Theoretical Computer Science 661 (2017) 56–64

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Parameterized complexity of Strip Packing and Minimum Volume Packing

Pradeesha Ashok[a], Sudeshna Kolay[a], S.M. Meesum[a,*], Saket Saurabh[a,b,1]

[a] Institute of Mathematical Sciences, India
[b] University of Bergen, Norway

ARTICLE INFO

Article history:
Received 19 August 2016
Accepted 28 November 2016
Available online 8 December 2016
Communicated by F.V. Fomin

Keywords:
Strip Packing
Volume minimization
Greedy packing

ABSTRACT

We study the parameterized complexity of Minimum Volume Packing and Strip Packing. In the two dimensional version the input consists of a set of rectangles S with integer side lengths. In the Minimum Volume Packing problem, given a set of rectangles S and a number k, the goal is to decide if the rectangles can be packed in a bounding box of volume at most k. In the Strip Packing problem we are given a set of rectangles S, numbers W and k; the objective is to find if all the rectangles can be packed in a box of dimensions $W \times k$. We prove that the 2-dimensional Volume Packing is in FPT by giving an algorithm that runs in $(2 \cdot \sqrt{2})^k \cdot k^{\mathcal{O}(1)}$ time. We also show that Strip Packing is W[1]-hard even in two dimensions and give an FPT algorithm for a special case of Strip Packing. Some of our results hold for the problems defined in higher dimensions as well.

1. Introduction

The problem of packing objects optimally inside a bin/box is studied in various forms. Two prominent examples are the Bin Packing problem and the Knapsack problem. We study generalizations of these problems in a geometric setting. A natural generalization of the Knapsack problem to two dimensions is the Cutting Stock problem [3]. More specifically, we study the problem of packing axis-parallel rectangles inside a rectangular box when only translations are allowed. This restriction of not allowing rotations does not make the problem artificial as it still finds practical application [15]. These problems also find applications in VLSI design [16], scheduling [17], packing television commercial into station breaks [4] etc.

We consider two versions of this problem. In the Strip Packing problem, given a set of n axis-parallel rectangles and a rectangular box (strip) of width W, the goal is to pack all rectangles into this strip so that the height used is minimized. In the Minimum Volume Packing problem, given a set of n axis-parallel rectangles, goal is to pack these rectangles in a rectangular container so that the volume of the container is minimized. We also study these problems in higher dimensions. Now we formally define the problems:

[*] Corresponding author.
E-mail addresses: pradeesha@imsc.res.in (P. Ashok), skolay@imsc.res.in (S. Kolay), meesum@imsc.res.in (S.M. Meesum), saket@imsc.res.in (S. Saurabh).
[1] The research leading to these results has received funding from the European Research Council (ERC) via grant PARAPPROX, reference 306992.

http://dx.doi.org/10.1016/j.tcs.2016.11.034
0304-3975/© 2016 Elsevier B.V. All rights reserved.

d-Strip Packing
Input: A list of boxes $\{b_i \in \mathbb{N}^d \mid 1 \le i \le n\}$ and a vector of positive integers $W \in \mathbb{N}^{d-1}$.
Output: The minimum integer k such that all the boxes can be packed under translation into a strip with dimension vector $W \times k$, k being the height of the strip.

Volume packing in two-dimensions is an active area of research due to its immense practical importance [11].

d-Volume Packing
Input: A list of n boxes $\{b_i \in \mathbb{N}^d \mid 1 \le i \le n\}$.
Output: The minimum volume rectangular container into which the input boxes can be packed under translation.

The decision versions of both these problems are proved to be NP-Complete and are well studied in the context of approximation algorithms. Two-dimensional Strip Packing admits an AFPTAS [14]. Since Bin Packing, which is a special case of Strip Packing, cannot have a PTAS [2], the same holds for Strip Packing. For recent approximation algorithms and results on Minimum Volume Packing see [1]. The online version of these problems are also studied [10,18,12]. For a survey of these problems we refer the reader to [15,5].

In this paper we study Strip Packing and Minimum Volume Packing from the perspective of *parameterized complexity* with the hope that it would lead to a better understanding of these problems. An instance of the parameterized version of a problem comes with a parameter k. A problem with an input instance of the form (I, k) is said to be *fixed parameter tractable* (FPT) if there is an algorithm that solves the problem in $f(k)|I|^{\mathcal{O}(1)}$ time where $|I|$ is the input size and f is a computable function that depends only on k. Similar to classical complexity, there is a notion of *hardness* in parameterized complexity and tools and techniques for reductions. There is a set of problems called as W-hard problems which are not believed to have an FPT algorithm.

We consider the standard parameterized version of these problems, namely the case when the parameter is the size of the solution. To the best of our knowledge, this is the first study on the parameterized version of these problems. For the Strip Packing problem, we prove that the parameterized version is W-hard for the general case. However, the problem becomes FPT for a special case where the dimensions of the boxes and therefore the number of types of boxes is bounded by a constant, this special case is also inspired by a variant of the Bin Packing problem with similar constraints [9]. We also consider several special kind of input box dimensions where PTAS and even polynomial time algorithms exist.

Our results:

1. We prove that 2-Minimum Volume Packing is in FPT by giving an algorithm with running time $(2 \cdot \sqrt{2})^k \cdot k^{O(1)}$. This algorithm can be generalized to d-dimensions.
2. We prove that Strip Packing is W-hard even in two dimensions.
3. We consider a special case of Strip Packing where every dimension of the input boxes is bounded by a constant. For this case, we show that Strip Packing problem admits an FPT algorithm.
4. We consider some special cases of Strip Packing where the input rectangles are squares whose dimensions come from a special set and show that they are solvable in polynomial time.

Section 2 gives some notations and definitions that will be used in subsequent sections. In Section 3, we discuss the FPT algorithm for Minimum Volume Packing problem. Section 4 gives algorithmic and hardness results for Minimum Strip Packing. Section 5 gives polynomial time algorithms for special cases.

2. Preliminaries

The symbols $\mathbb{R}$ and $\mathbb{N}$ denote the set of real number and positive integers respectively. For a positive integer n, we use the notation $[n]$ to denote the set $\{1, \dots, n\}$.

Parameterized complexity: A parameterized problem Π is a subset of $\Sigma^* \times \mathbb{N}$. Given a parameterized decision problem with input $x \in \Sigma^*$ of size n, and an integer parameter k, the goal in parameterized complexity is to design a deterministic algorithm which decides the membership of the instance (x, k) in Π in time $f(k)n^{\mathcal{O}(1)}$, where f is a function of k alone. Problems which admit such algorithms are said to be fixed parameter tractable (FPT).

It is believed that there is a hierarchy of problems in the W-class, such that the class $\mathrm{W}[i] \subset \mathrm{W}[i+1]$, for all $i \ge 1$. It is also believed that a W-hard problem does not have an FPT algorithm. To establish these classes, a notion of FPT-reductions is defined. Given two parameterized problems Π, Γ, an FPT-reduction from Π to Γ is an FPT algorithm that takes in an instance I of Π and outputs an instance I' of Γ such that I is a YES instance of Π if and only if I' is a YES instance of Γ. To determine the hardness of a parameterized problem, it is enough to give an FPT-reduction from a known W-hard problem to our required problem. For the context of this paper, it is enough to know that Bin Packing, in unary representation and parameterized by the number k of bins allowed, is W[1]-hard [13]. To show W[1]-hardness of a given problem, it is enough to give an FPT-reduction from Bin Packing, in unary representation and parameterized by k, to the given problem. For more information about these concepts, tools and techniques we refer the reader to monographs such as [8,7,6].

Packing: In this paper, we only consider axis parallel boxes and containers with dimensions that are positive integer values. When the boxes are in $\mathbb{N}^2$ they are referred to as either *integral* rectangles or rectangles. The faces of rectangles are also referred to as edges. The dimensions of a box in $\mathbb{N}^d$ is denoted by a vector of length d, where the ith index indicates the length of the projection of the axis-parallel box onto the ith axis in $\mathbb{R}^d$. Similarly, we can define a dimension vector for a container. For a dimension d, given a set of positive integers $\ell_1, \dots, \ell_d \in \mathbb{N}$, a container C in d-dimensions with dimension vector $(\ell_1, \dots, \ell_d)$ is formally defined to be the set of points $R = \{(x_1, \dots, x_d) \in \mathbb{R}^d | x_i \in [0, \ell_i], \forall i \in [d]\}$. An ith lower *face* of R is the set $R \cap \{x_i = 0\}$. Similarly an ith upper *face* of R is the set $R \cap \{x_i = \ell_i\}$. Each rectangle in d dimensions has $2 \cdot d$ faces in total. A *corner* of a rectangle is defined as the point of intersection of d non-parallel faces. The *corner* of a rectangle which is at closest distance (L_2-norm) to the origin shall be referred to as a *min-corner* point. A dimension vector, where each index stores a positive integer, is also referred to as an *integral* dimension vector. Given a dimension vector $(\ell_1, \dots, \ell_d)$, the *volume* corresponding to the dimension vector is $\prod_{i=1}^{d} \ell_i$. Two boxes with the same dimension vector will be referred to as the same *type*. In this paper, we allow the boxes to be packed in such a way that they are axis-parallel and only translations are allowed.

For ease of presentation, in the case of $\mathbb{R}^2$, we refer to directions like up, left, down, right. In 2-dimension, the lower left corner of the container is assumed to be at the origin.

Given a container and input boxes, such that all dimension vectors are integral, we call an arrangement of rectangles inside a container a *packing* if the rectangles are completely contained inside the container and do not overlap (except at the faces). A *packing* is said to be *integral* if the input rectangles are packed in such a way that each corner point of every rectangle is placed on a point with integer coordinates. A packing is called as a *perfect packing* if for each box b and each axis i, the minimum value taken by the ith coordinate, over all points belonging to a face of b, cannot be decreased, while keeping the coordinates of all other boxes fixed and maintaining a packing of all input boxes. Given a fixed container C and a collection of boxes $\mathcal{R}$, a packing of $\mathcal{R}$ in C is said to be a *tiling* if every point inside C is contained inside some box in $\mathcal{R}$. For this to happen the sum of *volume* of boxes must be equal to the *volume* of the container.

Parameterized packing problems: In this paper, we look at the following parameterized variants of d-Strip Packing and d-Volume Packing.

d-p-Strip Packing **Parameter:** k
Input: A list of boxes $\{b_i \in \mathbb{N}^d \mid 1 \le i \le n\}$, $W \in \mathbb{N}^{d-1}$ and $k \in \mathbb{N}$.
Question: Is there a container with dimensions $W \times k$ such that all the boxes can be packed into it under axis-parallel translation?

d-p-Strip Packing with bounded dimensions **Parameter:** k
Input: A list of boxes $\{b_i \in \mathbb{N}^d \mid 1 \le i \le n\}$, where each side length of a box is at most $\ell \in \mathbb{N}$, a vector of positive integers $W \in \mathbb{N}^{d-1}$ and $k \in \mathbb{N}$.
Question: Is there a container with dimensions $W \times k$ such that all the boxes can be packed under axis-parallel translation into it?

d-p-Volume Packing **Parameter:** k
Input: A list of boxes $\{b_i \in \mathbb{N}^d \mid 1 \le i \le n\}$ and a positive integer k.
Question: Is there a container with volume at most k such that all the rectangles can be packed under translation into the container?

3. Volume Packing

In this section, we show that d-p-Volume Packing is FPT. In order to prove this, we first derive a few relations between optimal packings and integral packings. These relations will be useful in later sections as well.

Lemma 1. *Keeping the dimension vector of the container same, any packing of boxes can be changed into a* perfect packing. *Moreover, every* perfect packing *of boxes with integral dimension vectors is also an* integral packing.

Proof. We apply a sweeping line algorithm to rearrange the rectangles in 2-dimensions. In higher dimensions, it may be looked upon as a sweeping hyperplane algorithm. We call a *hyperplane* perpendicular to the ith axis in d dimensions as an i-plane. For each $i \in [d]$, we take an i-plane and sweep it towards the positive direction of the ith axis starting from the zero value of the ith axis. The moment when the face of a box gets completely contained inside an i-plane is referred to as an *event*. At every *event* t, we keep a list L_t^i of boxes where a *face*, which is perpendicular to the ith axis, is contained in it completely. As the packing is axis parallel there are only two faces for every box which will result in an *event* and they can be unambiguously identified as an upper face or a lower face depending on the value of their ith coordinate. Divide L_t^i into two disjoint sets depending on which face of the box is completely contained in it; $L_{t_u}^i$ denotes the boxes where the upper face lies on the sweeping hyperplane at t, and $L_{t_l}^i$ denotes the boxes with the lower face lying on the sweeping hyperplane at t. Let $F_t^i \subseteq L_{t_l}^i$ be the set of boxes whose lower face does not intersect with the upper face of any box in $L_{t_u}^i$. Observe that the boxes in F_t^i can be "slid down" parallel to the ith axis until either they hit a face of a rectangle below or hit a face of the container. More formally, it is possible to change the position of a box in F_t^i, without changing the volume of the bounding container, such that the value at the ith coordinate of each point in the box is strictly less than the current value. For each box in F_t^i, we change the packing, such that all points in the boxes of F_t^i have the minimum possible value at the ith coordinate, without changing the position of boxes not in F_t^i. This operation can be thought of as "moving down" of boxes in F_t^i. After all the boxes in F_t^i have been moved down, any box in $L_{t_l}^i \setminus F_t^i$ intersects with some rectangle face in $L_{t_u}^i$. The invariant maintained is that the boxes whose lower face has been processed by an *event* have their lower face intersecting with the upper face of some rectangle whose upper face has been processed by an event. As the boxes have integral dimension vectors, inductively it gives us the result that, when the i-plane has swept through the lower face of every box in the packing, the faces perpendicular to the ith axis have integral co-ordinates. We keep repeating this procedure for every $i \in [d]$ until none of the i-plane sweep results in the set F_t^i being non-empty. It is possible that multiple sweeps have to be made for each i-axis. At the end of the procedure the corner points of each box get integral co-ordinates as they are the intersection points of the faces of the boxes.

To see why this procedure terminates, consider the sum S_b of the coordinates of the *min-corner* point of each box b in the packing, let $S = \sum_{i=1}^{b} S_b$ be the sum of S_bs. After each i-plane has been swept once, by the invariant proved above, each corner point has integral coordinates. As the container is in the positive quadrant, S is an integer greater than or equal to zero, now if any i-sweep results in F_t^i being non-empty, the value of S drops by at least one after the sweep. Since S is bounded from below by zero and is a finite number at the start of the algorithm, the algorithm has to terminate.

With this proof, we have also proved that an arbitrary *packing* can be converted into an *integral packing* for boxes with integral dimension vectors. Thus, given any packing, we can constructively convert it into an integral perfect packing without using extra volume. For the purpose of our proofs it is enough to know that an integral perfect packing exists that is also optimal. □

As an easy consequence of the above we have the following Lemma.

Lemma 2. *There is an integral perfect packing for the input boxes, such that the volume of the container in which this packing is done is minimized.*

Now we are ready to design an FPT algorithm for d-p-Volume Packing. To illustrate the idea, we first show that 2-p-Volume Packing is FPT.

Theorem 1. 2-p-Volume Packing *is FPT with running time* $(2 \cdot \sqrt{2})^k \cdot k^{\mathcal{O}(1)}$.

Proof. Let the input set of rectangles be represented using the set $B = \{b_i = (w_i, h_i) | i \in [n]\}$, with integers w_i and h_i being the width and height of the rectangle b_i respectively. Each rectangle contributes an area of at least one. Therefore, if k is less than the number of rectangles we can safely output NO. Thus, in the rest of the proof, we assume that $n \leq k$.

Fixing a container: First, we enumerate the dimensions of a rectangle having area k. For each choice, say $k_1 \times k_2$, of the dimensions of a potential container C, we do the following. We assume that C is kept on the co-ordinate plane with lower-left corner at the origin, each point having integral co-ordinates inside C is referred to as a grid point. We will construct a set of *tilings* of the container C using rectangles. For each tiling, we try to fit each rectangle of B inside some rectangle of the tiling. The key observation is that any integral perfect packing of C using B can be extended into a tiling by introducing some extra 1×1 rectangles in B used to fill in the unoccupied area in a packing. By Lemma 1 we can assume that there is an optimal packing that is integral and perfect. also, Lemma 2 implies that a container with optimal area will be such that k_1 and k_2 are integers. Thus, there are at most k choices of containers, for which we enumerate a set of integral tilings and try to derive an optimal integral perfect packing from at least one of the tilings.

Grid points in a tiling: A grid point is referred to as an interior point if it is not contained on any edge of C. For any tiling of a rectangular container, there are a total of eight ways in which the edges of the participating rectangles, or tiles, may pass through an interior grid point of a container in a tiling. We refer to each configuration of edges at a grid point as a *meeting*. Fig. 1 illustrates the *meetings* for an interior grid point. The *meeting* 1(a) corresponds to a corner point of four tiling rectangles. The *meeting* 1(b) corresponds to a grid point contained on the edges of two adjacent rectangular *tiles*. The

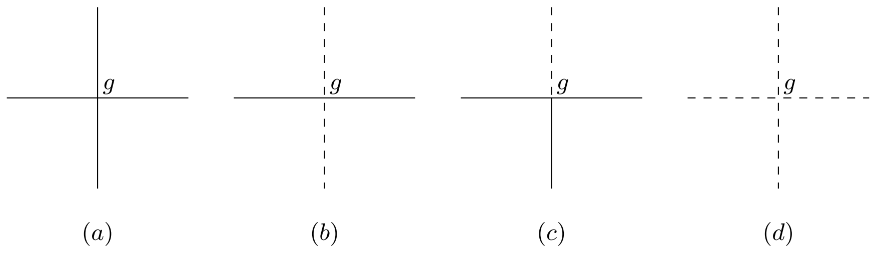

Fig. 1. The 8 possible meetings of a two-dimensional interior grid point g: (a) all solid lines, (b) there is one other meeting obtained by rotation through 180°, (c) there are three other meetings obtained by rotation through multiples of 90° and (d) all dotted lines.

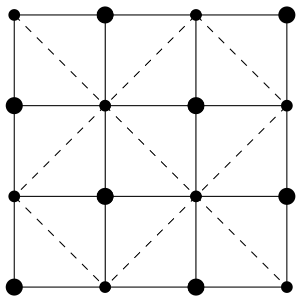

Fig. 2. An R_C grid and its m-grid, the lower left point is the origin. The solid lines are grid lines. The dashed lines represent the m-grid lines. The large black vertices represent the internal grid points of each m-cell.

meeting 1(c) corresponds to a grid point which is the corner point of two *tiles* and contained on the edge of another tiling rectangle. The *meeting* 1(d) corresponds to a grid point contained inside a tiling rectangle. The *meeting* corresponding to a corner point of a container consists of two unit length solid lines intersecting at 90°. There are exactly two *meetings* which correspond to a non-corner edge point of a container. The part of a *meeting* on the container edge must be solid, while the unit segment perpendicular to it at the center can be either dotted or solid.

For any *meeting* M, the grid point will be denoted as M_p and will be called a *meeting point*. There are exactly four line segments incident to a meeting point. The lines incident on M_p are of two types, dotted or solid. The solid line segments correspond to the edges of the tiles that have M_p as a corner point. The dotted line segments denotes tiles where M_p is not a corner point. There can be at most 4 tiling rectangles that can intersect at M_p.

Generating a Configuration: We denote the dimensions of the container by $k_1 \times k_2$ for a rectangular grid R_C having an area of k units. The grid R_C is assumed to be contained in the co-ordinate plane with lower left corner at the origin. Each point in it is in natural correspondence with the grid points of the actual container. For any two points u and v in R_C at a distance of one unit from each other, the axis-parallel line segment uv of unit length is referred to an edge. We say that two points u, v in R_C are adjacent if they are connected by an edge. Note that the line segment uv does not contain any other grid point. A cell in R_C is defined as a minimal square containing exactly four grid points at its corners. By minimality, there is no grid point contained inside a cell of R_C. Two cells in R_C are said to be adjacent if they share an edge.

A configuration of R_C is obtained by associating a solid or a dotted line with each edge contained in it. Consider the set of at most $(2 \cdot \sqrt{2})^k$ configurations constructed as follows. Firstly, we construct a new grid, referred to as an *m-grid* on top of the old grid points in R_C as follows. Draw a line passing through the point $(1, 0)$ at 45° to the x-axis in the first quadrant and draw lines parallel to it spaced regularly at a distance of $\sqrt{2}$ from each other. We call the family of lines as $\mathcal{L}$. Similarly, draw a line passing through the point $(1, 0)$, but now at 135° to the x-axis. Then draw lines parallel to it spaced regularly at a distance of $\sqrt{2}$ from each other. We call this family of lines $\mathcal{L}_\perp$. The *m-grid* consists of the lines of $\mathcal{L}$ and $\mathcal{L}_\perp$ restricted to the grid R_C. The intersection points of $\mathcal{L}$ and $\mathcal{L}_\perp$ define the grid points of *m-grid*. We analogously refer to a cell in the *m-grid* as an *m-cell*, it has an area of 2 units. For clarity it is depicted in Fig. 2. Associate a meeting with each of the internal grid point in R_C not contained on the *m-grid* i.e. with the *m-cells* contained completely inside R_C. Observe that this associates a solid or a dotted line with each of the edges in R_C, except some edges contained in half-*m-cells*, with centers lying on the boundary of R_C. In a tiling configuration, the boundary of R_C is always selected i.e. it is solid. So we only need to make a choice for one line in each half-*m-cell*, with its center at the boundary. Take two half-*m-cells* and observe that there are at most 4 choices for the undetermined edges in them. Finally, the quarter-m-cells can occur only at the corner of R_C and both their edges are determined (solid). The number of *m-cells* in R_C is $\lfloor \frac{k}{2} \rfloor$, thus the total number of configurations is at most $8^{\frac{k}{2}}$ which is $(2 \cdot \sqrt{2})^k$.

Obtaining a tiling: Let $\mathcal{R}_C$ be a configuration. We call two adjacent cells in $\mathcal{R}_C$ *connected* if they both share a dotted edge. We construct an undirected graph G_C with the cells in $\mathcal{R}_C$ being its vertex set. For any two vertices $u, v \in V(G_C)$, (u, v) is an edge in G_C if the cells corresponding to u and v are *connected*. For each connected component in G_C, we check if the cells corresponding to the connected component in G_C form a rectangle in R_C. This can be done in polynomial time. If all the connected components in G_C correspond to a rectangle in R_C then we have generated a *tiling* of C. For a tiling $\mathcal{R}_C$, the connected components $C_1, C_2, \dots, C_m$ of G_C correspond to the set of tiles $T = \{t_1, t_2, \dots, t_m\}$. Note that t_i and t_j may have the same dimensions for $i \neq j$.

Obtaining a packing: We say a rectangle a is *compatible* with b if the dimension vector of a and b are the same. Informally, it means that the rectangle a can be translated to exactly cover b. Next, for a tiling $\mathcal{R}_C$, we define an undirected bipartite graph M_C as follows. The graph M_C consists of vertex sets $B \uplus T$, where B corresponding to the input collection of rectangles and T corresponds to the set of tiles in $\mathcal{R}_C$. For $b \in B$ and $t \in T$ there is an edge in M_C if b is *compatible* with t. Finally, a tiling T of $\mathcal{R}_C$ contains a packing of B if and only if there exists a matching in M_C which saturates B.

Proof of correctness: By Lemma 2, there is a container with optimal area and an integral dimension vector. If the rectangles in B can be packed inside some container of area k, there is a container C^* with dimensions $c_1 \times c_2$, with $c_1, c_2 \in \mathbb{N}$, and $c_1 \cdot c_2 = k$. This choice of dimensions will be generated by the first step of our algorithm. By Lemma 1, for a YES instance, there is an integral packing in a container of minimum area. As we are enumerating all possible ways in which a rectangle edge may pass through a grid point, we effectively enumerate all tilings of a fixed container such that the tiling is an integral packing of the participating tiles. If the given input instance is a YES instance, there is an optimal integral packing P^* can be extended to a tiling T^*, with T as the participating set of tiles and where the corners of each tile has integral co-ordinates: this can be done by adding sufficiently many 1×1 tiles to cover the gaps. The bipartite graph on B and T will have a matching saturating B, since P^* is contained in T^*. On the other hand, if there is a container C^* and a tiling T^* such that there is a matching saturating B in the bipartite graph M_{T^*}, then the matching gives us a packing of B in the container C^*. This shows that the given instance is a YES instance.

Running time: Except for the step of generating configurations, all other steps of the algorithm run in polynomial time (recall that finding a matching in a bipartite graph is polynomial time solvable). The set of configurations that are generated take $(2 \cdot \sqrt{2})^k \cdot k^{\mathcal{O}(1)}$ time, for a fixed container. Therefore, the running time of the algorithm is $(2 \cdot \sqrt{2})^k \cdot k^{\mathcal{O}(1)}$. □

Theorem 1 can be generalized to higher dimensions. To get dimension vectors for a possible optimal container we try all factorizations of k with d factors in it. To avoid any ambiguity, we redefine the analogues of structures defined in Theorem 1. A cell in d-dimensions is a minimal d-dimensional hypercube with unit side length and containing 2^d grid points, all of which are corner points. A face of a cell is a $d-1$ dimensional hyper-surface bounding it (corresponding to a rectangle edge in 2-dimensions). To generate a configuration in d-dimensions for $d \geq 2$ we assign each face to be either a solid or a dotted face. The number of faces bounding a d-dimensional cell is $2d$. For a tiling to be valid each boundary face of container is assigned a solid face. To bound the number of internal faces, observe that there are k unit cells in a given bounding box and each face is shared by two of the cells. Hence the number of faces is upper bounded by $k \cdot d$, this gives us a bound of at most $2^{k \cdot d}$ configurations.

Analogous to proof of Theorem 1, we can extract a tiling T from a configuration and check if it gives us a packing. We get the following result.

Theorem 2. *d-p-Volume-Packing for a set of rectangles in $\mathbb{N}^d$ is FPT parameterized by d and k. The running time of the algorithm is $2^{k \cdot d} \cdot k^{\mathcal{O}(d)}$.*

Theorem 1 gives us the following structural property, which will be useful later.

Corollary 1. *Given a d-dimensional container with volume k, there are at most $2^{k \cdot d}$ configurations, and therefore at most $2^{k \cdot d}$ tilings such that for each tile the corner points have integral coordinates. These configurations and tilings can also be found in $2^{k \cdot d} \cdot k^{\mathcal{O}(d)}$ time. For $d = 2$, the number of such configurations and tilings is at most $(2\sqrt{2})^k$ and can be found in time $(2\sqrt{2})^k \cdot k^{\mathcal{O}(1)}$.*

4. Strip Packing

In this section, we will study the parameterized complexity of Strip Packing problem. First we show that Strip Packing in its general form is W[1]-hard, even when the input is represented in unary form.

Lemma 3. *d-p-Strip Packing is* W[1]*-hard for $d \geq 2$.*

Proof. We prove the result by giving a reduction from Bin Packing problem to the d-p-Strip Packing problem.

An instance of the Bin Packing problem takes in as input a set of n objects $a_1, a_2, \ldots, a_n$, with each object a_i weighing w_i, and positive integers W and k. The goal is to decide whether all the input objects can be fit into at most k bins, each of capacity W. It was shown in [13] that Bin Packing is W[1]-hard, even when the input is represented in unary form.

Let $(\{a_1, a_2, \ldots, a_n\}, \{w_1, w_2, \ldots, w_n\}, W, k)$ be an instance of Bin Packing. We construct an instance of d-p-Strip Packing as follows:

1. First we construct a vector $W' \in \mathbb{N}^{d-1}$. The vector has W in its first index and 1 at all other indices.
2. Next, we construct a set of n boxes. For each object a_i, we create a box b_i whose dimension vector in $\mathbb{N}^d$ has w_i in the first index, and 1 in all other indices.

This completes the construction of our reduced instance. We claim that the Bin Packing is a YES instance if and only if the reduced instance of d-p-Strip Packing is a YES instance.

First, suppose the given instance of Bin Packing is a YES instance. This means that at most k bins of capacity W are sufficient to pack all the input objects. In the d-p-Strip Packing instance, we construct the following arrangement for the input boxes. For $1 \le j \le k$, let $\{a_{i_1}, a_{i_2}, \ldots, a_{i_j}\}$ be the set of objects that are placed in the jth bin. Then, for each $1 \le \ell \le i_j$, we place the box b_ℓ such that the left-most bottom-most coordinate is at $(\Sigma_{1 \le \ell' < \ell} w_{\ell'}, 1^{d-2}, j-1)$. By the construction, for a bin j, all the boxes, corresponding to the objects in bin j, can be placed without overlap in such a way that their right-most top-most coordinate is at $(\Sigma_{1 \le \ell' \le \ell} w_{\ell'}, 1^{d-2}, j)$. Also, when $j_1 \neq j_2$, then a box corresponding to an object in bin j_1 does not overlap with a box corresponding to an object in bin j_2. This means that all boxes can be packed within a height of k of the given container, whose left-most bottom-most point is placed at the origin.

In the reverse direction, let the constructed d-p-Strip Packing instance be a YES instance. By Lemma 1, we know that there is a witnessing packing such that all the input boxes are placed in integral coordinated in an arrangement that requires the height of the box to be at most k. Since the height of each input box is 1, an integral arrangement would mean that the last coordinate for the base and the top of each box is an integer. Thus, the container of dimension $W' \times k$ can be divided into k smaller containers, each of dimension $W' \times 1$. Let $\{b_{i_1}, b_{i_2}, \ldots, b_{i_j}\}$ be the boxes in the jth subcontainer. Then, we place the objects $\{a_{i_1}, a_{i_2}, \ldots, a_{i_j}\}$ in the jth bin for the Bin Packing instance. By construction, in each bin, the sum of weights of objects packed is at most W. Therefore, we require at most k bins to pack all given objects, thereby showing that the given Bin Packing instance is a YES instance.

This completes the proof of W[1]-hardness of d-p-Strip Packing. □

Now we consider a special case where Strip Packing becomes FPT. This special case is motivated by the variant of the Bin Packing problem, where the number of types of weights is bounded by a constant. Determining the complexity of this problem was a long standing open problem. In [9], the problem was shown to be polynomial time solvable. We study the case where the dimension of each rectangle is bounded by a constant ℓ. This also implies that the types of rectangles are bounded by ℓ^2.

Lemma 4. 2-Strip Packing *is FPT when the dimensions of the input rectangles belong to a set S such that $\ell = \max_{a \in S} a$ is a constant. The running time of the algorithm is $8^{k\ell} \cdot (n^{\mathcal{O}(\ell^2)} W)$.*

Proof. There are at most ℓ^2 shapes of boxes that have their dimensions in the set S. Assume that among the n input boxes, n_i boxes of shape $i \le \ell^2$ are there. Let this be denoted by the vector $(n_1, n_2, \ldots, n_{\ell^2})$.

Consider a packing of rectangles in a strip of width W and height k. Assume this is a perfect packing. We will consider vertical layers of fixed width in this packing. The first layer consists of all rectangles whose right edge is at most ℓ distance away from the left boundary of the strip. The second layer consists of rectangles whose right edge is at a distance of at most ℓ from the rightmost point of the first layer and so on. Since this is a perfect packing, for each layer, there is a bounding box of dimension $2\ell \times k$. Note that a bounding box can intersect with rectangles in the previous and next layers also. The packing contains at most W such layers. Moreover, by the bound on the number of configurations given in Corollary 1, the number of possibilities for the pattern for each layer is a $8^{k\ell}$. Let $\mathcal{P}$ be the set of all $8^{k\ell}$ such patterns. We build a directed graph, G as follows. Each vertex v in $V(G)$ corresponds to a pattern P_v and there exists a directed edge (u, v), from u to v, if P_u can be immediately followed by P_v i.e., for each rectangle $r_1 \in P_v$ whose left edge intersects with the left boundary of P_v there is a rectangle $r_2 \in P_u$ whose the right boundary intersects with the right boundary of P_u and the left boundary of r_1. The weight of edge (u, v) is a vector $((t_1^v, t_2^v, \ldots, t_{\ell^2}^v), \mathsf{dist}_{uv})$, where t_i^v is the number of boxes of type i in the pattern P_v, and dist_{uv} is the distance (l_1 norm in X axis) between the rightmost point on the right boundary of P_u and the rightmost point on the right boundary of P_v. Now, suppose there is a walk in the graph G of total weight $((N_1, N_2, \ldots, N_{\ell^2}), W')$, such that for each i, $N_i \ge n_i$ and $W' \le W$. Then this corresponds to an arrangement of boxes in a container of dimension $W \times k$. On the other hand, if there is an arrangement of boxes in a container of dimension $W \times k$, we can always construct a walk in G that has weight $((N_1, N_2, \ldots, N_{\ell^2}), W')$, such that for each i, $N_i \ge n_i$ and $W' \le W$.

What remains is to find such a walk if it exists, or correctly detect that the instance is a NO instance. For this, we design a dynamic programming algorithm. For each vector $(M_1, M_2, \ldots M_{\ell^2})$, where each $M_i \le n_i$, for each $W' \le W$ and for each pattern P_v, we want to determine whether there is a walk of weight $((M_1, M_2, \dots M_{\ell^2}), W')$ such that the last vertex in the walk is v. In the base case, it is trivial to update entries of walks with weight 1. As an induction hypothesis, let us suppose that for each vector $(M_1, M_2, \dots M_{\ell^2})$, where each $M_i \le n_i$, for each $W'' < W'$ and for each pattern P_v, we have determined whether there is a walk of weight $((M_1, M_2, \dots M_{\ell^2}), W'')$ such that the last vertex in the walk is v. To determine whether there is a walk of weight $((M_1, M_2, \dots M_{\ell^2}), W')$ such that the last vertex in the walk is v, it is enough to determine whether there is a in-neighbor u of v such that there is a walk of weight $((M_1 - t_1^v, M_2 - t_2^v, \dots M_{\ell^2} - t_{\ell^2}^v), W' - \mathsf{dist}_{uv})$ such that the last vertex in the walk is u. By definition, $W' - \mathsf{dist}_{uv} < W'$. Therefore, by induction hypothesis, if such a walk ending at u exists then we have stored the answer correctly. If such a walk exists for an in-neighbor u of v, this also implies that there is a walk of weight $((M_1, M_2, \dots M_{\ell^2}), W')$ such that the last vertex in the walk is v. Hence, in polynomial time, we can update each entry of the table. In the end, we search over all vertices $v \in V(G)$, and all weights $W' \le W$ whether there is a walk of weight $((M_1, M_2, \dots M_{\ell^2}), W')$ such that the last vertex in the walk is v, and for each i, $M_i \ge n_i$.

The size of the table is $n^{\ell^2} \cdot W \cdot 8^{k\ell}$. Each entry takes polynomial time to be updated. In the end, searching for a required walk also needs polynomial time. Thus, we have given an algorithm with running time $8^{k\ell} \cdot (n^{\mathcal{O}(\ell^2)} W)$, which is an FPT algorithm parameterized by k, when ℓ is a constant. □

5. Polynomial time packings

In this section, we look at some special cases, where square boxes of specific types are taken. Moreover, the dimension of a smaller square divides that of a larger square. For these cases, we give combinatorial arguments to show that an optimal packing can be found in polynomial time. First, we consider the case when 1×1 squares are given as input.

Lemma 5. *Given a container whose width is a positive integer W, n_1 boxes of width 1 and height 1, and n_2 boxes of width l and height l, we can determine in polynomial time the minimum height required such that all input boxes can be packed into the given container.*

Proof. We use a greedy approach where all the bigger rectangles are packed first and the remaining space is filled with 1×1 rectangles. n_2 rectangles of dimension $l \times l$ are packed first as tightly as possible with each row (except the top most) containing $\lfloor \frac{W}{l} \rfloor$ rectangles. Next, we pack the 1×1 rectangles starting with any row that is not filled and continuing till we pack all rectangles. We claim that this packing optimizes the height used.

Let k_1 be the height to which $l \times l$ rectangles are packed and k_2 be the height to which 1×1 rectangles are packed. If $k_1 \ge k_2$, then this is an optimum packing since packing only $l \times l$ rectangles needs a height of k_1 in the strip. Otherwise, there is no gap till height $k_2 - 1$. This implies that k_2 is the optimum height since the combined area of the input rectangles, $A = n_1 + n_2 l^2$, is such that $W \cdot (k_2 - 1) < A \le W \cdot k_2$. □

We can generalize Lemma 5 to the following.

Lemma 6. *Given a container of width W, n_1 boxes of width l and height l, and n_2 boxes of width cl and height cl, where c is an integer, we can determine in polynomial time the minimum height required such that all input boxes can be packed into the given container.*

Proof. We use a greedy algorithm as before and pack all the bigger rectangles first and later fill up the gaps with the smaller rectangles. Let k_1 be the height to which $l \times l$ rectangles are packed and k_2 be the height to which $cl \times cl$ rectangles are packed. If in this packing, the rectangles are packed without leaving any gaps upto a height k', then the packing is optimum by reasons given in the proof of Lemma 5. Otherwise, either all the small rectangles are packed in one layer in which case this packing is optimum since k_2 is a lower bound on the height needed for packing all big rectangles. Or the rectangles are packed to a width $W' > W - l$. Consider an optimal packing of the boxes. We show that for any optimal packing, at each integer row of the container, the total width covered by the rectangles intersecting with the row is at least $W - W'$. This is because $W - W'$ is the smallest remainder when dividing W by l. Therefore, for any packing of height k in a container of width W, there exists a packing of height k in a container of width W'. By the arguments of Lemma 5, the packing is optimum for width W'. This implies that the packing is optimum for width W. □

Finally, we can give a polynomial packing for the following case.

Lemma 7. *We are given a container of width W, and integers $a_1 < a_2 \dots . < a_t$ such that for each $i \le t$, a_i is a divisor for all a_j, $j > i$. Let there be n_i boxes of width a_i and height a_i. We can determine in polynomial time the minimum height required such that all the input boxes can be packed into the given container.*

Proof. We give a packing for these squares as follows: We start with the largest available square and place it as low and left as possible. We continue this process till all squares have been packed. We claim that this is an optimal packing for the given set of squares.

We prove the optimality of this packing by induction on t. The base case, when $t = 2$, is true due to Lemma 6. Suppose the algorithm finds an optimal packing when the container has width W, the dimension vectors of the input squares are $\{b_1 \times b_1, b_2 \times b_2, \ldots b_{t-1} \times b_{t-1}\}$, and for each $i \le t-1$, b_i is a divisor for all b_j, $j > i$. Now, we try to give a packing for the input set of squares, having dimensions $a_j \times a_j$, $j \ge 1$. First, by induction hypothesis, we use our algorithm to pack the set of squares with dimension vectors $a_i \times a_i$, $i > 1$. There are $t-1$ distinct dimension vectors. Therefore, our algorithm gives an optimal packing of these squares. Since this is an optimal packing of these squares, the height k_1 required for this packing is a lower bound for any optimal packing of the original set of squares. Now we try to add the squares of dimension $a_1 \times a_1$ according to the algorithm. Suppose the height taken after packing the $a_1 \times a_1$ squares is k_1 then we know that this is an optimal packing. Otherwise, let the height required be $k > k_1$. Except for the top row (which contains only squares of size $a_1 \times a_1$), all other rows incur a gap which is strictly less that a_1. As argued in Lemma 6, for any packing of the input boxes, all rows except the top row incur at least as much gap as our packing. Let the sum of all the gaps in all but the top row be G. By arguments similar to Lemma 5, the total sum of the areas of the squares and G is strictly greater than $W(k-1)$. Therefore, k is the optimum height. □

6. Conclusion

In this paper, we study Strip Packing and Volume Packing in the parameterized setting, when the inputs are boxes with integral dimension vectors, and the containers are also boxes with integral dimension vectors. We show that d-p-Volume Packing is FPT, but d-Strip Packing is W[1]-hard. Because of extensive applications of these problems, it is also interesting to study special variants of these problems. We studied 2-Strip Packing with bounded dimensions. This problem is motivated from the study of Bin Packing with bounded types. An open question is regarding the status of 2-Strip Packing with bounded types. In other words, if the number of types of input boxes was a constant, then what is the complexity of the problem. The same can be asked for d-Strip Packing with bounded types. Lastly, we saw some polynomial time optimal packings for some special cases of Strip Packing where the input boxes are squares with dimension vectors satisfying certain constraints. It would be interesting to study these packing problems when the input boxes are arbitrary squares.

References

[1] H. Alt, M. Berg, C. Knauer, Approximating minimum-area rectangular and convex containers for packing convex polygons, in: Algorithms – ESA 2015: Proceedings of the 23rd Annual European Symposium, Patras, Greece, September 14–16, 2015, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 25–34.
[2] H. Alt, N. Scharf, Approximating smallest containers for packing three-dimensional convex objects, CoRR arXiv:1601.04585, 2016.
[3] J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Oper. Res. 33 (1) (1985) 49–64.
[4] A. Brown, Optimum Packing and Depletion: The Computer in Space – And Resource-Usage Problems, Computer Monographs, vol. 14, MacDonald, 1971.
[5] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: a survey, in: Approximation Algorithms for NP-Hard Problems, PWS Publishing Co., Boston, MA, USA, 1997, pp. 46–93.
[6] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[7] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer-Verlag, 1999, 530 pp.
[8] J. Flum, M. Grohe, Parameterized Complexity Theory, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[9] M.X. Goemans, T. Rothvoß, Polynomiality for bin packing with a constant number of item types, in: Proceedings of the Twenty-Fifth Annual ACM–SIAM Symposium on Discrete Algorithms, SODA '14, SIAM, 2014, pp. 830–839.
[10] R. Harren, W. Kern, Improved lower bound for online strip packing, Theory Comput. Syst. 56 (1) (2015) 41–72.
[11] E. Huang, R.E. Korf, Optimal rectangle packing: an absolute placement approach, J. Artificial Intelligence Res. 46 (1) (January 2013) 47–87.
[12] C. Imreh, Online strip packing with modifiable boxes, Oper. Res. Lett. 29 (2) (2001) 79–85.
[13] K. Jansen, S. Kratsch, D. Marx, I. Schlotter, Bin packing with fixed number of bins revisited, J. Comput. System Sci. 79 (1) (2013) 39–49.
[14] C. Kenyon, E. Rémila, A near-optimal solution to a two-dimensional cutting stock problem, Math. Oper. Res. 25 (4) (November 2000) 645–656.
[15] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems: a survey, European J. Oper. Res. 141 (2) (2002) 241–252.
[16] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, Vlsi module placement based on rectangle-packing by the sequence-pair, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 15 (12) (December 1996) 1518–1524.
[17] Y. Xia, M. Chrzanowska-Jeske, B. Wang, M. Jeske, Using a distributed rectangle bin-packing approach for core-based soc test scheduling with power constraints, in: International Conference on Computer Aided Design, ICCAD-2003, November 2003, pp. 100–105.
[18] D. Ye, X. Han, G. Zhang, A note on online strip packing, J. Comb. Optim. 17 (4) (2008) 417–423.