Problem is the task of deciding whether a given multi-set S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2 .*

Lemma 1 *Whether the problem: Given a finite set I of items, a size $s(i) \in \mathbb{Z}^+$ for each $i \in I$, a positive integer bin capacity B , and a positive integer K . Is there a partition of I into disjoint sets $I_1, \dots, I_K$ such that the sum of the sizes of the items in each I_j is B or less is NP-Hard or not.*

Given an instance $i_1, i_2, i_3, \dots, i_n$ of the Partition problem let us consider that, $I = \sum i_j$ and an input instance of the Bin Packing to be $s_j = 2i_j/I$. The above equation postulates for the splitting of a single instance of Bin Packing into two distinct instances such that the summation of both the sub-sets are equal, given that both are disjoint. This instance proves that the answer to the Partition problem is True if and only if the number of bins to pack for the Bin Packing problem is 2. If in case there exists a k -approximation algorithm which would solve Bin Packing in polynomial time ($k < 3/2$) then the answer to the above problem of “2 bins” would be always found. If the algorithms were to require more than 2 bins, i.e 3 bins then the optimal factor would be greater than $3/2$. Hence, the algorithm would solve the Partition problem in polynomial time which implies that $P = NP$, in which case Bin Packing can then be solved in polynomial time. Therefore, Bin Packing is an NP-Hard problem with approximation value $\geq 3/2$.

Theorem 2 *It is NP-complete to decide if an instance of Bin Packing admits a solution with two bins.*

Corollary 2.1 *There is no ρ -approximation algorithm with $\rho < 3/2$ for Bin Packing unless $P = NP$.*

The NP-Completeness of the problem can be derived using the 3-Partition Problem, a variant of the problem we used to prove 2D-BPP’s NP-Hardness.

4 Optimization Techniques

In this section we will review some of the most promising mathematical and computational models that yield consistent and precise arrangements for the 2D-BPP. Over the years, there have been numerous heuristic solutions to the problem suitable for industrial applications. These solutions represent vital findings in Operations Research and Industrial Engineering. We will broadly review 2 such solutions, based on entirely different techniques. One of the techniques uses a computational evolutionary algorithm whereas the other uses multi-variable calculus to offer solutions. We will also discuss the drawbacks and upsides of both the approaches in detail.

4.1 Approach 1

The m-M Calculus algorithm is the basis for our first approach. It starts with a reshaped bounding box and performs cell-decomposition until there are no more feasible cells, or some predefined level of accuracy is accomplished. In [7], the authors implemented a heuristic model to obtain an ILP equation equation (6) to maximise and used the m-M Calculus algorithm to solve it.

$$\max \sum_{i=1}^n \frac{\max(0, B - y_i)}{B - y_i} a_i b_i \quad (3)$$

Through the algorithm, they could obtain a theoretical expression for the x and y coordinates for an arbitrary rectangle that must be placed in the region. The expression is in equation (7).

$$\frac{b_i}{2} - \min(|by_i - B|, |B - ay_i|) \quad (4)$$

Here, ay_i and by_i are lower-left and upper-right y-coordinate of the i^{th} rectangular, respectively. The authors then bench marked the model's efficiency. The results of their study, run on a PC with P4 2G CPU and 960Mb RAM are shown in Figure 2. It is apparent to understand why this approach will be less resource intensive. The algorithm is a simple culmination of the m-M method, and even machines dating back 2 decades have the computational power to run this program without bottleneck. However, since the essence of the algorithm, the ILP equation itself is derived theoretically, and the parameters for computation are rooted deeply in combinatorial mathematics, inclusion of additional complexities as listed in section 5 will prove to be quite challenging.

	Test 1	Test 2
Number of rectangles	4	5
Dimensions of rectangles	2×6 3×2 3×4 5×1	1×7 2×2 2×2 2×5 4×2
Time for detecting first optimal solution (s)	2	69
Memory cost (MB)	61	681
Detected centers	1.0, 3.0. 4.5, 5.0. 4.5, 2.0. 3.5, 6.5.	4.5, 3.5. 3.0, 4.5. 3.0, 6.0. 1.0, 4.5. 2.0, 1.0.

Figure 2: Results from m-M Calculus Method

4.2 Approach 2

A genetic algorithm works on the fundamental principle of Evolution by Natural Selection, proposed by Darwin in [8]. In the case of our problem, a genetic algorithm would introduce mutations and crossovers to produce different generations of solution arrangements. It will then pick the fittest generations to create the next generation of arrangements. This process is iterative, and continues until the desired outcome (heuristically) is achieved, or subsequent generations offer no advantage. In [9], the authors combine an MSGA(Multi-Start Genetic Algorithm) with a BLF(Bottom Left Filling) Heuristic Algorithm to solve the 2D Bin Packing problem, and their study provides promising results. Figure 2 depicts a standard crossover introduced in a generation of solutions in a genetic algorithm for the 2D-BPP. A Mutation on the other hand, is a more local change, such as the swapping of elements within a solution. Multi-start genetic algorithm is an improved version of traditional genetic algorithm which enhances the search capability for global optimization. The details of the algorithm along with a flowchart depiction of its functioning for the purposes of this problem is depicted by Figure 4.

Figure 3: Sample Crossover for 2D Bin Packing
[9]

To solve the problem using a Multi-Start Genetic Algorithm, we need an algorithm for placing the rectangular bins inside the region. For this purpose, we use the Bottom-Left-Filling (BLF) Heuristic algorithm. The BLF Algorithm is an algorithm with a $O(N^2)$ time-complexity. As it arranges the bins, our genetic algorithm introduces previously explained mutations and crossovers to achieve the desired results. Every genetic algorithm requires a 'Lower Bound', a threshold to signal the stop of the algorithm, even before all the iterations are complete. This bound must be chosen optimally, and must satisfy precision constraints in the use-case. The authors find the Lower Bound in equation (8) to be pareto-optimal for an MSGA-BLF implementation.

$$LB = \sum_{i=1}^n \frac{w_i h_i}{WH} \quad (5)$$

Figure 4: Flowchart of the MSGA Algorithm
[9]

5 Additional Complexities

A variety of complexities can be introduced to the nascent problem. In reality, it makes more pragmatic sense for the problem to be in R^3 . This problem already exists, and trivially goes by the moniker "The 3D Bin Packing Problem". It minimises the blank space in an arrangement of cuboids (or boxes) in a given volume V . In Figure 4, we have provided a visualization of how cuboids of different dimensions might be placed in a fixed volume.

Figure 5: Sample Arrangement for 3D Bin Packing
[10]

Additionally, in industrial use, it is sometimes useful to have some area reserved while packing objects in fixed space. This area can be for transportation purposes, machinery space or ease of accessibility. In this case, the problem expands on the simple rectangular region, and an arbitrary region is constructed, as shown in Figure 5. There are ways to tackle this complexity without making changes to the core algorithm we use. For instance, the region can be divided into ever-so-fitting smaller rectangular regions and the algorithm could then be used for each of the sub-rectangles individually.

Figure 6: How inclusion of a path changes an arrangement

6 Conclusion

Through the means of this paper, we have explored two approaches that can be used to construct arrangements that act as solutions to the 2D Bin Packing Problem. Using the inference from the Multi-start Evolutionary Genetic Algorithm, we also discussed possibilities of extension of the problem to more nuanced combinatorial arrangements for real world use.

7 Scope for Further Research

With the combinatorial problem at hand, possibilities of further research are multifold. More optimization research can be undertaken to truly generalize an evolutionary algorithm to accommodate paths in the rectangular region. Additionally, research can be conducted to understand in greater detail the functionalities of NP-Hard optimization problems in Virtual Machine resource allocation in cloud servers.

References

- [1] R. J. Pargas, Roy P., “A parallel stochastic optimization algorithm for solving 2d bin packing problems.,” in *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, pp. 18–25, IEEE, 1993.
- [2] R.-S. Y. Cid-Garcia NM, “Positions and covering: A two-stage methodology to obtain optimal solutions for the 2d-bin packing problem.,” *PloS ONE*, vol. 15(4).
- [3] T.-K. C. H. S. Y. G. C. K. . H. W. K. Liu, D. S., “On solving multiobjective bin packing problems using evolutionary particle swarm optimization.,” *European Journal of Operational Research*, vol. 190(2), pp. 357–382, 2008.
- [4] P. T. Silvano Martello, *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and Sons, 1990.
- [5] G. R. R. Jakob Puchinger, “An evolutionary algorithm for column generation in integer programming: An effective approach for 2d bin packing,” in *PPSN VIII, 8th International Conference*, pp. 642–651, 2004.
- [6] S. T. Bhuvaneshwaran Ravi, Kameswaran Rangasamy, “Bin packing, proof of np completeness and hardness.,” *University of Bishops*, 2006.
- [7] F. V. Tijana Šukilović, “Solving the two-dimensional packing problem with mm calculus.,” *Yugoslav Journal of Operations Research*, vol. 21, pp. 93–102, 2011.
- [8] C. R. Darwin, *On the Origin of Species*. Routledge, 2004.
- [9] N. L. Khaoula Hamdi-Dhaoui and A. Yalaoui, “Algorithms for the two dimensional bin packing problem with partial conflicts,” *RAIRO Operations Research*, vol. 46, pp. 41–62, 2012.

- [10] B. Y. Samir Elhedhli, Fatma Gzara, “Three-dimensional bin packing and mixed-case palletization,” *Inform's Journal on Optimizaiton*, vol. 1(4), pp. 323–352, 2019.