APPROXIMATION ALGORITHMS FOR 
MULTIDIMENSIONAL BIN PACKING

A Thesis 
Presented to 
The Academic Faculty

by

Arindam Khan

In Partial Fulfillment 
of the Requirements for the Degree 
Doctor of Philosophy in 
Algorithms, Combinatorics, and Optimization

School of Computer Science 
Georgia Institute of Technology 
August 2015
APPROXIMATION ALGORITHMS FOR
MULTIDIMENSIONAL BIN PACKING

Approved by:

Professor Prasad Tetali, Advisor
School of Mathematics and School of
Computer Science
Georgia Institute of Technology

Professor Santosh Vempala
School of Computer Science
Georgia Institute of Technology

Professor Sebastian Pokutta
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Professor Nikhil Bansal (Reader)
Department of Mathematics and
Computer Science
Eindhoven University of Technology,
Eindhoven, Netherlands

Professor Dana Randall
School of Computer Science
Georgia Institute of Technology

Professor Santanu Dey
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Date Approved: xx August 2015
To Maa, Baba and Bhai
# Contents

**DEDICATION**  

**LIST OF TABLES**  

**LIST OF FIGURES**  

**SUMMARY**

## I  INTRODUCTION

1.1 Contribution of the Thesis  
1.2 Organization of the Thesis

## II  PRELIMINARIES

2.1 Combinatorial Optimization and Complexity:  
2.2 Approximation Algorithms and Inapproximability  
2.3 Relaxation and Rounding  
   2.3.1 Relaxation  
   2.3.2 Relaxation techniques: LP and SDP  
   2.3.3 Rounding of solution of relaxation  
2.4 One Dimensional Bin Packing  
   2.4.1 Offline 1-D Bin Packing:  
   2.4.2 Online 1-D Bin Packing:  
2.5 Multidimensional Bin Packing  
   2.5.1 Geometric Packing  
   2.5.2 Vector Packing  
   2.5.3 Weighted Bipartite Edge Coloring  
   2.5.4 Relation between the problems  
2.6 Techniques:  
   2.6.1 Next Fit Decreasing Height (NFDH)  
   2.6.2 Configuration LP  
   2.6.3 Round and Approx (R&A) Framework
2.6.4 Algorithms based on rounding items to constant number of
types: ................................. 29

III GEOMETRIC BIN PACKING .............................. 32

3.1 Prior Works ........................................... 34
  3.1.1 Geometric Bin Packing ......................... 34
  3.1.2 Square Packing ................................. 35
  3.1.3 Online Packing: ................................. 36
  3.1.4 Heuristics .................................. 37
  3.1.5 Packing with Augmentation .................... 39
  3.1.6 Strip Packing ................................. 39
  3.1.7 Shelf and Guillotine Packing .................. 41
  3.1.8 Geometric Knapsack ......................... 43

3.2 R&A Framework for Rounding Based Algorithms ...... 44

3.3 A rounding based 1.5-approximation algorithm ........ 52
  3.3.1 Technique ................................. 52
  3.3.2 Details of the Jansen-Prädel Algorithm: ........ 54
  3.3.3 Analysis .................................. 62
  3.3.4 Bin packing with rotations .................... 63

3.4 Lower bound for rounding based algorithms ............ 63

3.5 Final Remarks .................................. 67

IV VECTOR BIN PACKING ................................. 68

4.1 Prior Works ........................................... 71
  4.1.1 Offline Results: .......................... 71
  4.1.2 Online Results .............................. 72
  4.1.3 Heuristics ................................. 72

4.2 Preliminaries .................................. 73
  4.2.1 Limitations of Round and Approx Framework .... 75
  4.2.2 Multi-objective/multi-budget matching problem .... 76

4.3 Vector Packing with Resource Augmentation .......... 77
List of Tables

1 Approximation algorithms for one dimensional bin packing . . . . . . 19
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Two rectangles of size $\frac{1}{2} \times \frac{3}{4}$ and $\frac{1}{2} \times \frac{3}{4}$ can be packed into one bin</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>Two vectors $(\frac{1}{2}, \frac{3}{4})$ and $(\frac{1}{4}, \frac{3}{4})$ cannot be packed into one vector bin as their sum exceeds one in second dimension</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>Generalizations of bin packing problems</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Example of two stage packing</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>Example of guillotine packing</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Example of non-guillotine packing</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>Lower bound example for rounding based algorithms</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>The case when $C_{i+1} &gt; \frac{1}{2}$</td>
<td>64</td>
</tr>
<tr>
<td>9</td>
<td>Configuration ${P, P, S}$</td>
<td>65</td>
</tr>
<tr>
<td>10</td>
<td>Configurations ${R, R, Q}$</td>
<td>65</td>
</tr>
<tr>
<td>11</td>
<td>A Clos network with $m = 2, \mu = 3, r = 4$ [54]</td>
<td>115</td>
</tr>
<tr>
<td>12</td>
<td>Algorithm for Edge Coloring Weighted Bipartite Graphs</td>
<td>123</td>
</tr>
</tbody>
</table>
SUMMARY

The bin packing problem has been the cornerstone of approximation algorithms and has been extensively studied starting from the early seventies. In the classical bin packing problem, we are given a list of real numbers in the range $(0, 1]$, the goal is to place them in a minimum number of bins so that no bin holds numbers summing to more than 1. In this thesis we study approximation algorithms for three generalizations of bin packing: geometric bin packing, vector bin packing and weighted bipartite edge coloring.

In two-dimensional (2-D) geometric bin packing, we are given a collection of rectangular items to be packed into a minimum number of unit size square bins. Geometric packing has vast applications in cutting stock, vehicle loading, pallet packing, memory allocation and several other logistics and robotics related problems. We consider the widely studied orthogonal packing case, where the items must be placed in the bin such that their sides are parallel to the sides of the bin. Here two variants are usually studied, (i) where the items cannot be rotated, and (ii) they can be rotated by 90 degrees. We give a polynomial time algorithm with an asymptotic approximation ratio of $\ln(1.5) + 1 \approx 1.405$ for the versions with and without rotations. We have also shown the limitations of rounding based algorithms, ubiquitous in bin packing algorithms. We have shown that any algorithm that rounds at least one side of each large item to some number in a constant size collection values chosen independent of the problem instance, cannot achieve an asymptotic approximation ratio better than $3/2$.

In $d$-dimensional vector bin packing (VBP), each item is a $d$-dimensional vector that need to be packed into unit vector bins. The problem is of great significance
in resource constrained scheduling and appeared also recently in virtual machine placement in cloud computing. Even in two dimensions, it has novel applications in layout design, logistics, loading and scheduling problems. We give a polynomial time algorithm with an asymptotic approximation ratio of $\ln(1.5) + 1 \approx 1.405$ for 2-D VBP. For $d$ dimensions we give a polynomial time algorithm with an asymptotic approximation ratio of $\ln(d/2) + 1.5 \approx \ln d + 0.81$. We also consider vector bin packing under resource augmentation. We give a polynomial time algorithm that packs vectors in $(1 + \epsilon)\text{Opt}$ bins when we allow augmentation in $(d - 1)$ dimensions and $\text{Opt}$ is the minimum number of bins needed to pack the vector into $(1, 1)$ bins.

In weighted bipartite edge coloring problem, we are given an edge-weighted bipartite graph $G = (V, E)$ with weights $w : E \to [0, 1]$. The task is to find a proper weighted coloring of the edges with as few colors as possible. An edge coloring of the weighted graph is called a proper weighted coloring if the sum of the weights of the edges incident to a vertex of any color is at most one. This problem is motivated by rearrangeability of 3-stage Clos networks which is very useful in various applications in interconnected networks and routing. We show a polynomial time approximation algorithm that returns a proper weighted coloring with at most $\lceil 2.2223m \rceil$ colors where $m$ is the minimum number of unit sized bins needed to pack the weight of all edges incident at any vertex. We also show that if all edge weights are $> 1/4$ then $\lceil 2.2m \rceil$ colors are sufficient.
Chapter I

INTRODUCTION

The bin packing problem has been the corner stone of approximation algorithms and has been extensively studied starting from the early seventies. In the classical bin packing problem, we are given a list $I = \{i_1, i_2, \ldots, i_n\}$ of real numbers in the range $(0, 1]$, the goal is to place them in a minimum number of bins so that no bin holds numbers summing to more than 1.

Bin packing is a special case of the one-dimensional cutting stock problem [80], loading problem [56] and several scheduling related problems [38]. In theoretical computer science, the bin packing problem was probably first studied by Garey, Graham and Ullman in 1972 [75] from the standpoint of memory allocation problems such as table formatting, prepaging and file allocation. They noticed that finding a general placement algorithm for attaining the minimum number of bins appears to be impractical, and thus provided four heuristics: first fit (FF), best fit (BF), first fit decreasing height (FFDH) and best fit decreasing heights (BFDH). Soon Johnson, Demers, Ullman, Garey and Graham [115] published the first definitive analysis of the worst case guarantees of several bin packing approximation algorithms. The problem is well-known to be NP-hard [78] and the seminal work of Johnson et al. initiated an extremely rich research area in approximation algorithms [98]. In fact the term approximation algorithm was coined by David S. Johnson [114] in an influential and prescient paper in 1974 where he studied algorithms for bin packing and other packing and covering related optimization problems.

Bin packing is extremely useful in practice and has a lot of applications in various fields. Skiena [173] has presented market research for the field of combinatorial
optimization and algorithms, attempting to determine which algorithmic problems are most in demand for applications by studying WWW traffic. Both bin packing and related knapsack problem were among top five most popular \textbf{NP}-hard problems. The implementations of bin packing and knapsack were the most needed among all \textbf{NP}-hard problems, even more than problems such as set-cover, traveling salesman and graph-coloring.

Garey and Johnson [79], followed by Coffman et al. [37] gave comprehensive surveys on bin packing algorithms. Coffman and Lueker also covered probabilistic analyses of packing algorithms in detail [35]. Galambos and Woeginger [73] gave an overview restricted mainly to online variants of bin packing problems. There had been many surveys on bin packing problems thereafter [84, 34, 45]. The most recent, extensive coverage on 1-D bin packing was given by Coffman et al. [36].

In this thesis we primarily focus on packing in higher dimensions due to its prominence in many real world applications. These generalizations of bin packing also help us to understand the power and limitations of existing algorithmic techniques. An ambitious goal is to translate insights from these classical problems to general results for other related combinatorial optimization problems.

We primarily consider three generalizations of bin packing: geometric bin packing, vector bin packing and weighted bipartite edge coloring.

In two-dimensional (2-D) \textit{geometric bin packing} (GBP), we are given a collection of rectangular items to be packed into a minimum number of unit-size square bins. This variant and other higher dimensional GBP variants have vast applications in cutting stock, vehicle loading, pallet packing, memory allocation and several other logistics and robotics related problems [80, 158]. In two dimensions, packing objects into containers have many important applications, e.g., in the context of cutting out a given set of patterns from a given large piece of material minimizing waste, typically in sheet metal processing and apparel fabrication. In three dimensions, these
problems are frequently encountered in minimizing storage space or container space for transportation. In this thesis we consider the widely studied orthogonal packing case, where the items must be placed in the bin such that their sides are parallel to the sides of the bin. In any feasible solution, items are not allowed to overlap. Here two variants are usually studied, (i) where the items cannot be rotated (packing by translations), and (ii) they can be rotated by 90 degrees (packing by restricted rigid motions). These variants are also recurrent in practice, e.g., in apparel production usually there are patterns of weaving or texture on the material so that the position where a piece should be cut cannot be rotated arbitrarily.

In $d$-dimensional vector bin packing (VBP), each item is a $d$-dimensional vector that needs to be packed into unit vector bins. The problem is of great significance in resource constrained scheduling and appeared also recently in virtual machine placement in cloud computing [152]. For example, consider each job (item) has multiple resource requirements (dimensions) such as CPU, memory, I/O, disk, network etc. and each server (bin) has a bounded amount of these resources. The goal to assign all jobs to minimum number of servers, without violating the resource constraints, translates to the vector packing problem. Even in two dimensions, vector packing has many novel applications in layout design, logistics, loading and scheduling problems [163, 175].

In the weighted bipartite edge coloring problem, we are given an edge-weighted bipartite graph $G = (V, E)$ with weights $w : E \rightarrow [0, 1]$. The task is to find a proper weighted coloring of the edges with as few colors as possible. An edge coloring of the weighted graph is called a proper weighted coloring, if the sum of the weights of the edges incident to a vertex of any color is at most one. This problem is motivated by rearrangeability of 3-stage Clos networks which is very useful in various applications in interconnected networks and routing [113, 102]. This problem is a generalization of two classical optimization problems: bin packing and bipartite edge coloring problem.
These generalizations have been well studied since the 1970s. Baker, Coffman, and Rivest first considered orthogonal packings in two dimensions [7]. At the same time Coffman et al. [116] gave performance bounds for level-oriented two-dimensional packing algorithms such as Next Fit Decreasing Height and First Fit Decreasing Height. Lodi, Martello and Monaci first gave a survey on two-dimensional packing problems [136]. Epstein and van Stee gave a survey in [84] on multi-dimensional bin packing. There has been consistent progress in the area since then. We will provide a detailed survey of prior works in the later corresponding chapters.

1.1 Contribution of the Thesis

The dissertation obtains improved approximation algorithms for the three generalizations of bin packing problems, mentioned above. We summarize the contributions below.

**Geometric Bin Packing:** We give a polynomial time algorithm with an asymptotic approximation ratio of $\ln(1.5) + 1 \approx 1.405$ for 2-D GBP. This holds both for the versions with and without rotations.

The main idea behind this result is to show that the *Round and Approx* (R&A) framework introduced by [11] (See section 2.6.3) can be applied to the result of Jansen and Prädel [104]. Roughly speaking, this framework states that given a packing problem, if (i) the configuration LP for the problem (with the original item sizes) can be solved up to error $1 + \epsilon$ for any $\epsilon > 0$, and (ii) there is a $\rho$-approximation for the problem that is *subset-oblivious* (See section 2.6.3 for a formal description) ; then one can obtain a $(1 + \ln \rho)$-asymptotic approximation for the problem.

In [11], it was shown that the APTAS for 1-D BP due to [47] and the 2-D BP algorithm of [21] are subset-oblivious. However, the notion of subset-obliviousness as defined in [11] is based on various properties of dual-weighting functions, making it somewhat tedious to apply and also limited in scope (e.g. it is unclear to us how to
We give a more general argument to apply the R&A framework directly to a wide class of algorithms\(^1\), and without any reference to dual-weighting functions. In particular, we show that any algorithm based on rounding the (large) items into \(O(1)\) types, is subset-oblivious. The main observation is that any \(\rho\)-approximation based on rounding the item sizes can be related to another configuration LP (on rounded item sizes) whose solution is no worse than \(\rho\) times the optimum solution. As the item sizes are rounded, there are only \(O(1)\) constraints in this LP and it can be easily shown to be subset-oblivious.

For the particular case of 2-D BP, we present the algorithm of Jansen and Prädel that directly fits in the above framework. As most algorithms for bin-packing problems are based on rounding into \(O(1)\) types, this also makes the framework widely applicable. For example, this gives much simpler proofs of all the results in [11].

Finally, we give some results to show the limitations of rounding based algorithms in obtaining better approximation ratios. Rounding of items to \(O(1)\) types has been used implicitly [14] or explicitly [47, 121, 21, 104, 120], in almost all bin packing algorithms. There are typically two types of rounding: either the size of an item in some coordinate (such as width or height) is rounded up in an instance-oblivious way (e.g. in Harmonic rounding [131, 21], or rounding sizes to geometric powers [121]), or it is rounded up in an input sensitive way (e.g. in linear grouping [47]). We show that any rounding based algorithm that rounds at least one side of each large item to some number in a constant-size collection values chosen independent of problem instance (let us call such rounding \emph{input-agnostic}), cannot have an approximation ratio better than \(3/2\).

These results are based on joint work with Nikhil Bansal.

\(^{1}\)This includes all known algorithms that we know of for bin-packing type problems, except the ones based on R&A method.
Vector Bin Packing: Our main result is improved approximation for multidimensional vector packing. In particular we give a polynomial time algorithm with an asymptotic approximation ratio of $(1 + \ln(1.5) + \epsilon) \approx (1.405 + \epsilon)$ for 2-D vector packing and a polynomial time algorithm with an asymptotic approximation ratio of $(1.5 + \ln(d/2) + \frac{3}{d} + \epsilon)$ for $d$-D vector packing.

The main idea behind these algorithms is to use R & A framework along with a structural property of vector packing and multi-objective multi-budget matching. We show that one can not obtain better than $d$-approximation using constant rounding based algorithms, thus giving a barrier of $(1 + \ln d)$ of present R & A framework. We circumvent this problem for $d$-dimensional vector packing, by showing a structural property of vector packing that any optimal packing of $m$ bins can be transformed into $\lceil \frac{(d+1)m}{2} \rceil$ bins such that the bins are of two types:

1. Either a bin contains at most two big items, or
2. The bin has slack in $d - 1$ dimensions (i.e., for each of these $(d - 1)$ dimensions, the sum of all vectors in the bin is at most $1 - \delta$ for some constant $\delta$ that we will fix later).

Then we show that we can pack all items in type 2 bins almost optimally using resource augmentation. This resource augmented packing itself is an interesting observation. Due to the pathological nature of lower bound examples for bin packing, resource augmentation is well studied. For geometric bin packing similar results were known for resource augmentation packing [12, 15]. We specifically show a polynomial time algorithm for packing vectors into at most $(1 + \epsilon + \frac{\epsilon^2}{2d}) \text{Opt} + 1$ bins with $\epsilon$ resource augmentation in $(d - 1)$ dimensions (i.e., bins have length $(1 + \epsilon)$ in $(d - 1)$ dimensions and 1 in the other dimension), where $\text{Opt}$ denotes minimum number of unit vector bins to pack these vectors.

Then we show that we can approximately separate items in type 1 and type 2 bins using a multi-objective, multi-budget matching algorithm from [29]. As a corollary
this gives tight \((1.5 + \xi)\) absolute approximation guarantee. This is interesting as even for 1-D bin packing, the lower bound for absolute approximation is \(3/2\). For absolute approximation, it has been an interesting open question to reach \(3/2\) absolute approximation for other generalizations of bin packing such as 2-D strip packing \([93]\).

Finally, we combine these structural properties, resource augmented packing and matching algorithm with R & A framework to obtain better asymptotic approximation for 2-D and \(d\)-D vector packing. We believe that these techniques might be useful in many similar packing or covering related problems. Our result on asymptotic approximation for 2-D vector packing also matches the present best asymptotic approximation for 2-D GBP. The result on absolute approximation is better than present best asymptotic approximation for 2-D GBP, which is also interesting to understand the hardness of different variants of bin packing.

These results are based on joint work with Nikhil Bansal and Marek Elias.

**Weighted Bipartite Edge Coloring:** We show a polynomial time approximation algorithm that returns a proper weighted coloring with at most \(\lceil 2.2223m \rceil\) colors where \(m\) is the minimum number of unit-sized bins needed to pack the weight of all edges incident at any vertex.

The algorithm is a combination of König’s Theorem \([127]\) with the greedy algorithm based on a threshold. Though the algorithm is purely combinatorial, the analysis heavily uses configuration LP, dual LP, and other structural properties of bin packing. This makes progress towards the resolution of the conjecture \([32]\) that there is always a proper weighted coloring using at most \(2m - 1\) colors.

We also show \(\lceil 2.2m \rceil\) colors are sufficient when all edge weights are \(> 1/4\). We also give an alternate proof of König’s Theorem using the skew-supermodular theorem (See Section 5.1). Our techniques might be useful in the analysis of other algorithms related to Clos networks.

These results are based on joint work with Mohit Singh.
1.2 Organization of the Thesis

In Chapter 2 we discuss related definitions and techniques for approximation algorithms and bin packing. In Chapter 3 we discuss prior works in geometric bin packing and related techniques. In Chapter 4 we cover vector bin packing. In Chapter 5 we cover weighted bipartite edge coloring. We conclude in Chapter 6 with a list of open problems.
Chapter II

PRELIMINARIES

In this chapter we introduce relevant notation and definitions required to define, analyze and classify bin packing related problems. Additional definitions will be introduced later on as required.

2.1 Combinatorial Optimization and Complexity:

Combinatorial optimization problems are ubiquitous in theory and practice. In a combinatorial optimization problem, the goal is to find a solution that maximizes or minimizes a certain objective value among a discrete set of feasible solutions. Edmonds in his seminal paper [55], advocated that an algorithm is efficient if the number of atomic operations the algorithm takes to return a solution is polynomial in the size of the problem instance. In many cases, a brute force search might take exponential or superpolynomial time in the problem size, and hence, are not considered efficient. Depending on this time complexity, the problems are naturally classified into simple and hard problems. Cook [39] formalized this classification defining the complexity classes $\mathcal{P}$ (the set of languages that can be recognized by Turing machines in deterministic polynomial time) and $\mathcal{NP}$ (the set of languages that can be recognized by Turing machines in nondeterministic polynomial time), and the notion of $\mathcal{NP}$-completeness (A decision problem $\Pi$ is $\mathcal{NP}$-complete or in $\mathcal{NPC}$, if $\Pi \in \mathcal{NP}$ and every problem in $\mathcal{NP}$ is reducible to $\Pi$ in polynomial time). Karp’s seminal work [122] established the pervasive nature of $\mathcal{NP}$-completeness by showing a vast majority of problems in combinatorial optimization are $\mathcal{NP}$-complete. These $\mathcal{NP}$-complete problems do not admit a polynomial time algorithm unless $\mathcal{P} = \mathcal{NP}$. For a detailed introduction to complexity classes, we refer the readers to the book by Arora and Barak [3].
Notion of NP and NP-completeness is defined in terms of decision problems. In this thesis we deal with NP optimization problems.

**Definition 2.1.1.** An NP optimization problem $\Pi$ is a four-tuple $(\mathcal{I}, \mathcal{S}, obj, optimize)$ such that:

- $\mathcal{I}$ is the set of input instances of $\Pi$ and is recognizable in polynomial time.

- For any instance $I \in \mathcal{I}$, $\mathcal{S}(I)$ is the set of feasible solutions for $\mathcal{I}$. Solutions for an instance $I$ are constrained to be polynomially bounded in $|I|$, the size of $I$; i.e., there exists a polynomial $poly$ such that $j \in \mathcal{S}(I)$ implies that $|j| \leq poly(|I|)$. Further, there should exist a polynomial time computable boolean function $f$ such that $f(I, j)$ is true if and only if $j \in \mathcal{S}(I)$.

- For each instance $I$, the objective function $obj$ assigns a positive value to each solution $j \in \mathcal{S}(I)$ and this function is polynomial time computable.

- The parameter $optimize \in \{max, min\}$ specifies whether the objective function should be minimized or maximized.

An NP optimization problem $\mathcal{P}$ is NP-hard if there is a polynomial time algorithm $\mathcal{A}$ for a problem $\mathcal{Q} \in \text{NPC}$, given $\mathcal{A}$ has an oracle access to problem $\mathcal{P}$. In general, completeness and hardness are natural notions associated with complexity classes. A problem is complete for a class if it is a member of the class and all other problems in the class reduce to it under an appropriate notion of reducibility. A problem is hard for a complexity class if all problems in the class reduce to it (it need not be a member of the class). Now let us define some other closely related terms.
Definition 2.1.2. **Pseudo-polynomial time algorithm**: An algorithm runs in pseudo-polynomial time if its running time is polynomial in the numeric value of the input (i.e., polynomial in the size of input if the numeric data are encoded in unary), but is exponential in the length of the input if the numeric data are encoded in binary.

Definition 2.1.3. **Weakly NP-complete**: An NP-complete problem $\mathcal{F}$ called *weakly NP-complete* or *binary NP-complete* if it has a pseudo-polynomial time algorithm.

Definition 2.1.4. **Strongly NP-complete**: An NP-complete problem $\mathcal{F}$ is called *strongly NP-complete* or *unary NP-complete* if it is proven that it cannot be solved by a pseudo-polynomial time algorithm unless $P = NP$. These problems are NP-complete even when their numeric data are encoded in unary.

The strong/weak kinds of NP-hardness are defined analogously.

Definition 2.1.5. **Quasi-polynomial time algorithm**: Quasi-polynomial time algorithms are algorithms which run slower than polynomial time, yet not so slow as to be exponential time. The worst case running time of a quasi-polynomial time algorithm is $2^{O((\log n)^c)}$ for some fixed $c$.

The complexity class $QP$ consists of all problems which have quasi-polynomial time algorithms. It can be defined in terms of $DTIME$ as follows. $QP = \bigcup_{c \in \mathbb{N}} DTIME(2^{O((\log n)^c)})$. Exponential time hypothesis implies that NP-complete problems do not have quasi-polynomial time algorithms.

We have already seen in the previous section that several variants of bin packing problem are indispensable in many practical applications. These problems are strongly NP-hard [77] and do not even admit a pseudo-polynomial time algorithm unless $P = NP$. Thus it is imperative to develop heuristics to cope with their intractability. A large body of work has emerged to deal with intractability by exploiting additional structures:
• Efficient algorithms for problems over special classes of problem instances such as problems restricted to bipartite, planar or bounded tree-width graphs (for graph problems) or constant number of item types (for packing problems).

• Algorithms that are guaranteed to perform well with very high probability when the input is generated from a certain distribution.

• Parameterized Complexity: Here the goal is to design algorithms that are efficient on inputs where the associated parameter is small (For details, see [52]).

However in many real-life scenarios, the inputs are generated from complex processes that make discovering additional structure in them a daunting task. In this thesis we restrict ourselves to approximation algorithms that have provable worst-case guarantees even when there is no or little additional information available about the inputs. We will mention few other related practical heuristics in the relevant chapters.

2.2 Approximation Algorithms and Inapproximability

Approximation Algorithms is an attempt to systematically measure, analyze, compare and improve the performance of heuristics for intractable problems. It gives theoretical insight on how to find fast solutions for practical problems, provides mathematical rigor to study and analyze heuristics and also gives a metric for the difficulty of different discrete optimization problems.

Definition 2.2.1. Approximation ratio: Given an algorithm $\mathcal{A}$ for a minimization problem $\Pi$, the (multiplicative) approximation ratio is:

$$\rho_{\mathcal{A}} = \sup_{I \in \mathcal{I}} \left\{ \frac{\mathcal{A}(I)}{\text{Opt}(I)} \right\},$$

where $\mathcal{A}(I)$ is the value of the solution returned by algorithm $\mathcal{A}$ on instance $I \in \mathcal{I}$ and $\text{Opt}(I)$ is the value of the corresponding optimal solution.
In other words, an algorithm $\mathcal{A}$ for a minimization problem $\Pi$ is called a $\rho$-approximation algorithm if $\mathcal{A}(I) \leq \rho \cdot \text{Opt}(I)$ holds for every instance $I$ of $\Pi$. An algorithm $\mathcal{A}$ for a maximization problem $\Pi$ is called a $\rho$-approximation algorithm if $\mathcal{A}(I) \geq \frac{1}{\rho} \cdot \text{Opt}(I)$ holds for every instance $I$ of $\Pi$. This asymmetry ensures that $\rho \geq 1$ for all approximation algorithms.

In some cases, quality of the heuristic is measured in terms of additive approximation. In other words, an algorithm $\mathcal{A}$ for a minimization problem $\Pi$ is called a $\sigma$-additive approximation algorithm if $\mathcal{A}(I) \leq \text{Opt}(I) + \sigma$ holds for every instance $I$ of $\Pi$. Additive approximation algorithms are rare. Karmarkar-Karp’s algorithm [121] for one-dimensional bin packing is one such example.

For detailed introduction to approximation algorithms, we refer the readers to the books on approximation algorithms [183, 186].

**Definition 2.2.2. Polynomial time approximation scheme (PTAS) :** A problem is said to admit a *polynomial time approximation scheme* (PTAS) if for every constant $\epsilon > 0$, there is a $\text{poly}(n)$-time algorithm with approximation ratio $(1 + \epsilon)$ where $n$ is the size of the input. Here running time can be as bad as $O(n^{f(1/\epsilon)})$ for some function $f$ that depends only on $\epsilon$.

If the running time of PTAS is $O(f(1/\epsilon) \cdot n^c)$ for some function $f$ and a constant $c$ that is independent of $\epsilon$, we call it to be an *efficient polynomial time approximation scheme* (EPTAS).

On the other hand, if the running time of PTAS is polynomial in both $n$ and $1/\epsilon$, it is said to be a *fully polynomial time approximation scheme* (FPTAS).

Assuming $\text{P} \neq \text{NP}$, a PTAS is the best result we can obtain for a strongly NP-hard problem. Already in the 1D case, a simple reduction from the *Partition* problem shows that it is NP-hard to determine whether a set of items can be packed in two bins or not, implying that no approximation better than $3/2$ is possible. However, this
does not rule out the possibility of an $\text{Opt} + 1$ guarantee where $\text{Opt}$ is the number of bins required in the optimal packing. Hence it is insightful to consider the asymptotic approximation ratio.

**Definition 2.2.3. Asymptotic approximation ratio (AAR)**: The asymptotic approximation ratio of an algorithm $\mathcal{A}$ is $\rho$ if the output of the algorithm has value at most $\rho \cdot \text{Opt}(I) + \delta$ for some constant $\delta$, for each instance $I$.

In this context the approximation ratio defined as in Definition 2.2.1, is also called to be the (absolute) approximation ratio. If $\delta = 0$, then $\mathcal{A}$ has (absolute) approximation guarantee $\rho$.

**Definition 2.2.4. Asymptotic PTAS (APTAS)**: A problem is said to admit an asymptotic polynomial time approximation scheme (APTAS) if for every $\epsilon > 0$, there is a poly-time algorithm with asymptotic approximation ratio of $(1 + \epsilon)$.

If the running time of APTAS is polynomial in both $n$ and $1/\epsilon$, it is said to be an asymptotic fully polynomial time approximation scheme (AFPTAS).

Note that NP optimization problems whose decision versions are all polynomial time reducible to each other (due to NP-completeness), behave very differently in their approximability. For example classical bin packing problem admit an APTAS, whereas no polynomial factor approximation is known for the traveling salesman problem. This anamoly is due to the fact that reductions between NP-complete problems preserve polynomial time computability, but not the quality of the approximate solution.

$\text{PTAS}$ is the class of problems that admit polynomial time approximation scheme. On the other hand $\text{APX}$ is the class of problems that have a constant-factor approximation. Clearly $\text{PTAS} \subseteq \text{APX}$. In fact the containment is strict.

**Theorem 2.2.5.** [42] If a problem $\mathcal{F}$ is APX-hard then it does not admit PTAS unless $P = NP$. 

14
Online Algorithms: Bin packing is also one of the key problems in online algorithms. Let us define the notion of a competitive ratio which will be useful when we discuss some related results in online algorithms in later chapters.

Definition 2.2.6. Competitive Ratio: Competitive ratio of an online algorithm \( A \) is the worst case ratio of the solution returned by \( A \) and the best offline algorithm.

Asymptotic competitive ratio is defined analogously.

2.3 Relaxation and Rounding

A plethora of approximation algorithms follow a two-step approach. It considers a relaxation of the original problem, followed by a rounding of the solution of the relaxation. Especially, linear program (LP) and semidefinite program (SDP) relaxations have been instrumental in the design of approximation algorithms and have led to the development of several algorithmic techniques such as deterministic rounding, randomized rounding, primal-dual methods, dual-fitting, iterative methods, entropy-based rounding etc.

2.3.1 Relaxation

The space of feasible solutions of a combinatorial problem is discrete and hence, every combinatorial problem can be reformulated as an optimization problem with integer valued variables, i.e., an integer program (IP). In other words, given an instance \( I \) of a combinatorial optimization problem \( \Pi \), we can encode it as maximizing or minimizing a function of a set of variables (say \( \{x_1, x_2, \ldots, x_n\} \)) that take certain integer values (say \( \{0, 1\} \)) and are required to satisfy a set of constraints specific to the problem. As a polynomial time reformulation, the resulting integer program is also \( \text{NP} \)-hard. By a suitable relaxation, intractable IP is converted to a convex optimization problem that can be solved in polynomial time. Specifically, the relaxation allows the variables to be assigned real numbers or even vectors instead of just integer values. This is called
a relaxation as the relaxation permits more solutions than the original program does. Thus if Opt\((I)\) is the optimal value of the problem with instance \(I\) and Conv\((I)\) is the optimal value of the relaxation, then Conv\((I)\) \(\leq\) Opt\((I)\). However it is not clear, how good an estimate Conv\((I)\) is. *Integrality gap* is a coarse measure of this quality.

**Definition 2.3.1. Integrality Gap:** The integrality gap is the worst case ratio between Opt\((I)\) and Conv\((I)\) over all instances \(I\).

The hard instances for a relaxation, where worst-case ratio is obtained, are called *integrality gap instances*.

### 2.3.2 Relaxation techniques: LP and SDP

**Linear Programs:** A vast majority of approximation algorithms use a specific type of convex relaxation - *linear programming* (LP). A linear program consists of optimizing a linear function over real-valued variables while satisfying certain linear constraints among them. Simplex method and its variants [46] are extensively used to solve linear programs in practice, whereas interior point methods [148] are provably efficient. In this thesis we will use linear programs and their duals extensively.

First let us define primal and dual linear programs. Consider the following minimization problem. We call this to be the *primal* problem.

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{n} c_j x_j \\
\text{subject to :} & \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i \in [m] \\
& \quad x_j \geq 0, \quad j \in [n].
\end{align*}
\]
We get the corresponding dual problem by introducing variable $y_i$ for the $i$'th inequality.

$$\text{maximize} \quad \sum_{i=1}^{m} b_i y_i$$

subject to:

$$\sum_{i=1}^{m} a_{ij} y_i \leq c_j, \quad j \in [n]$$

$$y_i \geq 0, \quad i \in [m].$$

**Theorem 2.3.2. LP-Duality Theorem:** The primal program has finite optimum if and only if its dual has finite optimum. Moreover if $x^* := (x_1^*, x_2^*, \cdots, x_n^*)$ and $y^* := (y_1^*, y_2^*, \cdots, y_m^*)$ are optimal solution for the primal and dual programs respectively, then

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*.$$

Linear programs can be solved in polynomial time even if it has exponentially many constraints provided a polynomial time separation oracle can be constructed, i.e., a polynomial time algorithm that given a point in $\mathbb{R}^n$ (where $n$ is the number of variables in the relaxation) either confirms the point is a feasible solution satisfying all constraints or produces a violated constraint. See [87, 86] for more details on the algorithms to find a solution to an LP, given a separation oracle.

**Semidefinite Programs:** Now we briefly mention semidefinite program (SDP) relaxations where we allow the variables to be vectors with linear constraints on their inner products. SDPs can be optimized to within an error $\epsilon$ in time polynomial in $\ln \frac{1}{\epsilon}$ and the size of the program. Seminal work of Goemans and Williamson [83] obtained 0.878 approximation for MAX – CUT whereas linear programs can not yield better than factor 1/2 [25]. SDPs have been helpful in obtaining improved approximations for constraint satisfaction problems (CSP), vertex ordering, sparsest cut, graph decomposition etc. We refer readers to [157] for more related references. Several increasingly stronger relaxations such as Lovász-Schrijver, Lasserre and Sherali-Adams...
hierarchies [130] are also used to strengthen a convex relaxation. However it is not clear if we can use SDPs or hierarchies to get better approximation for variants of bin packing problems.

2.3.3 Rounding of solution of relaxation

For some relaxations, the optimal solution is always integral. Total Dual Integrality and Total unimodularity [165] were developed as general techniques to show the integrality of linear programming formulations. However in general, the optimal solution to the relaxation might not be integral. Thus rounding scheme is devised to round the real (for LP) or vector (for SDP) valued solution to an integral solution incurring little loss in the objective value, say at most losing an $\alpha$-factor in the value. As $\text{Conv}(I) \leq \text{Opt}(I)$, the value of the rounded solution is at most $\alpha$-times the value of the optimal solution.

2.4 One Dimensional Bin Packing

Before going to multidimensional bin packing, we give a brief description of the results in 1-D bin packing. We also focus primarily on very recent results. For a detailed survey and earlier results we refer the interested reader to [36].

2.4.1 Offline 1-D Bin Packing:

The earliest algorithms for one dimensional (1-D) bin packing were simple greedy algorithms such as First Fit (FF), Next Fit (NF), First Fit Decreasing Heights (FFDH), Next Fit Decreasing Heights (NFDH) etc. In their celebrated work, de la Vega and Lueker [47] gave the first APTAS by introducing linear grouping that reduces the number of different item types. Algorithms based on other item grouping or rounding based techniques have been used in many related problems. The result was substantially improved by Karmarkar and Karp [121] who gave a guarantee of $\text{Opt} + O(\log^2 \text{Opt})$ by providing an iterative rounding for a linear programming formulation.
Table 1: Approximation algorithms for one dimensional bin packing

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Performance Guarantee</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Fit [115]</td>
<td>$2 \cdot \text{Opt}$</td>
<td>Greedy, Online,</td>
</tr>
<tr>
<td>Next Fit Decreasing [115]</td>
<td>$T_\infty \cdot \text{Opt} + O(1)$ [6]</td>
<td>Presorting</td>
</tr>
<tr>
<td>First Fit [115]</td>
<td>$\lceil 1.7 \text{Opt} \rceil$ [51]</td>
<td>Greedy, Online</td>
</tr>
<tr>
<td>First Fit Decreasing [115]</td>
<td>$\frac{11}{6} \text{Opt} + \frac{6}{5}$ [50]</td>
<td>Presorting</td>
</tr>
<tr>
<td>de la Vega and Lueker [47]</td>
<td>$(1 + \epsilon) \text{Opt} + O(\frac{1}{\epsilon})$</td>
<td>Linear grouping</td>
</tr>
<tr>
<td>Karp and Karmarkar [121]</td>
<td>$\text{Opt} + O(\log^2 \text{Opt})$</td>
<td>Iterative rounding</td>
</tr>
<tr>
<td>Rothvoß [161]</td>
<td>$\text{Opt} + O(\log \text{Opt} \cdot \log \log \text{Opt})$</td>
<td>Discrepancy methods</td>
</tr>
<tr>
<td>Hoberg and Rothvoß [97]</td>
<td>$\text{Opt} + O(\log \text{Opt})$</td>
<td>Discrepancy methods</td>
</tr>
</tbody>
</table>

Recently, this has been improved by Rothvoß [161] to $\text{Opt} + O(\log \text{Opt} \cdot \log \log \text{Opt})$ using ideas from discrepancy theory. Very recently, Hoberg and Rothvoß [97] further achieved approximation ratio of $\text{Opt} + O(\log \text{Opt})$ using discrepancy method coupled with a novel 2-stage packing approach. On the other hand, the possibility of an algorithm with an $\text{Opt} + 1$ guarantee is still open. This is one of the top ten open problems in the field of approximation algorithms mentioned in [186].

Table 1 summarizes different algorithms and their performance guarantees. Here $T_\infty$ is the well-known harmonic constant that appears ubiquitously in the context of bin packing.

The Gilmore-Gomory LP relaxation [80] is used in [47, 121, 161] to obtain better approximation. This LP is of the following form:

$$
\min \left\{ 1^T x | Ax = 1, x \geq 0 \right\}
$$

(1)

Here $A$ is the pattern matrix that consists of all column vectors $\{p \in \mathbb{N}^n | p^T s \leq 1\}$ where $s := (s_1, s_2, \ldots, s_n)$ is the size vector for the items. Each such column $p$ is called a pattern and corresponds to a feasible multiset of items that can be assigned to a single bin. Now if we only consider patterns $p \in \{0, 1\}^n$, LP (1) can be interpreted as an LP relaxation of a set cover problem, in which a set $I$ of items has to be covered by configurations from the collection $C \subseteq 2^I$, where each configuration $C \in C$ corresponds to a set of items that can be packed into a bin:
\[
\min \left\{ \sum_{C \in C} x_C : \sum_{C \ni i} x_C \geq 1 \quad (i \in I), x_C \in \{0, 1\} \quad (C \in C) \right\}.
\] (2)

This configuration LP is also used in other algorithms for multidimensional bin packing and we will discuss more on configuration LPs in later sections.

Let \( \text{Opt} \) and \( \text{Opt}_f \) be the value of the optimal integer solution and fractional solution for LP (1) respectively. Although LP (1) has an exponential number of variables, one can compute a basic solution \( x \) with \( 1^T x \leq \text{Opt}_f + \delta \) in time polynomial in \( n \) and \( 1/\delta \) using the Grötschel -Lovász-Schrijver variant of the Ellipsoid method [87], Plotkin-Shmoys-Tardos framework [155, 4]. In fact the analysis of [97], only shows an upper bound of \( O(\log \text{Opt}) \) on the additive integrality gap of LP (1). It has been conjectured in [164] that the LP has the Modified Integer Roundup Property, i.e., \( \text{Opt} \leq \lceil \text{Opt}_f \rceil + 1 \). The conjecture has been proved true for the case when the instance contains at most 7 different item sizes [167]. Recently, Eisenbrand et al. [57] found a connection between coloring permutations and bin packing that shows that Beck’s Three Permutation Conjecture (any three permutations can be bi-colored with \( O(1) \) discrepancy) would imply a \( O(1) \) integrality gap for instances with all items sizes bigger than 1/4. However Newman, Neiman and Nikolov [149] found a counterexample to Beck’s conjecture. Using these insights Eisenbrand et al. [58] showed that a broad class of algorithms can not give an \( o(\log n) \) gap. Rothvoß [160] further explored this connection with discrepancy theory and gave a rounding using Beck’s entropy method achieving \( O(\log^2 \text{Opt}) \) gap alternatively. The later improvement to \( O(\log \text{Opt}) \) in [161, 97] arose from the constructive partial coloring lemma [138] and gluing techniques. Recently Goemans and Rothvoß [82] also have shown polynomiality for bin packing when there are \( O(1) \) number of item types.
2.4.2 Online 1-D Bin Packing:

An online bin packing algorithm uses \( k \)-bounded space if, for each item, the choice of where to pack it, is restricted to a set of at most \( k \) open bins. Lee and Lee [131] gave a \( O(1) \)-bounded space algorithm that achieve asymptotic competitive ratio of \( T_\infty \approx 1.69 \). They also showed it to be tight. Seiden [168] gave a new algorithm, HARMONIC++, whose asymptotic performance ratio is at most 1.58889, the current best among online algorithms that are not bounded space. Ramanan et al. [159] showed that Harmonic-type algorithms can not achieve better than 1.58333 asymptotic competitive ratio. In general the best known lower bound for asymptotic competitive ratio is 1.54014 [182]. Very recently, Balogh et al. [9] presented an online bin packing algorithm with an absolute competitive ratio of 5/3 which is optimal.

2.5 Multidimensional Bin Packing

In this section we discuss the preliminaries related to multidimensional bin packing. We will consider the offline version, when all items are known a priori. We also briefly survey results in the online version, when the items appear one at a time and we need to decide packing an item without knowing the future items.

2.5.1 Geometric Packing

Definition 2.5.1. Two-Dimensional Geometric Bin Packing (2-D GBP) :

In two-dimensional geometric bin packing (2-D GBP), we are given a collection of \( n \) rectangular items \( I := \{r_1, r_2, \ldots, r_n\} \) where each rectangle \( r_k \) is specified by their width and height \( (w_k, h_k) \) such that \( w_k, h_k \) are rational numbers in \([0, 1]\). The goal is to pack all rectangles into a minimum number of unit square bins.

We consider the widely studied orthogonal packing case, where the items must be placed in the bin such that their sides are parallel to the sides of the bin. In any feasible solution, items are not allowed to overlap. Here two variants are usually
studied, (i) where the items cannot be rotated, and (ii) they can be rotated by 90 degrees.

We will also briefly discuss results related to strip packing and geometric knapsack problems, two other geometric generalizations of bin packing, in Section 3.

**Definition 2.5.2. Strip Packing (2-D SP)**: In two-dimensional *strip packing* (2-D SP), we are given a strip of unit width and infinite height, and a collection of $n$ rectangular items $I := \{r_1, r_2, \ldots, r_n\}$ where each rectangle $r_k$ is specified by their width and height $(w_k, h_k)$ such that $w_k, h_k$ are rational numbers in $[0, 1]$. The goal is to pack all rectangles into the strip minimizing the height.

**Definition 2.5.3. Geometric Knapsack (2-D GK)**: In two-dimensional *geometric knapsack* (2-D GK), we are given a unit square bin and a collection of two-dimensional rectangles $I := \{r_1, r_2, \ldots, r_n\}$ where each rectangle $r_k$ is specified by their width and height $(w_k, h_k)$ and profit $p_k$ such that $w_k, h_k, p_k$ are rational numbers in $[0, 1]$. The goal is to find the maximum profit subset that can be feasibly packed into the bin.

Multidimensional variants of above three geometric problems are defined analogously using $d$-dimensional rectangular parallelepipeds (also known as $d$-orthotope, the generalization of rectangles in higher dimensions) and $d$-dimensional cuboids (also known as $d$-cube, the generalization of squares in higher dimensions). We will discuss some more on 3-dimensional variants in Section 3.

### 2.5.2 Vector Packing

Now we define vector bin packing, the nongeometric generalization of bin packing.

**Definition 2.5.4. Vector Bin Packing ($d$-D VBP)**: In $d$-dimensional *vector packing* ($d$-D VBP), we are given a set of $n$ rational vectors $I := \{v_1, v_2, \ldots, v_n\}$ from $[0, 1]^d$. The goal is to partition them into sets (bins) $B_1, B_2, \ldots, B_m$ such that
\[ \|\sigma_{B_i}\|_\infty \leq 1 \text{ for } 1 \leq j \leq m \text{ where } \sigma_{B_i} = \sum_{v_i \in B_1} v_i \text{ is the sum of vectors in } B_1, \text{ and we want to minimize } m, \text{ the number of bins.} \]

In other words, the goal is to pack all the vectors into minimum number of bins so that for every bin the sum of packed vectors in the bin should not exceed the vector of the bin in each dimension.

### 2.5.3 Weighted Bipartite Edge Coloring

Now we discuss the weighted bipartite edge coloring problem.

**Definition 2.5.5. Weighted Bipartite Edge Coloring (WBEC) :** In *weighted bipartite edge coloring* problem, we are given an edge-weighted bipartite graph \( G = (V, E) \) with weights \( w : E \to [0, 1] \). The task is to find a *proper weighted coloring* of the edges with as few colors as possible.

Here proper weighted coloring is defined as follows.

**Definition 2.5.6. Proper Weighted Coloring :** An edge coloring of the weighted graph is called a *proper weighted coloring* if the sum of the weights of the edges incident to a vertex of any color is at most one.

Thus when all edge weights are one, the problem reduces to classical edge coloring problem. Whereas, if there are only two vertices, the problem reduces to classical bin packing problem. On the other hand if the graph has \( n \) vertices, it becomes a special case of \( O(n) \)-dimensional vector packing.

### 2.5.4 Relation between the problems

Figure 1 and 2 show the difference between geometric packing and vector packing. Given a set of vectors, one can easily determine whether they can be packed into one unit bin by just checking whether the sum along each coordinate is at most one or not. However for geometric bin packing, it is \( \text{NP} \)-hard to determine whether a set of
rectangles can be packed in one unit square bin or not, implying that no (absolute) approximation better than 2 is possible even for 2-D GBP.

Note that both geometric knapsack and strip packing are closely related to geometric bin packing. Results and techniques related to strip packing and knapsack have played a major role in improving the approximation for geometric bin packing. If all items have same height then $d$-dimensional strip packing reduces to $(d-1)$-dimensional geometric bin packing. On the other hand to decide whether a set of rectangles $(w_i, h_i)$ for $i \in [n]$ can be packed into $m$ bins, one can define a 3-D geometric knapsack instance with $n$ items $(w_i, h_i, 1/m)$ and profit $(w_i \cdot h_i \cdot 1/m)$ and decide if there is a feasible packing with profit $\sum_{i\in[n]}(w_i \cdot h_i \cdot 1/m)$. Figure 3 shows the relation between different generalizations of bin packing.

2.6 Techniques:

Now we describe some of the techniques heavily encountered in multidimensional bin packing.
Next Fit Decreasing Height (NFDH)

Next Fit Decreasing Height (NFDH) procedure was introduced by Coffman et al. [116]. NFDH considers items in a non-increasing order of height and greedily packs items in this order into shelves, where a shelf is a row of items having their bases on a line that is either the base of the bin or the line drawn at the top of the highest item packed in the shelf below. More specifically, items are packed left-justified starting from bottom-left corner of the bin, until the next item does not fit. Then the shelf is closed and the next item is used to define a new shelf whose base touches the tallest (left most) item of the previous shelf. If the shelf does not fit into the bin, the bin is closed and a new bin is opened. The procedure continues till all the items are packed. This simple heuristic works quite good for small items. Some key properties of NFDH are following:

**Lemma 2.6.1.** [143] Let $B$ be a rectangular region with width $w$ and height $h$. We can pack small rectangles (with both width and height less than $\epsilon$) with total area $A$ using NFDH into $B$ if $w \geq \epsilon$ and $w \cdot h \geq 2A + w^2/8$. The result is best possible.

**Lemma 2.6.2.** [22] Given a set of items of total area of $V$ (each having height at most one), they can be packed in at most $4V + 3$ bins by NFDH.
Lemma 2.6.3. [116] Let $B$ be a rectangular region with width $w$ and height $h$. If we pack small rectangles (with both width and height less than $\epsilon$) using NFDH into $B$, total $w \cdot h - (w + h) \cdot \epsilon$ area can be packed, i.e., the total wasted volume in $B$ is at most $(w + h) \cdot \epsilon$.

In fact it can be generalized to $d$-dimensions.

Lemma 2.6.4. [12] Let $C$ be a set of $d$-dimensional cubes (where $d \geq 2$) of side smaller than $\epsilon$. Consider NFDH heuristic applied to $C$. If NFDH cannot place any other cube in a rectangle $R$ of size $r_1 \times r_2 \times \cdots r_d$ (with $r_i \leq 1$), the total wasted (unfilled) volume in that bin is at most: $\epsilon \sum_{i=1}^{d} r_i$.

2.6.2 Configuration LP

The best known approximations for most bin packing type problems are based on strong LP formulations called configuration LPs. Here there is a variable for each possible way of feasibly packing a bin (called a configuration). This allows the packing problem to be cast as a set covering problem, where each item in the instance $I$ must be covered by some configuration. Let $C$ denote the set of all valid configurations for the instance $I$. The configuration LP is defined as:

$$
\min \left\{ \sum_{C \in C} x_C : \sum_{C \ni i} x_C \geq 1 \quad \forall i \in I, \quad x_C \geq 0 \quad \forall C \in C \right\}. \tag{3}
$$

As the size of $C$ can possibly be exponential in the size of $I$, one typically considers the dual of the LP given by:

$$
\max \left\{ \sum_{i \in I} v_i : \sum_{i \in C} v_i \leq 1 \quad \forall C \in C, \quad v_i \geq 0 \quad \forall i \in I \right\}. \tag{4}
$$

The separation problem for the dual is the following knapsack problem. Given set of weights $v_i$, is there a feasible configuration with total weight of items more than 1. From the well-known connection between separation and optimization [85, 155, 87], solving the dual separation problem to within a $(1 + \epsilon)$ accuracy suffices to solve the
configuration LP within $(1 + \epsilon)$ accuracy. We refer the readers to [160] for an explicit proof that for any set family $S \subseteq 2^{[n]}$ that admits a PTAS for the dual separation problem, the corresponding column-based LP can be solved within an arbitrarily small additive error. This error term can not be avoided as otherwise we can decide the PARTITION problem in polynomial time.

Note that the configurations in (3) are defined based on the original item sizes (without any rounding). However, for more complex problems (say 3-D GBP) one cannot hope to solve such an LP to within $1 + \epsilon$ accuracy, as the dual separation problem becomes at least as hard as 2-D GBP. In general, given a problem instance $I$, one can define a configuration LP in multiple ways (say where the configurations are based on rounded sizes of items in $I$, which might be necessary if the LP with original sizes is intractable). For the special case of 2-D GBP, the separation problem for the dual (4) is the 2-D geometric knapsack problem for which the best known result is only a 2-approximation. However, Bansal et al. [10] showed that the configuration LP (3) with original sizes can still be solved to within $1 + \epsilon$ accuracy (this is a non-trivial result and requires various ideas).

Similarly for the case of vector bin packing, the separation problem for the dual (4) is the vector knapsack problem which could be solved to within $1 + \epsilon$ accuracy [71]. However there is no EPTAS even for 2-D vector knapsack [129].

The fact that solving the configuration LP does not incur any loss for 2-D GBP or VBP plays a key role in the present best approximation algorithms.

2.6.3 Round and Approx (R&A) Framework

Now we describe the R&A Framework as described in [11]. It is the key framework used to obtain present best approximation algorithms for the 2-D geometric bin packing and vector bin packing.

1. Solve the LP relaxation of (3) using the APTAS ([10] for 2-D GBP, [71] for
VBP). Let $x^*$ be the (near)-optimal solution of the LP relaxation and let $z^* = \sum_{C \in C} x^*_C$. Let $r$ the number of configurations in the support of $x^*$.

2. Initialize a $|C|$-dimensional binary vector $x^r$ to be a all-0 vector. For $\lceil \ln \rho \rceil z^*$ iterations repeat the following: select a configuration $C'' \in C$ at random with probability $x^*_{C''}/z^*$ and let $x^r_{C''} := 1$.

3. Let $S$ be the remaining set of items not covered by $x^r$, i.e., $i \in S$ if and only if $\sum_{C \ni i} x^r_C = 0$. On set $S$, apply the $\rho$-approximation algorithm $A$ that rounds the items to $O(1)$ types and then pack. Let $x^a$ be the solution returned by $A$ for the residual instance $S$.

4. Return $x = x^r + x^a$.

Let $\text{Opt}(S)$ and $A(S)$ denote the value of the optimal solution and the approximation algorithm used to solve the residual instance, respectively. Since the algorithm uses randomized rounding in step 2, the residual instance $S$ is not known in advance. However, the algorithm should perform “well” independent of $S$. For this purpose, Bansal, Caprara and Sviridenko [11] defined the notion of subset-obliviousness where quality of approximation algorithm to solve the residual instance is expressed using a small collection of vectors in $\mathbb{R}^{|I|}$.

**Definition 2.6.5.** An asymptotic $\rho$-approximation for the set covering problem defined in (1), is called subset-oblivious if, for any fixed $\epsilon > 0$, there exist constants $k, \Lambda, \beta$ (possibly dependent on $\epsilon$), such that for every instance $I$ of (1), there exist vectors $v^1, v^2 \cdots v^k \in \mathbb{R}^{|I|}$ that satisfy the following properties:

1. $\sum_{i \in C} v^j_i \leq \Lambda$, for each configuration $C \in C$ and $j = 1, 2, \cdots k$;

2. $\text{Opt}(I) \geq \sum_{i \in I} v^j_i$ for $j = 1, 2, \cdots k$;

3. $A(S) \leq \rho \sum_{i \in S} v^j_i + \epsilon \text{Opt}(I) + \beta$, for each $S \subseteq I$ and $j = 1, 2, \cdots k$. 28
Roughly speaking, the vectors are analogues of the sizes of items and are introduced to use the properties of dual of (1). Property 1 says that the vectors divided by constant $\lambda$ must be feasible for (2). Property 2 provides lower bound for $OPT(I)$ and Property 3 guarantees that the $\mathcal{A}(S)$ is not significantly larger than $\rho$ times the lower bound in Property 2 associated with $S$.

The main result about the R&A is the following.

**Theorem 2.6.6.** (simplified) If a problem has a $\rho$-asymptotic approximation algorithm that is subset oblivious, and the configuration LP with original item sizes can be solved to within $(1 + \epsilon)$ accuracy in polynomial time for any $\epsilon > 0$, then the R&A framework gives a $1 + \ln \rho$ asymptotic approximation.

One can derandomize the above procedure and get a deterministic version of R&A method in which Step 2 is replaced by a greedy procedure that defines $x^r$ guided by a suitable potential function. See [11] for the details of derandomization.

**2.6.4 Algorithms based on rounding items to constant number of types:**

Rounding of items to $O(1)$ types has been used implicitly [14] or explicitly [47, 121, 21, 104, 120], in almost all bin packing algorithms to reduce the problem complexity and make it tractable. One of the key properties of rounding items is as follows:

**Lemma 2.6.7.** [121] Let $I$ be a given set of items and $s_x$ be the size of item $x \in I$. Define a partial order on bin packing instances as follows: $I \leq J$ if there exists a bijective function $f : I \rightarrow J$ such that $s_x \leq s_{f(x)}$ for each item $s \in I$. $J$ is then called a rounded up instance of $I$ and $I \leq J$ implies $\text{Opt}(I) \leq \text{Opt}(J)$.

There are typically two types of rounding: either the size of an item in some coordinate (such as width or height) is rounded in an instance-oblivious way (e.g. in harmonic rounding [131, 21], or rounding sizes to geometric powers[121]), or it is rounded in an input sensitive way (e.g. in linear grouping [47]).
**Linear grouping:** Linear grouping was introduced by Fernandez de la Vega and Lueker [47] in the first approximation scheme for 1-D bin packing problem. It is a technique to reduce the number of distinct item sizes. The scheme works as follows, and is based on a parameter $k$, to be fixed later. Sort the $n$ items by nondecreasing size and partition them into $\lceil 1/k \rceil$ groups such that the first group consists of the largest $\lfloor nk \rfloor$ pieces, next group consists of the next $\lfloor nk \rfloor$ largest items and so on, until all items have been placed in a group. Apart from the last group all other groups contain $\lfloor nk \rfloor$ items and the last group contains $\leq nk$ items. The rounded instance $\bar{I}$ is created by discarding the first group, and for each other group, the size of an item is rounded up to the size of the largest item in that group. Following lemma shows that the optimal packing of these rounded items is very close to the optimal packing of the original items.

**Lemma 2.6.8.** [47] Let $\bar{I}$ be the set of items obtained from an input $I$ by applying linear grouping with group size $\lfloor nk \rfloor$, then

$$\text{Opt}(\bar{I}) \leq \text{Opt}(I) \leq \text{Opt}(\bar{I}) + \lfloor nk \rfloor$$

and furthermore, any packing of $\bar{I}$ can be used to generate a packing of $I$ with at most $\lfloor nk \rfloor$ additional bins.

If all items are $> \epsilon$, then $n\epsilon < \text{Opt}$. So, for $k = \epsilon^2$ we get that any packing of $\bar{I}$ can be used to generate a packing of $I$ with at most $\lfloor n\epsilon^2 \rfloor \leq n\epsilon^2 + 1 < \epsilon \cdot \text{Opt} + 1$ additional bins. In Chapter 4, we will use a slightly modified version of linear grouping that does not loose the additive constant of 1.

**Geometric grouping:** Karmarkar and Karp [121] introduced a refinement of linear grouping called geometric grouping with parameter $k$. Let $a(I)$ be the smallest item size of an instance $I$. For $r = 0, 1, \ldots \lfloor \log_2 \frac{1}{a(I)} \rfloor$, let $I_r$ be the instance consisting of
items \( i \in I \) such that \( s_i \in (2^{-(r+1)}, 2^{-r}] \). Let \( J_r \) be the instances obtained by applying linear grouping with parameter \( k \cdot 2^r \) to \( I_r \). If \( J = \bigcup_r J_r \) then:

**Lemma 2.6.9.** [121] \( \text{Opt}(J) \leq \text{Opt}(I) \leq \text{Opt}(J) + k \lceil \log_2 \frac{1}{\alpha(I)} \rceil \)

**Harmonic rounding:** Lee and Lee [131] introduced harmonic algorithm (\( \text{harmonic}_k \)) for online 1-D bin packing, where each item \( j \) with \( s_j \in (\frac{1}{q+1}, \frac{1}{q}] \) for \( q \in \{1, 2, \cdots, (k-1)\} \), is rounded to \( 1/q \). Then \( q \) items of type \( 1/q \) can be packed together in a bin. So for each type \( q \), only items of the same type are packed into the same bin and new bins are opened when the current bin is full. Let \( t_1 = 1, t_{q+1} := t_q(t_q + 1) \) for \( q \geq 1 \). Let \( m(k) \) be the integer with \( t_{m(k)} < k < t_{m(k)+1} \). It is shown in [131] that the asymptotic approximation ratio of \( \text{harmonic}_k \) is \( T_k = \sum_{q=1}^{m(k)} \frac{1}{t_q} + \frac{1}{t_{m(k)+1}} \cdot k \). When \( k \to \infty \), \( T_\infty = 1.691.. \), this is the harmonic constant, ubiquitous in bin packing. Caprara [21] introduced \( \text{harmonic decreasing height} \) algorithm for 2-D GBP with asymptotic approximation ratio of \( T_\infty \), where widths are rounded according to harmonic rounding and then same width items are packed using NFDH. We refer the readers to [62] for more related applications of the Harmonic algorithm in online and bounded space algorithms.
Chapter III

GEOMETRIC BIN PACKING

In this chapter we discuss our results related to geometric bin packing and its variants. Our main result is an improved algorithm for the 2-D bin packing problem stated as follow:

Theorem 3.0.10. There is a polynomial time algorithm with an asymptotic approximation ratio of $\ln(1.5) + 1 \approx 1.405$ for 2-D GBP. This holds both for the versions with and without rotations.

The main idea behind Theorem 3.0.10 is to show that the Round and Approx (R&A) framework introduced by [11] (See section 2.6.3) can be applied to the result of Jansen and Prädel [104]. Roughly speaking, this framework states that given a packing problem, if (i) the configuration LP for the problem (with the original item sizes) can be solved up to error $1 + \epsilon$ for any $\epsilon > 0$, and (ii) there is a $\rho$-approximation for the problem that is subset-oblivious; then one can obtain a $(1 + \ln \rho)$-asymptotic approximation for the problem.

In [11], it was shown that the APTAS for 1-D BP due to [47] and the 2-D BP algorithm of [21] are subset-oblivious. However, the notion of subset-obliviousness as defined in [11] is based on various properties of dual-weighting functions, making it somewhat tedious to apply and also limited in scope (e.g. it is unclear to us how to apply this method directly to the algorithm of [104]).

In this chapter we give a more general argument to apply the R&A framework directly to a wide class of algorithms$^1$, and without any reference to dual-weighting

$^1$This includes all known algorithms that we know of for bin-packing type problems, except the ones based on R&A method.
functions. In particular, we show that any algorithm based on rounding the (large) items into $O(1)$ types, is subset-oblivious.

The main observation is that any $\rho$-approximation based on rounding the item sizes can be related to another configuration LP (on rounded item sizes) whose solution is no worse than $\rho$ times the optimum solution. As the item sizes are rounded, there are only $O(1)$ constraints in this LP and it can be easily shown to be subset oblivious.

For the particular case of 2-D BP, we present the algorithm of Jansen and Prädel that directly fits in the above framework. As most algorithms for bin-packing problems are based on rounding into $O(1)$ types, this also makes the framework widely applicable. For example, this gives much simpler proofs of all the results in [11].

Finally, we give some results to show the limitations of rounding based algorithms in obtaining better approximation ratios. Rounding of items to $O(1)$ types has been used implicitly [14] or explicitly [47, 121, 21, 104, 120], in almost all bin packing algorithms. There are typically two types of rounding: either the size of an item in some coordinate (such as width or height) is rounded up in an instance-oblivious way (e.g. in Harmonic rounding [131, 21], or rounding sizes to geometric powers [121]), or it is rounded up in an input sensitive way (e.g. in linear grouping [47]). We also show limitations of rounding based algorithms as follows.

**Theorem 3.0.11.** Any rounding algorithm that rounds at least one side of each large item to some number in a constant-size collection of numbers, independent of the problem instance (let us call such rounding input-agnostic), cannot have an approximation ratio better than $3/2$. 

33
Organization of the chapter: In Section 3.1 we survey previous works in geometric bin packing and its variants. In Section 3.2, we describe how the Round and Approx framework can be applied to rounding based algorithms. In Section 3.3, we present the 1.5 approximation algorithm of [104] and show how the Round and Approx framework applies to it. Then in Section 3.4 we show our lower bounds for rounding based algorithms. Finally, in Section 3.5 we make some final remarks.

3.1 Prior Works

In this section we give an extensive survey of the literature related to geometric packing and other related problems.

3.1.1 Geometric Bin Packing

The two-dimensional geometric bin packing (GBP) is substantially different from the 1-D case. Bansal et al. [12] showed that 2-D bin packing in general does not admit an APTAS unless \( P = NP \).

On the positive side, there has also been a long sequence of works giving improved algorithms. We refer readers to [136] for a review of several greedy heuristics such as Next Fit Decreasing, First Fit Decreasing, Best Fit Decreasing, Finite Best Strip, Floor-Ceiling Algorithm, Finite First Fit, Knapsack Packing algorithm, Finite Bottom-Left, Alternate Directions etc. For the case when we do not allow rotation, until the mid 90’s the best known bound was a 2.125 approximation [31], which was improved by Kenyon and Rémila [125] to a \( 2 + \epsilon \) approximation (this comes as a corollary of their APTAS for 2-D strip packing) for any \( \epsilon > 0 \). Caprara in his breakthrough paper [21] gave an algorithm for 2-D bin packing attaining an asymptotic approximation ratio of \( T_\infty(\approx 1.69103) \).

For the case when rotations are allowed, Miyazawa and Wakabayashi [145] gave an algorithm with asymptotic performance guarantee 2.64. Epstein and Stee [64] improved it to 2.25 by giving a packing where in almost all bins an area of 4/9
is occupied. Finally an asymptotic approximation guarantee arbitrarily close to 2 followed from the result of [111].

This was later improved by Bansal et al. [11] to \((\ln(T_\infty) + 1) \approx 1.52\) for both the cases with and without rotation introducing a novel randomized rounding based framework: Round and Approx (R & A) framework. Jansen and Prädel [104] improved this guarantee further to give a 1.5-approximation algorithm. Their algorithm is based on exploiting several non-trivial structural properties of how items can be packed in a bin. This is the best algorithm known so far, and holds both for the case with and without rotations.

We remark that there is still a huge gap between these upper bounds and known lower bounds. In particular, the best known explicit lower bound on the asymptotic approximation for 2-D BP is currently \(1 + 1/3792\) and \(1 + 1/2196\) for the versions with and without rotations respectively [30]. The best asymptotic worst-case ratio that is achievable in polynomial time for \(d\)-dimensional GBP for \(d > 2\) is \(T_\infty^{d-1}\) [21], and in fact it can be achieved by an online algorithm using bounded space. There are no known explicit better lower bounds for higher dimensions.

In non-asymptotic setting, without rotations there is a 3-approximation by Zhang [189] and by Harren and van Stee [96]. Harren and van Stee [94] gave a non-asymptotic 2-approximation with rotations. Independently this approximation guarantee is also achieved for the version without rotations by Harren and van Stee [95] and Jansen et al. [106]. These 2-approximation results match the non-asymptotic lower bound of this problem, unless \(P = NP\).

### 3.1.2 Square Packing

Leung et al. [132] have shown that even the special case of packing squares into square is still \(NP\)-hard. Kohayakawa [126] et al. gave a \((2 - (2/3)^d + \epsilon)\) approximation for packing \(d\)-dimensional cubes into unit cubes. Later Bansal et al. [12] have given
an APTAS for the problem of packing $d$-dimensional cubes into $d$-dimensional unit cubes.

### 3.1.3 Online Packing:

Coppersmith and Raghavan [40] first studied online 2-D GBP and gave algorithms with asymptotic performance ratio of 3.25 and 6.35 for $d = 2$ and 3 respectively. Csirik and van Vliet [43] showed algorithm with performance ratio $T^d_\infty$ (This gives 2.859 for 2-D) for arbitrary dimension $d$. Epstein and van Stee [64] achieved the same ratio of $T^d_\infty$ only using bounded space and showed it to be the optimal among all bounded space algorithms. In 2002, Seiden and van Stee [169] proposed an elegant algorithm called $H \otimes C$, comprised of the Harmonic algorithm $H$ and the Improved Harmonic algorithm $C$, for the two-dimensional online bin packing problem and proved that the algorithm has an asymptotic competitive ratio of at most 2.66013. Since the best known online algorithm for one-dimensional bin packing is the Super Harmonic algorithm [168], a natural question was could a better upper bound be achieved by using the Super Harmonic algorithm instead of the Improved Harmonic algorithm? Han et al. [90] gave a positive answer for this question and a new upper bound of 2.5545 is obtained for the two-dimensional online bin packing. The main idea is to develop new weighting functions for the Super Harmonic algorithm and propose new techniques to bound the total weight in a rectangular bin. The best known lower bound is 1.907 by Blitz, van Vliet and Woeginger [18]. We refer the readers to [181] for a survey of online algorithms for geometric bin packing in multiple dimensions.

When we allow rotation, Epstein [61] gave an algorithm with asymptotic performance ratio of 2.45. Later Epstein and van Stee [65] gave an algorithm with asymptotic performance ratio of 2.25.

For the special case where items are squares, there is also a large number of results. Coppersmith and Raghavan [40] showed their algorithm gives asymptotic
performance ratio of 2.6875 in this case. They also gave a lower bound of 4/3. Seiden and van Stee [169] gave an algorithm with asymptotic performance ratio of 2.24437. Epstein and van Stee [63] have shown an upper bound of 2.2697 and a lower bound of 1.6406 for online square packing, and an upper bound of 2.9421 and a lower bound of 1.6680 for online cube packing. The upper bound for squares can be further reduced to 2.24437 using a computer-aided proof. Later Han et al. [91] get an upper bound of 2.1187 for square packing and 2.6161 for cube packing. Recently Kamali et al. [118] gave an upper bound of 1.84. For bounded space online algorithms, Epstein and van Stee [66] showed lower and upper bounds for optimal online bounded space hypercube packing till dimensions 7. In particular, for 2-D it lies in (2.3634, 2.3692) and for 3-D it lies in (2.956, 3.0672).

3.1.4 Heuristics

Lodi et al. have reviewed several exact algorithms based on enumerative approach or branch-and-bound in [136]. They have also studied several integer programming models for 2-D GBP and other related problems. Pisinger and Sigurd [154] gave an algorithm based on Dantzig-Wolfe decomposition. Here the master problem is a set covering problem, solved by delayed column generation. The subproblem deals with the packing of rectangles into a single bin and is solved as a constraint-satisfaction problem (CSP). Martello and Vigo [141] had considered exact solutions for 2-D GBP for instance sizes upto 120.

Jylanki [117] reviewed several greedy heuristics in detail and did empirical study. Primarily he considered five broad classes of algorithms: shelf algorithms, fit-based guillotine algorithms, split-based guillotine algorithms, maximal rectangle algorithms and skyline algorithms. Lodi [135] also has surveyed several one-phase, two-phase and non-shelf algorithms.
In the past two decades many local search and meta-heuristic algorithms for rectangle packing problems have been proposed. We refer interested readers to Aarts and Lenstra [1] and Glover and Laguna [81] for a detailed survey. Dowsland [53] proposed one of the first meta-heuristic approaches to strip packing using simulated annealing. During the search the objective was to minimize pair-wise overlapping area of rectangles and the neighborhood contained all solutions representing vertical or horizontal items shifting. Jakobs [103] presented a genetic algorithm based approach. It is based on representation of packing pattern by an order given by permutation, and packing positions are determined by a Bottom-Left strategy. There are many similar algorithms based on a permutation coding scheme. These algorithms consists of two phases: (1) find a good permutation using some meta-heuristic, (2) the decoding algorithm puts items following the permutation order. Several interesting coding schemes are proposed such as $n$-leaf binary tree [110], sequence pair [146], bounded sliceline grid (BCG) [147], O-tree [88], $B^*$-tree [27], quarter-state sequence [162] etc.

Lodi et al. [137] introduced a Tabu Search framework exploiting a new constructive heuristic for the evaluation of the neighborhood for 3-D GBP. Kell and Hoeve [123] investigated the application of multivalued decision diagrams (MDDs) to multidimensional bin packing problems. Pasha [153] has studied geometric bin packing algorithms for arbitrary shapes and presented several genetic algorithms and simulated annealing based approach. Kroger [128] introduced a sequential and a parallel genetic algorithm based on guillotine cuts. Faroe et al. [67] have given a guided local search based algorithm for 3-D GBP. For other heuristics for cutting and packing related problems please see the application-oriented research bibliography [179] by Sweeney and Paternoster. For further details on meta-heuristics for rectangle packing we refer the readers to [101, 100, 19].

For practical problems such as pallet packing in warehouses, several other factor are needed to be considered such as stability of packing (under gravity), elasticity etc.
Several heuristics are considered in [166, 48] for the stable pallet packing problem. In VLSI design, simulated annealing algorithms are used in practice to solve 3-D Bin Packing problem. Though these algorithms do not have a good worst-case guarantee, they still sometimes work well in practice. There are multiple ways e.g., sequence triple, transitive closure, 3D-CBL [139] to map each solution of bin packing to a list of 0-1 integers to apply simulated annealing and different ways to move in the solution space.

3.1.5 Packing with Augmentation

Due to pathological worst case examples, bin packing has been well studied under resource augmentation, i.e., the side length of the bin is augmented to $(1 + \epsilon)$ instead of one. Though 2-D GBP does not admit an APTAS, Bansal et al. [12] gave a polynomial time algorithm to pack rectangles into at most $m$ number of bins of size $(1 + \epsilon) \times (1 + \epsilon)$ where $m$ is the optimal number of unit bins needed to pack all items. Later Bansal and Sviridenko [15] showed that this is possible even when we relax the size of the bin in only one dimension.

3.1.6 Strip Packing

A closely related problem is the Strip Packing problem. It is another generalization of one dimensional bin packing problem and closely tied with the geometric bin packing problem. As we had stated earlier, the best approximation algorithm for 2-D GBP used to be factor $2 + \epsilon$ and was a corollary from the APTAS for 2-D strip packing due to Kenyon and Rémiña [125]. In two dimensions, we are given a strip with width one and unlimited height and the goal is to pack 2-D rectangular items into the strip so as to minimize the height of the packing. This two dimensional variant is also known as the Cutting Stock Problem. In three dimensions, we are given 3-D rectangular items each of whose dimensions is at most one and they need to be packed into a single 3-D box of unit depth, unit width and unlimited height so as to minimize the height of
First let us discuss offline algorithms for 2-D strip packing. Baker et al. [7] introduced the problem in 1980 and gave an algorithm with absolute approximation ratio of 3. Later Coffman et al. introduced Next-Fit Decreasing Height (NFDH) [116], First-Fit Decreasing Height (FFDH) [116] for 2-D strip packing without rotations, achieving asymptotic approximation ratio as 2 and 1.7 respectively. The upper bound of 2 in NFDH remains valid even for the case with rotations, since the proofs use only area arguments. Epstein and Stee [64] gave a 3/2 approximation algorithm for the problem with rotation. Finally APTAS was given for 2-D strip packing without rotations [125] and with rotations in [111] using nice interplay of techniques like fractional strip packing, linear grouping and a variant of NFDH. For the absolute approximation, Harren et al. [92] have given a $5/3 + \epsilon$ approximation whereas the lower bound is $3/2$ (follows from 2-D bin packing).

Now we discuss the online algorithms for 2-D strip packing. Baker and Schwartz [8] showed First-Fit Shelf has asymptotic performance ratio 1.7. Csirik and Woeginger [44] improved it to $T_\infty \approx 1.691$ using harmonic algorithm as a subroutine. They also mention a lower bound of 1.5401. For the absolute performance ratio, Brown et al. [20] have given a lower bound of 2.

3-D strip packing is a common generalization of both the 2-D bin packing problem (when each item has height exactly one) and the 2-D strip packing problem (when each item has width exactly one). Li and Cheng [134] were among the first people who considered the problem. They showed 3-D versions of FFDH and NFDH have unbounded worst-case ratio. They gave a 3.25 approximation algorithm, and later gave an online algorithm with upper bound of $T_\infty^2 \approx 2.89$ [133] using Harmonic algorithm as a subroutine. Bansal et al. [13] gave a 1.69 approximation for offline case. Recently Jansen and Prädel [105] further improved it to 1.5.
3.1.7 Shelf and Guillotine Packing

For $d = 2$, many simplified structures of the item packings have been considered in the literature, because they are both easy to deal with and important in practical applications. Among these, very famous are the two-stage packing structures, leading to two-dimensional shelf bin packing (2SBP) and two-dimensional shelf strip packing (2SSP). Two-stage packing problems were originally introduced by Gilmore and Gomory [20] and, thinking in terms of cutting instead of packing, require that each item be obtained from the associated bin by at most two stages of cutting.

![Figure 4: Example of two stage packing](image)

In two-stage packing, in the first stage, the bins are horizontally cut into shelves. The second stage produces slices, which contain a single item by cutting the shelves vertically. Finally, an additional stage (called trimming) is allowed in order to separate an item from a waste area. Two-stage packing is equivalent to packing the items into the bins in shelves, where a shelf is a row of items having their bases on a line that is either the base of the bin or the line drawn at the top of the highest item packed in the shelf below. Formally, a shelf is a set $S$ of items such that $\sum_{j \in S} w_j \leq 1$; its
height $h(S)$ is given by $\max_{j \in S} h_j$. Many classical heuristics for 2-D strip packing ([116], [8], [44]) and 2-D GBP ([31]) construct solutions that are in fact feasible for the two-stage versions. Moreover, Caprara et al. [22] presented an APTAS, each for 2SBP and 2SSP. Given this situation, it is natural to ask for the asymptotic worst-case ratio of 2SBP versus 2BP. Csirik and Woeginger [44] showed ratio of 2SSP versus 2-D strip packing is equal to $T_\infty$. Caprara [21] showed the ratio of 2SBP versus
2BP is also equal to $T_\infty$. Both their algorithms are online. Seiden and Woeginger [170] observed that the APTAS of Kenyon and Remila [125] can easily be adapted to produce a near-optimal packing in three stages for 2-D strip packing, showing that the asymptotic worst-case ratio of 2-D strip packing versus its $k$-stage version is 1 for any $k > 2$, and leading to an APTAS for the latter. As to 2-D GBP, Bansal et al. [14] provided an APTAS for the guillotine case, i.e., the case in which the items have to be packed in stages but there is no limit on the number of stages that can be used. In the guillotine case, there is a sequence of edge-to-edge cuts parallel to one of the edges of the bin. Note that, though it is expected, it is not known if an APTAS exists for $d$-SSP and $d$-SBP for $d \geq 3$. Caprara[21] also has shown that Harmonic Decreasing Height (HDH) heuristic is the best shelf algorithm for 2-D Strip packing.

Consider the optimal 2SBP solution in which the shelves are horizontal as well as the optimal 2SBP solution in which they are vertical. (Recall that near-optimal 2SBP solutions can be found in polynomial time [22].) There is no evidence that the asymptotic worst-case ratio between the best of these two solutions and the optimal 2BP one can be as bad as $T_\infty$, and in fact Caprara conjectured that this ratio is 3/2. On the other hand, he also mentions that there are examples where we cannot do better than $T_\infty$, if both solutions are formed by the Harmonic Decreasing Height algorithm in [21].

### 3.1.8 Geometric Knapsack

For 2-D Geometric Knapsack (GK), a result of Steinberg [176] for strip packing translates into a $3 + \epsilon$ approximation [23]. Present best known approximation algorithm is due to Jansen and Zhang [112] and has approximation guarantee of $2 + \epsilon$. On the other hand, no explicit inapproximability results are known. PTAS are known for special cases when resource augmentation in allowed in one dimension [107], all items are square [108] or all items are small [69]. Bansal et al. [10] gave a PTAS for the
special case when the range of the profit-to-area ratio of the rectangles is bounded by a constant. Recently Adamaszek and Wiese [2] gave a quasi-PTAS for the problem. This implies that the problem is not APX-hard (thus we can still hope for a PTAS) unless \( \text{NP} \subseteq \text{QP} \).

### 3.2 R&A Framework for Rounding Based Algorithms

We describe here a general approach to show that a wide class of algorithms for bin-packing type problems, in particular those based on rounding the item sizes to \( O(1) \) types is subset-oblivious. While such algorithms are hard to define formally, we state their general approach below, which subsumes all the known algorithms that we are aware of.

**General form of a rounding based algorithm.** Now let us intuitively present the general form of a rounding based algorithm, which appears frequently in the literature [156, 22]. A typical rounding based algorithm for a \( d \)-dimensional problem has the following form. Given some accuracy parameter \( \epsilon > 0 \), one first defines two functions \( f(\epsilon) \) and \( g(\epsilon) \) (that only depend on \( \epsilon \)) with \( g(\epsilon) \ll f(\epsilon) \).

Now items are divided into three types based on their sizes:

- **Big**: if all its coordinates are at least \( f(\epsilon) \),
- **Small**: if all its coordinates are at most \( g(\epsilon) \),
- **Medium** if at least one coordinate lies in the range \((g(\epsilon), f(\epsilon))\).

A standard argument [22, 156] shows that the functions \( g \) and \( f \) can be chosen such that their ratio is as large as desired, while ensuring that the volume of medium items is at most \( \frac{\epsilon}{10} \) times \( \text{Vol}(I) \), the total volume of items in the input instance \( I \). These items can be ignored as they can be packed in \( \frac{\epsilon}{2} \cdot \text{Opt} \) separate bins using NFDH. We refer readers to Lemma 3.21 in [156] for a detailed proof.
Now, all items have each coordinate either small (in $[0, g(\epsilon)]$) or big (in $[f(\epsilon), 1]$). Call an item \textit{skewed} if it is neither big or small (i.e., some coordinates are less than $g(\epsilon)$ and some more than $f(\epsilon)$). Skewed items can be classified into at most $2^d - 2$ types based on which subset of coordinates is large and $d$ is the total number of coordinates.

Now, the algorithm rounds the large dimensions of big and skewed items to $O(\epsilon(1))$ values (possibly in a very complex way, including guessing of sizes). As $\epsilon$ is a constant, now on we will consider $O(\epsilon(1))$ to be a constant. Now consider another functions $h(\epsilon)$ that only depend on $\epsilon$ with $g(\epsilon) \ll h(\epsilon) \cdot f(\epsilon) \leq f(\epsilon)$. Finally, one argues that in any packing almost all skewed and small items are placed (possibly fractionally) in large regions called \textit{containers}, where each container has large size in each dimension. Specifically one dimension has size $\geq f(\epsilon)$ and the other dimension has size $\geq f(\epsilon) \cdot h(\epsilon)$ and these containers have $O(1)$ types of sizes. Thus these containers can be viewed as big items and the algorithm focuses on packing of constant (say $q$) types of large items and containers. Now there are only $O(1)$ number of possible configurations as there are $O(1)$ types of items (large items or containers). So there are only polynomial number of possible packings of these large items. One can simply use brute-force method to enumerate all possible packing and select the best. The running time can be improved using integer programming techniques for fast mixed linear integer program with few integer variable from [59]. Later almost all skewed items are packed integrally into containers using an assignment LP or other techniques. Afterwards the small items and the remaining skewed items are filled using NFDH in the empty spaces in the packing of big and skewed items. The large separation between $g$ and $f$ ensures that this incurs negligible loss in volume. Then one defines some algorithm $\mathcal{A}$ that finds an almost optimal (say within a factor $(1 + \epsilon_a)$) packing of these rounded big items and containers.

It is easily checked that the algorithm of [104], as stated in section 3.3, falls directly
in this framework. In their algorithm they choose \( f(\epsilon) = \delta, g(\epsilon) = \delta^4, h(\epsilon) = \delta/2 \), all big items are rounded to \( O(\frac{1}{\delta^2} \cdot \frac{1}{\delta^4}) \) types and there are \( O(\frac{1}{\delta^6}) \) types of containers where \( \delta = \epsilon/48 \). We will explain the properties of the algorithm in [104] in more details in the later section. We remark that the rounding of sizes in their algorithm is non-trivial and actually depends on which the bin patterns are used in the optimum solution (that the algorithm will guess).

Relating the algorithm to the configuration LP with rounded item sizes.

For concreteness, let us consider the case of 2-D GBP. Suppose \( \mathcal{A} \) is a rounding based \( \rho \)-approximation algorithm for it. Then, \( \mathcal{A}(I) \leq \rho \cdot \text{Opt}(I) \) for any instance \( I \) of the problem. The items are classified into big and small. There are two types of skewed items: long (with height \( \geq f(\epsilon) \) and width \( \leq g(\epsilon) \)) and wide (with width \( \geq f(\epsilon) \), height \( \leq g(\epsilon) \)). Upon rounding, the big items are rounded to \( O(1) \) types and long (and wide) items are assigned to \( O(1) \) groups of different heights (or widths). Let \( \tilde{I} \) denote instance obtained from \( I \) by rounding the large dimensions according to the rounding performed by \( \mathcal{A} \). Clearly, \( \mathcal{A}(I) \geq \text{Opt}(\tilde{I}) \) and hence

\[
\text{Opt}(\tilde{I}) \leq \rho \cdot \text{Opt}(I).
\] (5)

Now, consider the configuration LP defined on the instance \( \tilde{I} \), where the configurations correspond to feasible packing of items in \( \tilde{I} \). As there are only a constant number (say \( t \)) of such item types, this LP has only a constant number \( t \) of non-trivial constraints, one for each item type.

For \( i \in \{1, \ldots, p_1\} \), let \( B_i \) denote the group of big items rounded to type \( i \) and \( c_{rB_i}^r \) be the number of items of type \( j \) (i.e., from the set \( B_j \)) in the \( r' \)th configuration. Long items are rounded to \( p_2 \) types of heights and assigned to groups \( L_1, \ldots, L_{p_2} \). Similarly, wide items are rounded to \( p_3 \) types of widths and assigned to groups \( W_1, \ldots, W_{p_3} \). Let \( w(L_k) \) denote the total width of items in \( L_k \), and \( h(W_k) \) denote the total height of items in \( W_k \). Similarly, let \( c_{rL_k}^r \) (resp. \( c_{rW_k}^r \)) denote the total width of items in \( L_k \).
(resp. total height of items in $W_\ell$) in the configuration $r$. Let $T$ be the set of small items and $c_T^r$ be the total volume (area in 2-D) of small items in configuration $r$. Let $vol_T(\tilde{I})$ be the total volume of small items (i.e., total area of small items in 2-D) in $\tilde{I}$.

Consider the following configuration LP: LP($\tilde{I}$)

\[
\begin{align*}
\text{Min} & \quad \sum_r x_r \\
\text{s.t.} & \quad \sum_r c_{B_j}^r x_r \geq |B_j| \quad \forall j \in [p_1] \\
& \quad \sum_r c_{L_k}^r x_r \geq w(L_k) \quad \forall k \in [p_2] \\
& \quad \sum_r c_{W_j}^r x_r \geq h(W_\ell) \quad \forall \ell \in [p_3] \\
& \quad \sum_r c_T^r x_r \geq vol_T(\tilde{I}) \\
\end{align*}
\]

\[x_r \geq 0 \quad (r = 0, 1, \ldots, m)\]

Let $t = p_1 + p_2 + p_3 + 1$. As the LP has only $t$ constraints,

\[\text{IP}^*(\tilde{I}) \leq \text{LP}^*(\tilde{I}) + t. \quad (6)\]

where IP$^*$($\tilde{I}$) and LP$^*$($\tilde{I}$) are the optimal integral and linear program solution for LP($\tilde{I}$) respectively.

Intuitively, the integral solution of LP gives a fractional packing of items. Big items are packed integrally. For long (resp. wide) items it preserves the total width (resp. height) in each class of same height (resp. width). However individual long (resp. wide) items may be needed to be sliced vertically (resp. horizontally) and packed fractionally in their small dimension. For small items it preserves the total volume, but the items might be needed to pack fractionally (may be sliced vertically and/or horizontally).

Now any packing of rectangles can be transformed into a structured packing using the following lemma, implicit in Section 3.3.2 in [156].
**Lemma 3.2.1.** [156] If there is a packing of $\tilde{I}$ into $m$ bins where skewed and small items can be packed fractionally into containers, then one can get a packing in $(1 + \epsilon_b) \cdot m + O(1)$ bins where all items are packed integrally.

Hence, we get,

$$A^*(\tilde{I}) \leq (1 + \epsilon_a) \cdot \text{Opt}(\tilde{I}) + O(1) \quad (7)$$

$$\leq (1 + \epsilon_a)(1 + \epsilon_b) \cdot \text{IP}^*(\tilde{I}) + O(1) \quad (8)$$

$$\leq (1 + \epsilon_a)(1 + \epsilon_b) \cdot (\text{LP}^*(\tilde{I}) + t) + O(1) \quad (9)$$

$$\leq (1 + \epsilon_a)(1 + \epsilon_b) \cdot \text{LP}^*(\tilde{I}) + O(1) \quad (10)$$

Here inequality (7) follows from the fact that $A$ finds optimal solution within factor $(1 + \epsilon_a)$. Inequality (8) follows from Lemma 3.2.1. and inequality (9) follows from (6).

**R&A framework for rounding based algorithms.**

We can now show the following.

**Theorem 3.2.2.** If there is a $\rho$ approximation algorithm $A$ that rounds the large coordinate of items to $O(1)$ types before packing (these sizes could depend on the instance $I$), then the R&A method gives a $(1 + \ln \rho + \epsilon)$ asymptotic approximation bin packing for $I$.

**Proof.** First, we consider the configuration LP on original items in step 1 of the R&A framework (see Section 2.6.3) and apply the randomized rounding step to it. The probability that an item $i \in I$ is not covered by any configurations selected in some iteration, is at most $(1 - \sum_{C \ni i} x_C^*/z^*)$. Let $S$ be the set of residual items not covered by any of the bins selected in $[(\ln \rho)z^*]$ iterations. Thus the probability that $i$ is not
covered in any of the \(\lceil (\ln \rho) z^* \rceil\) iterations is at most:

\[
\mathbb{P}(i \in S) \leq (1 - \sum_{C \ni i} x^*_C / z^*)^{\lceil (\ln \rho) z^* \rceil}
\leq (1 - \sum_{C \ni i} x^*_C / z^*)^{(\ln \rho) z^*}
\leq e^{(-\ln \rho)}
= \frac{1}{\rho}.
\]

(11)

(12)

where inequality (11) follows as \(\sum_{C \ni i} x^*_C \geq 1\) for all \(i \in I\) and \((1 - x^{-1})^{\alpha x} \leq e^{-\alpha}\) for \(x > 0\).

Let \(\text{Opt}(I)\) be the number of bins used in the optimal packing of \(I\). Now in step 2 of R & A framework at most \(\lceil (\ln \rho) z^* \rceil \leq 1 + (\ln \rho) \cdot \text{Opt}\) bins were used. Let \(S\) denote the set of items that are still unpacked. It remains to bound the number of bins used for packing \(S\) using \(A\).

To this end, consider the rounding that \(A\) would apply to the items when given instance \(I\), and consider the instance obtained by applying this rounding to items in \(S\). Let us denote this instance as \(\tilde{I} \cap S\). Now consider the following configuration LP for \(\tilde{I} \cap S\): LP(\(\tilde{I} \cap S\))

\[
\begin{align*}
\text{Min} & \quad \sum_r x_r \\
\text{s.t.} & \quad \sum_r c_{lr}^T x_r \geq v(\tilde{T} \cap S) \quad (r = 0, 1, \ldots, m) \\
& \quad \sum_r c_{Bj}^r x_r \geq |B_j \cap S| \quad \forall j \in [p_1] \\
& \quad \sum_r c_{Lk}^r x_r \geq w(Lk \cap S) \quad \forall k \in [p_2] \\
& \quad \sum_r c_{W\ell}^r x_r \geq h(W\ell \cap S) \quad \forall \ell \in [p_3] \\
& \quad x_r \geq 0 \quad (r = 0, 1, \ldots, m)
\end{align*}
\]

Now by (12), we get the following expected values:

\[
\mathbb{E}[|B_j \cap S|] \leq |B_j| / \rho.
\]
\[ \mathbb{E}[w(L_k \cap S)] \leq w(L_k)/\rho, \]
\[ \mathbb{E}[h(W_\ell \cap S)] \leq h(W_\ell)/\rho, \]
\[ \mathbb{E}[\text{vol}_T(\tilde{I} \cap S)] = \frac{\text{vol}_T(\tilde{I})}{\rho}. \]

Now we will show that these values are in fact not very far from the expectations using concentration inequality.

We will need the following concentration inequality [142] in the analysis.

**Lemma 3.2.3.** [Independent Bounded Difference Inequality] [142] Let \( X = (X_1, \ldots, X_n) \) be a family of independent random variables, with \( X_j \in A_j \) for \( j = 1, \ldots, n \), and \( \phi : \prod_{j=1}^n A_j \to \mathbb{R} \) be a function such that

\[ \phi(x) - \phi(x') \leq c_j \]

whenever the vectors \( x \) and \( x' \) differ only in the \( j \)-th coordinate. Let \( \mathbb{E}[\phi(X)] \) be the expected value of the random variable \( \phi(X) \). Then for any \( t \geq 0 \),

\[ \mathbb{P}[\phi(X) - \mathbb{E}(\phi(X)) \geq t] \leq e^{-2t^2/\sum_{j=1}^n c_j^2} \]

Now consider the function \( \phi_i^B(x) = B_i \cap S \), i.e., the number of items of type \( B_i \) in \( S \). This is a function of \( X = (X_1, \ldots, X_{[(\ln \rho)z^*]}) \), i.e., \( [(\ln \rho)z^*] \) independent random variables (selected configurations in randomized rounding phase of R & A framework) where \( X_i \) corresponds to the random variable for the configuration selected in \( i \)-th iteration. Now changing value of any of these random variables may lead to selection of a different configuration \( C' \) in place of configuration \( C \) in the corresponding iteration. Let vectors \( x \) and \( x' \) differ only in \( j \)-th coordinate, i.e., a different configuration \( C' \) is selected in place of configuration \( C \) in \( j \)-th iteration. This might lead to a different set of residual items \( S' \). Then

\[ \phi_i^B(x) - \phi_i^B(x') \leq (|B_i \cap S| - |B_i \cap S'|) \]
\[ \leq \max\{|B_i \cap C|, |B_i \cap C'|\} \]
\[ \leq 1/f(\epsilon) \]

(13)
Here inequality (30) follows from the fact that there can be at most $1/f(\epsilon)$ items of type $B_i$ in a bin as big items are $\geq f(\epsilon)$ in at least one of the dimensions.

Therefore, from independent bounded difference inequality, we get,

$$
P[|B_i \cap S| - E(B_i \cap S) \geq \gamma z^*] \leq e^{-2(\gamma z^*)^2/(\lceil (\ln \rho) z^* \rceil) f(\epsilon)^2)}
$$

where $z^* = LP^*(\tilde{I})$.

Similarly consider the function $\phi^w_k(x) = w(L_k \cap S)$. We get,

$$
\phi^w_k(x) - \phi^w_k(x') \leq (w(L_k \cap S) - w(L_k \cap S'))
\leq \max\{w(L_k \cap C), w(L_k \cap C')\}
\leq 1/f(\epsilon)
$$

Therefore, from independent bounded difference inequality, we get,

$$
P[w(L_k \cap S) - E(w(L_k \cap S)) \geq \gamma z^*] \leq e^{-2(\gamma z^*)^2/(\lceil (\ln \rho) z^* \rceil) f(\epsilon)^2)}
$$

Similarly for wide items we get,

$$
P[h(W_\ell \cap S) - E(h(W_\ell \cap S)) \geq \gamma z^*] \leq e^{-2(\gamma z^*)^2/(\lceil (\ln \rho) z^* \rceil) f(\epsilon)^2)}
$$

For small items, we take $\phi^T(x) = vol(T \cap S)$ and we get $\phi^T(x) - \phi^T(x') \leq 1$ as total volume of small items in a bin is at most 1. Thus,

$$
P[h(W_\ell \cap S) - E(h(W_\ell \cap S)) \geq \gamma z^*] \leq e^{-2(\gamma z^*)^2/(\lceil (\ln \rho) z^* \rceil) f(\epsilon)^2)}
$$

Thus in the asymptotic case (when $z^*$ is sufficiently large, i.e., $> t/\gamma^2 (f(\epsilon))^2$) we can take union bound over all $t$ cases and with high probability for all large item classes, $\phi^B_i(X) - E(\phi^B_i(X)) < \gamma z^*$.

Now take $\gamma = \frac{\epsilon}{5d\rho}$, then from each of $t$ classes, items of volume at most $\gamma z^*$ can be packed into total $\frac{\epsilon z^*}{\rho}$ bins. Let us call $J$ to be these removed elements. Then after removing these items with high probability, the right hand side for each constraint
in LP((\~I \cap S) \setminus J) is at most $1/\rho$ times the right hand side of the corresponding constraint in LP(\~I).

Thus,

$$\text{LP}(\~I \cap S) \leq \text{LP}((\~I \cap S) \setminus J) + \frac{\epsilon_x z^*}{\rho} \leq \frac{(1 + \epsilon_x)}{\rho} \text{LP}^*(\~I)$$  \hspace{1cm} (15)

This gives us that,

$$\mathcal{A}(\~I \cap S) \leq (1 + \epsilon_a)(1 + \epsilon_b) \cdot \text{LP}^*(\~I \cap S) \leq (1 + \epsilon_a)(1 + \epsilon_b) \cdot \frac{(1 + \epsilon_x)}{\rho} \text{LP}^*(\~I) + O(1) \leq (1 + \frac{\epsilon_x}{2}) \text{Opt}(\~I) + O(1) \leq (1 + \frac{\epsilon}{2}) \text{Opt}(I) + O(1).$$ \hspace{1cm} (19)

Here, inequality (16) follows from (10). Inequality (17) follows from (15), inequality (18) follows by taking $(1 + \epsilon) = (1 + \epsilon_a)(1 + \epsilon_b)(1 + \epsilon_x)$ and the last inequality follows by (5).

Thus in step 2 at most $\lceil (\ln \rho) z^* \rceil \leq 1 + (\ln \rho) \cdot \text{Opt}(I)$ bins were used. Medium items are packed in at most $\frac{\epsilon}{2} \cdot \text{Opt}(I)$ bins and at most $(1 + \frac{\epsilon}{2}) \text{Opt}(I) + O(1)$ bins were used to pack the remaining items. This gives the desired $(1 + \ln \rho + \epsilon)$ asymptotic approximation.

The above algorithm can be derandomized using standard techniques as in [11].

### 3.3 A rounding based 1.5-approximation algorithm

In this section we present Jansen-Prädel algorithm [104] that rounds the large dimensions of items into $O(1)$ types of sizes before packing them into bins.

#### 3.3.1 Technique

The algorithm works in two stages. In the first stage, the large dimensions of items in the input instance are rounded to $O(1)$ (specifically $O(\frac{1}{\sqrt{\epsilon}})$) types. By guessing
structures of the rounded items, we guess the rounded values and how many items are rounded to each such value. In the second stage rounded rectangles are packed into bins. The algorithm uses the following structural theorem.

**Theorem 3.3.1.** [104] For any value $\epsilon_c$, with $\frac{1}{\epsilon_c}$ being a multiple of 24, and for any solution that fits into $m$ bins, the widths and the heights of the rectangles can be rounded up so that they fit into $(3/2 + 5\epsilon_c)m + 37$ bins, while the packing of each of the bins satisfies either Property 1.1 (The width of each rectangle in bin $B_i$ of width at least $\epsilon_c$ is a multiple of $\frac{\epsilon_c^2}{2}$) or Property 1.2 (The height of each rectangle in bin $B_i$ of height at least $\epsilon_c$ is a multiple of $\frac{\epsilon_c^2}{2}$).

Using the above structural theorem they show that given any optimal packing they can remove all items intersected with a thin strip in the bin and round one side of all remaining items to some multiple of $\frac{\epsilon_c^2}{2}$. Then they pack the cut items separately to get a packing in at most $(3/2 + \epsilon) \cdot \text{Opt} + O(1)$ bins that satisfy either property 1.1 or property 1.2. After rounding one side of the rectangle, the other side is rounded using techniques similar to those used by [125]. In this version of the algorithm after items are rounded to $O(1)$ types, we can find the optimal packing of these rounded items by brute-force. The algorithm is actually guessing the structure of optimal packing, i.e., rounded values for each item, to use the structural theorem to get a feasible packing in $\leq \left(\frac{3}{2} + \epsilon\right)\text{Opt} + O(1)$ bins. The main structure of their algorithm is described below:

---

**Input:** A set of items $I := \{r_1, r_2, \ldots, r_n\}$ where $r_j \in (0, 1] \times (0, 1]$ for all $j \in [n]$ and set of bins of size $1 \times 1$.

**Output:** An orthogonal packing of $I$.

**Algorithm:**

1. **Guess Opt:** Guess $\text{Opt}$ by trying all values between 1 and $n$. 

---

53
2. For each guessed values of $\text{Opt}$ do

(a) **Classification of rectangles:** Compute $\delta$ using methods similar to [109] to classify rectangles and pack medium rectangles using Next Fit Decreasing Height (NFDH).

(b) **Guessing structures:** Enumerate suitably over all structures (sizes to which items are rounded to and for each size the number of items that are rounded to that size) of the set of big, long, wide rectangles and the set of wide and long containers.

(c) For each guess do Packing:

- Assign big rectangles by solving flow network with the algorithm of Dinic [49];
- Do greedy Assignment of long and wide rectangles into $O(1)$ groups;
- Pack $O(1)$ number of groups of long and wide rectangles into $O(1)$ types of containers using linear program;
- Pack the small rectangles using Next Fit Decreasing Height;
- Pack $O(1)$ types of containers and $O(1)$ types of big rectangles into bins using brute force;

3. Return a feasible packing;

---

### 3.3.2 Details of the Jansen-Prädel Algorithm:

In this section we describe the algorithm in [104], to fit in our framework. For more details on the algorithm, we refer the readers to [156].
3.3.2.1 Binary Search for $\text{Opt}$:

Using binary search between the number of rectangles (upper bound) and total area of the rectangles (lower bound), the algorithm finds the minimum $m$ such that there exists a feasible solution with $(3/2 + \epsilon) \cdot m + O(1)$ bins. For each guess of $\text{Opt}$, we first guess the rounding in the following way and then we pack the rounded items into the bins.

3.3.2.2 Classification of Rectangles:

A value $\delta (\leq \epsilon' = \epsilon/48)$ is selected similar to [109], such that $\frac{1}{\delta}$ is a multiple of 24 and rectangles with at least one of the side lengths between $\delta$ and $\delta^4$ have a small area ($\leq \epsilon' \cdot \text{Opt}$). Now we classify the rectangles into five types:

- **Big**: both width and height is at least $\delta$.
- **Wide**: width is at least $\delta$, height is smaller than $\delta^4$.
- **Long**: height is at least $\delta$, width is smaller than $\delta^4$.
- **Small**: both width and height is less than $\delta^4$.
- **Medium**: either width or height is in $[\delta^4, \delta)$. As medium rectangles are of total size $\leq \epsilon' \cdot \text{Opt}$, they can be packed using NFDH into at most additional $O(\epsilon' \cdot \text{Opt})$ bins. Choose $\epsilon_c = \delta$.

3.3.2.3 Further Classification:

First we will show that given any optimal packing, we can get a packing in at most $(\frac{3}{2} + \epsilon) \cdot \text{Opt} + O(1)$ bins where all large dimensions of items are rounded to $O(1)$ types. Then we will guess the structure of optimal packing to assign items to its rounded type.

Assuming we are given the optimal packing, we can get rounding of one side by using theorem 3.3.1. Now we will find the rounding of the other side using linear
grouping. Let $\text{Opt} = m$ and out of these $m$ bins $B_1, B_2, \ldots, B_m$ are of type 1 (that satisfy property 1.1, i.e., the width and the $x$-coordinate of each rectangle in $B_i$ of width at least $\epsilon_c$ is a multiple of $\frac{\epsilon_c^2}{2}$) and remaining $m_2 (= m - m_1)$ bins $B_{m_1+1}, B_{m_1+2}, \ldots, B_m$ are of type 2 (that satisfy Property 1.2, i.e., the height and the $y$-coordinate of each rectangle in $B_i$ of height at least $\epsilon_c$ is a multiple of $\frac{\epsilon_c^2}{2}$). Thus the widths of big and wide rectangles in bins of type 1 and the heights of big and long rectangles in bins of type 2 are rounded to some multiples of $\delta^2/2$. Let $B_i^w$ and $W_i^w$ are the set of big and wide rectangles respectively that are packed in type 1 bin and widths are rounded to $i \cdot \delta^2/2$ for $i \in \{2/\delta, 2/\delta + 1, \ldots, 2/\delta^2\}$. Similarly, let $B_i^h$ and $L_i^h$ are the set of big and long rectangles respectively that are packed in type 2 bin and heights are rounded to $i \cdot \delta^2/2$. Let $L^w$ and $W^h$ be the set of long rectangles in type 1 bin and set of wide rectangles in type 2 bins respectively. Set of small and medium rectangles are denoted by $M$ and $S$ respectively.

3.3.2.4 Rounding of big, long and wide rectangles:

The rounded widths of rectangles in $B_i^w$ and $W_i^w$ and rounded heights of rectangles in $B_i^h$ and $L_i^h$ are known. In this step we find the rounding of heights of rectangles in $B_i^w$ and $L^w$ and rounding of widths of rectangles in $B_i^h$ and $W^h$ using linear grouping techniques similar to Kenyon-Rémila [125] and introduced by Fernandez de la Vega and Lueker [47].

For any set $B_i^w$, we sort the items $r_1^w, r_2^w, \ldots, r_{|B_i^w|}$ according to non-increasing height, i.e., $h(r_e^w) \geq h(r_f^w)$ for $e \leq f$. Now define at most $\frac{1}{\delta^2}$ subsets $B_{i,j}^w$, each of which contains $\lceil \delta^2 \cdot |B_i^w| \rceil$ rectangles except possibly for the last subset. For any two rectangles $r_A \in B_{i,j}^w$ and $r_B \in B_{i,j_2}^w$ and $j_2 \geq j_1$, $h(r_A) \geq h(r_B)$. We round height of all rectangles in each set $B_{i,j}^w$ to the height of tallest rectangle in the subset (we call it to be the round rectangle of the subset). Set apart the set $B_{i,1}^w$ and for other rectangles in $B_{i,j}^w$ place them on the position of rectangles of $B_{i,j-1}^w$. This is possible as all subsets
(except possibly the last) have same cardinality and all rectangles have same width.

Similarly, sort long rectangles in $L^w$ according to non-increasing height. We divide the set $L^w$ into at most at most $\frac{1}{\delta^2}$ subsets $L^w_1, \ldots, L^w_{\frac{1}{\delta^2}}$ such that every subset has total width $\delta^2 \cdot w(L^w)$. If needed items are sliced vertically to create subsets. For any two rectangles $r_A \in L^w_j_1$ and $r_B \in L^w_j_2$ and $j_2 \geq j_1$, $h(r_A) \geq h(r_B)$. We round the height of each rectangle to the height of the tallest rectangle in it. Apart from $L^w_1$, rectangles of $L^w_j$ are packed on position of rectangles of $L^w_{j-1}$.

The rectangles in $L^w_1, B^{w}_{2/\delta^2} \cdot 1, \ldots, B^{w}_{2/\delta^2 \cdot 1}$ are packed separately into additional bins using NFDH. Note that width of all rectangles in $L^w, B^{w}_{2/\delta^2} \cdot 1, \ldots, B^{w}_{2/\delta^2 \cdot 1}$ is at most $\frac{1}{\delta} \cdot m_1$ as each rectangles has height at least $\delta$. So, $w(L^w_1) + w(B^{w}_{2/\delta^2} \cdot 1, \ldots, w(B^{w}_{2/\delta^2 \cdot 1}) \leq \delta^2 \cdot w(L^w) + w(B^{w}_{2/\delta^2}), \ldots, w(B^{w}_{2/\delta^2 \cdot 1}) \leq \delta^2 \cdot \frac{1}{\delta} \cdot m_1 = \delta \cdot m_1$. Thus the total area of the rectangles in $L^w_1 \cup B^{w}_{2/\delta^2} \cup \cdots \cup B^{w}_{2/\delta^2 \cdot 1}$ is $O(\delta \cdot m_1)$ and thus can be packed in additional $O(\delta \cdot m_1)$ bins using NFDH.

Widths of rectangles in $B^{h}_i, W^{h}$ are rounded in a similar manner.

3.3.2.5 Rounding of containers:

We have not rounded width of long rectangles and heights of wide rectangles. Now we construct rectangular containers for the wide and long rectangles for that purpose. We only show the rounding of containers for type 1 bins. Rounding containers for type 2 bins can be done analogously. Let $C^{w}_L$ be the set of containers for long rectangles and $C^{w}_W$ be the set of containers for wide rectangles in type 1 bins. Define $2/\delta^2$ vertical slots of width $\delta^2/2$ in each type 1 bin $B_i$. A long container is part of a slot that contains at least one long rectangle, and the container is bounded at the top and bottom by a wide or big rectangle or the boundary (ceiling or floor) of the bin. There can be at most $(1/\delta - 1)$ long containers in a slot. Thus there are at most $O(\delta^3)$ long containers per bin.
Next, construct wide containers by extending upper and lower edges of big rectangles and long containers in both directions till they hit another big rectangles, long container or boundary (left or right side of bin). Wide and small rectangles are horizontally cut by these lines. As there are \(O(\delta^3)\) big rectangles and long containers, there are \(O(\delta^3)\) wide containers in \(\mathcal{B}_i\). This way any packing in optimal bin is transformed into a packing of big rectangles and long and wide containers. All small rectangles are packed fractionally in the long and wide containers.

Now we do the rounding of containers. Heights of all containers in \(C^w_w\) are rounded down to nearest multiple of \(\delta^4\) cutting the uppermost wide and short rectangles. There are \(O(1/\delta^3) \cdot m_1\) wide containers and small rectangles have height less than \(\delta^4\). Thus the cut wide and short rectangles are packed using NFDH in additional \(O(\delta \cdot m_1)\) bins. For long containers we remove the short rectangles and push all long rectangles vertically down till they touch top of another rectangle or boundary. Then we round down the heights to either nearest multiple of \(\delta^4\) or combination of rounded heights of the long rectangles. Note that these heights are rounded down to although large but still \(O(1)\) number of types. Total area loss for each container is \(O(\delta^4 \cdot \delta^2/2)\) and number of long containers is \(O(1/\delta^3)\). So in the reduced container we pack the small items till we can and the remaining small rectangles are packed into additional \(O(\delta^3 \cdot m_1)\) bins using NFDH.

Similarly long containers can be constructed for the additional bins that are used to pack items of \(L^w_w\). These bins will have at most \((2\delta \cdot m_1 + 1) \cdot 2/\delta^2\) long containers of width \(\delta^2/2\) and height 1. Note that there are \(O(1/\delta^2)^{1/\delta}\) possible height of containers, which can be reduced to \(O(1/\delta^2)\) heights using linear grouping and loosing only a small constant.

Thus at the end of the rounding of containers, the containers have the following properties:

2.1. There are at most \(O(1/\delta^3) \cdot m_1\) wide containers in \(C^w_w\) with width of multiple of
\( \delta^2 \) and height of a multiple of \( \delta^4 \).

2.2. There are at most \( O(1/\delta^3) \cdot m_2 \) long containers in \( C_L^h \) with height of multiple of \( \delta^2 \) and width of a multiple of \( \delta^4 \).

2.3. There are at most \( O(1/\delta^3) \cdot m_2 \) wide containers in \( C_W^h \) with \( O(1/\delta^2) \) different widths (either a multiple of \( \delta^4 \) or a combination of rounded width of wide rectangles in \( W^h \)) and height \( \delta^2/2 \).

2.4. There are at most \( O(1/\delta^3) \cdot m_2 \) long containers in \( C_L^w \) with \( O(1/\delta^2) \) different heights (either a multiple of \( \delta^4 \) or a combination of rounded height of long rectangles in \( L^w \)) and width \( \delta^2/2 \).

At the end of the rounding step we get the following theorem.

**Theorem 3.3.2.** [104] Given an optimal packing \( I \) into \( m \) bins, it is possible to round the widths and heights of the long, wide and big rectangle to \( O(1) \) types such that it fits in at most \((3/2 + O(f_1(\delta)))m + O(f_2(\delta))\) bins for some functions \( f_1 \) and \( f_2 \) and these bins satisfy either property 1.1 or property 1.2. Furthermore the heights of long and big rectangles in \( L^w \) and \( B^w \), widths of wide and big rectangles in \( W^h \) and \( B^h \) are rounded up to \( O(1/\delta^2) \) values. The wide and long rectangles are sliced horizontally and vertically respectively and packed into containers satisfying properties 2.1-2.4 and small rectangles are packed fractionally into the wide and long containers. Medium rectangles are packed separately in \( O(\delta) \) bins.

### 3.3.2.6 Transformation of rectangles:

Now we guess the structure of the optimal packing for the assignment of rectangles to the rounded rectangles.

First we have to determine whether width or height of a big rectangle is rounded to a multiple of \( \delta^2/2 \). We guess the cardinality of sets \( B_i^w \) and \( B_i^h \) for \( i \in \{2/\delta, 2/\delta + 1, \cdots, 2/\delta^2\} \). This can be done by choosing less than \( 2 \cdot (2/\delta^2) \) values out of \( n \) and note that this is polynomial in \( n \). For each such guess we also guess \( 2 \cdot (2/\delta^2) \cdot (1/\delta^2) \)
round rectangles out of \( n \) rectangles. These values give us the structure of subsets as discussed in the rounding of big rectangles. Now to find the assignment of big rectangles to these subsets, we create a directed flow network \( G = (V, E) \). First we create source(s) and target node(t). For each rectangle \( r \in I \), we create a node and add an edge from \( s \) to \( r \) with capacity one. Next we create nodes for all subsets \( B^w_{i,j} \) and \( B^h_{i,j} \) and add an edge from \( r \) to \( B^y_{i,j} \) of capacity one if \( r \) might belong to \( B^y_{i,j} \) where \( y \in \{w, h\} \). Next add edges between nodes corresponding to subsets and the target node of infinite capacity. Now apply Dinic’s algorithm [49] or any other flow algorithm to find if there is a \( s - t \) flow of value same as the number of big rectangles. If there exists such a flow, we get a valid assignment of big rectangles into subsets. On the other hand, if there is no such flow then continue to other guesses.

Next we need to transform the wide and long rectangles. First we need to decide whether a wide rectangle belong to type 1 or type 2 bins. The case for long rectangle is analogous. Note that in the linear grouping of wide rectangles in \( W^h \), \( 1/\delta^2 \) subsets were created. The total height of all rectangles in \( W^h \) is bounded by \( n \cdot \delta^4 \). So we approximately guess the total height of \( W^h \) by choosing some \( t \in \{1, 2, \cdots, n\} \) so that \( t \cdot \delta^4 \leq h(W^h) \leq (t + 1) \cdot \delta^4 \). As all subsets \( W^h_1, W^h_2, \cdots, W^h_{1/\delta^2} \) have same height, each subset will have height \( h(W^h) \cdot \delta^2 \). We also guess the height of rectangles to which all rectangles in each subset are rounded to. This can be done by choosing \( 1/\delta^2 \) rectangles out of \( n \) rectangles. Thus we can approximately guess the structure of the rectangles in \( W^h \). Remaining wide rectangles are in \( W^w \) i.e., their width are rounded to next multiple of \( \delta^2 / 2 \). We guess approximately the total height of the rectangles in \( W^w_{2/\delta^2}, \cdots, W^w_{4/\delta^2} \) by choosing \( (2/\delta^2 - 2/\delta + 1) \) integral values \( t_k \) such that \( t_k \cdot \delta^4 \leq h(W^w_j) < (t_k + 1) \cdot \delta^4 \). Thus we can guess structure of all subsets of wide rectangles and we need to assign the wide rectangles into these subsets. For the assignment we sort the wide rectangles in non-increasing order. We assign wide rectangles greedily into all sets \( W^w_k \in W^w \) starting from \( W^w_{2/\delta^2} \) till total height exceeds
(\(t_k + 1\) \(\cdot \delta^4\)) for each set \(W^w_k\). The remaining rectangles are similarly greedily assigned into sets \(W^r_1, \ldots, W^r_{1/\delta^2}\). It is easy to show that for the right guesses of \(t_k\) values, all rectangles will have a valid assignment. Afterwards we remove the shortest rectangles in each subset to reduce the height to at most \(t_k \cdot \delta^4\). It can be shown that the total height of these removed wide rectangles is \(O(\delta^2)\) and thus can be packed into \(O(1)\) additional bins.

### 3.3.2.7 Construction of containers:

Here we describe the construction of long and wide containers that are placed in type-1 bins. The construction of long and wide containers that are placed in type-2 bins, is analogous.

Each wide container in \(C^w_W\) has height of a multiple of \(\delta^4\) and width of multiple of \(\delta^2/2\). Hence we can guess \(n^w_{i,j}\), number of wide containers that has width \(i\delta^2/2\) and height \(j \cdot \delta^4\) by choosing \(1/\delta^4 \cdot 2/\delta^2\) values out of \(n\).

Similarly long containers in \(C^w_L\) have same width and \(O(1/\delta^2)\) types of heights (either a combination of rounded heights of long rectangles or a multiple of \(\delta^4\)) that we can guess.

### 3.3.2.8 Packing long and wide rectangles into containers

There are four cases: packing of long rectangles into long containers in type 1 bins, packing of long rectangles into long containers in type 2 bins, packing of wide rectangles into wide containers in type 1 bin and packing of wide rectangles into wide containers in type 2 bins. Here let us only consider the wide containers in type 1 bins, other cases can be handled similarly. As there are \(O(\delta^3)\) types of wide containers and \(O(\delta^2)\) types of wide rectangles, we try out all possible way wide rectangles can be put into wide containers by brute force. The running time can be further improved if we use linear programs as in Kenyon-Rémilla [125]. Then we add back the small rectangles using NFDH in the empty regions of the \(O(\delta^3)\) types of containers till we
3.3.2.9 Complete packing

Now we have big rectangles and containers of $O(1)$ type, thus there are $O(1)$ number of possible configurations of packing of big rectangles and containers into bins. We try out all configurations by brute force to find the optimal packing of big rectangles and containers. Then we add back the small rectangles using NFDH in the empty regions of the bin till we can and the remaining small rectangles are packed into additional bins.

3.3.3 Analysis

In the rounding step, separate packing of rectangles in $L^w_1, B^w_{2/\delta,1}, \cdots, B^w_{2/\delta^2,1}, W^h_1, B^h_{2/\delta,1}, \cdots, B^h_{2/\delta^2,1}$ need at most $O(\delta \cdot \text{Opt})$ additional bins. In rounding containers the cut wide, long and small rectangles are packed in additional $O(\delta \cdot \text{Opt})$ bins. Packing of medium and remaining small rectangles take $O(\delta \cdot \text{Opt})$ bins. Removed wide rectangles in the step of transformation of wide rectangles require $O(1)$ extra bins. So using the structural theorem total $(3/2 + O(\delta))\text{Opt} + O(\delta)$ bins are sufficient.

The running time of the steps are given as follows. The binary search requires $O(\log n)$ time. Computing $\delta$ in a method similar to [109] takes $O(n/\epsilon)$ time. For the structure of the set of big rectangles, we guess $O(1/\delta^2)$ values out of $n$ to guess the cardinality of the sets and for such guess, $O(1/\delta^4)$ round rectangles are guessed. Similarly, we get the structure of wide and long rectangles, we guess $O(1/\delta^3)$ values out of $n$. Structure of long and wide containers require guessing $O(1/\delta^6)$ values out of $n$ and guessing $O(1/\delta^2)$ values out of $O(1/\delta^4 + (1/\delta^2)^{1/\delta})$ respectively. Solving the flow network takes $O(n^3)$ time. Assignment of wide and long rectangles into groups will take $O(n \log n)$ time. The running time for packing containers and big rectangles using the brute force method is a large, however, constant in triple exponential in $\delta$. It can be reduced using integer programs of Kannan et al. [119]. Packing medium
and small rectangles using NFDH require $O(n \log n/\delta^3)$ time. Totally the running time is bounded by $O(n^{h_1(1/\epsilon)} \cdot h_2(1/\epsilon))$, where $h_1, h_2$ are polynomial functions. Thus the total running time is polynomial for fixed $\epsilon$.

### 3.3.4 Bin packing with rotations

Bin packing is rotation is almost similar to the packing without rotation. In this case we only have bins with a packing that satisfy property 1.1. Remaining rounding steps are analogous to the versions with rotations. The step of transformation of rectangles, however, is slightly different when we allow rotations. For big rectangles, in the flow network we connect a big rectangle with all subsets that can contain the rectangle before and after rotating by $90^\circ$. On the other hand, for transformation of wide and long rectangles, we approximately guess $w(L^w)$ and the heights of the sets $W_{2/\delta}^w, \ldots, W_{2/\delta^2}^w$ and the height of the cut rectangles in $L^w$. Now we can rotate all long rectangles to have only wide rectangles and greedily assign them to the wide rectangles in $W_{2/\delta}^w, \ldots, W_{2/\delta^2}^w$. The remaining wide rectangles are rotated back and assigned to the sets $L_{1}^w, \ldots, L_{1/\delta^2}^w$. The analysis is also similar, however, gives slightly better constants in the approximation ratio.

### 3.4 Lower bound for rounding based algorithms

In this section, we describe some limitations of rounding based algorithms.

**Theorem 3.0.11.** (restated) Any rounding algorithm that rounds at least one side of each large item to some fixed constant independent of the problem instance (let us call such rounding input-agnostic), cannot have an approximation ratio better than $3/2$.

**Proof.** Consider an input-agnostic algorithm $\mathcal{A}$ that rounds at least one side of each large item to one of the values $c_1, c_2, \ldots, c_z$, that are chosen independent of the input instance. Let $i$ and $j$ be such that $c_i < 0.5 \leq c_{i+1}$ and $c_{j-1} \leq 0.5 < c_j$. Let
\( f = \min\{0.5 - c_i, c_j - 0.5\} \). Here we assume the algorithm rounds identical items with the same height and width to same types.

Now consider an optimum packing using \( m = 2k \) bins where each bin is packed as in Figure 7, for some fixed \( x \in (0, f) \). Under the rounding, an item \( (1/2 + x) \times (1/2 - x) \) is rounded to either \( (1/2 + x) \times (c_i + 1) \) (let us call such items of type P) or to \( (c_j) \times (1/2 - x) \) (let us call such items of type Q). Similarly, each item \( (1/2 - x) \times (1/2 + x) \) is rounded to either \( (c_i + 1) \times (1/2 + x) \) (call these of type R) or
to \((1/2 - x) \times (c_j)\) (call these of type S).

Let us first consider the easy case when \(c_{i+1} > 1/2\). It is easily checked that in this case, any bin can contain at most 2 rounded items: (i) either a P-item and a S-item or (ii) a Q-item and a R-item. See, for example Figure 8. This implies that a 2-approximation is the best one can hope for, if 1/2 is not included among the \(c_1, c_2, \ldots, c_z\).

![Figure 9: Configuration \(\{P, P, S\}\)](image)

![Figure 10: Configurations \(\{R, R, Q\}\)](image)

We now consider the case when \(c_{i+1} = 1/2\). We claim that the possible bin configurations are a) \([\{P, P, S\} \text{ and } \{S, S\}]\), which happens when the items are rounded to types P and S. See Figure 9. Or, b) \([\{R, R, Q\} \text{ and } \{Q, Q\}]\), which happens when items are rounded to types R and Q. See Figure 10. Furthermore, the remaining two cases can be ignored. That is, when items are rounded to type P and R or when items are rounded to type Q and S, as in these cases at most two items can be packed in a bin.
So, let us consider case (a). The proof for case (b) is analogous. Let \( x_1 \) and \( x_2 \) denote the number of configurations of type \{ P,P,S \} and \{ S,S \} respectively. Then we get the following configuration LP:

\[
\begin{align*}
\text{Min} & \quad x_1 + x_2 \\
\text{s.t.} & \quad 2x_1 \geq 4k \\
& \quad x_1 + 2x_2 \geq 4k \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

The dual is:

\[
\begin{align*}
\text{Max} & \quad 4k(v_1 + v_2) \\
\text{s.t.} & \quad 2v_1 + v_2 \leq 1 \\
& \quad 2v_2 \leq 1 \\
& \quad v_1, v_2 \geq 0
\end{align*}
\]

A feasible dual solution is \( v_1 = 0.25, v_2 = 0.5 \). This gives dual optimal as \( \geq 3k \).

Thus the number of bins needed is \( \geq 3k = 3m/2 \). \( \square \)

This in particular implies that to beat 3/2 one would need a rounding that is not input-agnostic, or which rounds identical items with the same height and width to different types, sometimes rounded by width and sometimes by height.

We also note that 4/3 is the lower bound for any rounding algorithm that rounds items to \( O(1) \) types. This seems to be a folklore observation, but we state it here for completeness.

**Theorem 3.4.1.** Any algorithm that rounds items to \( O(1) \) types cannot achieve better than 4/3 approximation.

**Proof.** Consider the packing in Figure 7. Assume there is an optimal packing of \( m = 3k \) bins where each bin is having similar packing as in Figure 7 for \( m \) different
values of $x \in (0.001, 0.01]$. Note that the sum of the height and width is exactly 1 for each rectangle. If we use rounding to $O(1)$ items, then for all but $O(1)$ items $i$, $w(i) + h(i)$ exceed 1. Without loss of generality, we assume each item touches boundary. Otherwise for these set of items, we can extend sides vertically and horizontally so that it touches boundary or another item. As the total sum of the side lengths of a bin is 4 and each item has intersection with boundary of length $> 1$, we can only pack 3 rounded items in a bin. Thus $4m = 12k$ items can be packed in at least $4k - O(1) = 4/3m - O(1)$ bins. This example packing is particularly interesting as it also achieves the best known lower bound for guillotine packing, i.e., $4/3$. \[\square\]

### 3.5 Final Remarks

The approach for the R&A framework described here applies directly to wide variety of algorithms and gives much simpler proofs for previously considered problems (e.g., vector bin packing, one dimensional bin packing) [11]. As rounding large coordinates to $O(1)$ number of types is by far the most widely used technique in bin-packing type problems, we expect wider applicability of this method. In fact in the next chapter we will extend this method to further improve the approximation for vector packing.

Moreover, improving our guarantee for 2-D BP will require an algorithm that is not input-agnostic. In particular, this implies that it should have the property that it can round two identical items (i.e., with identical height and width) differently. One such candidate is the guillotine packing approach [14]. It has been conjectured that this approach can give an approximation ratio of $4/3$. One way to show this would be to prove a structural result bounding the gap between guillotine and non-guillotine packings. At present the best known upper bound on this gap is $T_\infty \approx 1.69$ [22].
Chapter IV

VECTOR BIN PACKING

In this chapter, we consider $d$-dimensional ($d$-D) vector bin packing (VBP) problem, studied extensively starting from the classical work of Garey et al. in 1976 [76]. Here we are given a set of $d$-dimensional vectors $I := \{v_1, \ldots, v_n\}$ where each vector $v_i \in [0, 1]^d$. The goal is to pack all vectors into minimum number of unit vector bins so that for every bin in each dimension the sum of packed vectors should not exceed one, i.e., $\| \sum_{v_i \in B} v_i \|_{\infty} \leq 1$ for each bin $B$ in the packing. Here $d$ is an arbitrary constant number. The problem is of great significance in resource constrained scheduling [70] and appeared also recently in virtual machine placement in cloud computing [152]. Even in two dimensions (2-D), it has novel applications in layout design, logistics, loading and scheduling problems [175, 184, 89, 171, 163]. If the vectors are in $\{0, 1\}^d$, then the problem is called $\{0, 1\}$ vector packing.

Our main result is improved approximation for multidimensional vector packing. In particular we show the following two theorems for 2-D and $d$-D respectively.

**Theorem 4.0.1.** For any small constant $\epsilon > 0$, there is a polynomial time algorithm with an asymptotic approximation ratio of $(1 + \ln(1.5) + \epsilon) \approx (1.405 + \epsilon)$ for 2-D vector packing.

**Theorem 4.0.2.** For any small constant $\epsilon > 0$, there is a polynomial time algorithm with an asymptotic approximation ratio of $(1.5 + \ln(d/2) + \frac{2}{d} + \epsilon)$ for $d$-D vector packing.

The main idea behind Theorem 4.0.1 and 4.0.2 is to use R & A framework along with structural property of vector packing and multi-objective multi-budget matching. At present, R & A framework gives the best approximation for vector packing as...
well as 2-D GBP. In this framework given a packing problem, some bins are packed using a configuration LP. Remaining items are then packed using a $\rho$-approximation subset-oblivious algorithm to get a $(1 + \ln \rho)$ approximation for the original problem. However, these subset-oblivious algorithms round the large dimensions of items to constant number of item types. In fact, all known algorithms that we know of for bin packing type problems (except R & A framework) explicitly or implicitly uses these kinds of rounding. However, we show that one can not obtain better than $d$ approximation using constant rounding based algorithms, thus giving a barrier of $(1 + \ln d)$ of present R & A framework.

We circumvent this problem for $d$-dimensional vector packing, by showing a structural property of vector packing that any optimal packing of $m$ bins can be transformed into $\lceil \frac{(d+1)m}{2} \rceil$ bins such that the bins are of two types:
1. Either a bin contains at most two big items, or
2. The bin has slack in $d - 1$ dimensions (i.e., for each of these $(d - 1)$ dimensions, the sum of all vectors in the bin is at most $1 - \delta$ for some constant $\delta$ that we will fix later).

Then we show that we can pack all items in type 2 bins almost optimally using resource augmentation. This resource augmented packing itself is an interesting observation. Due to pathological nature of lower bound examples for bin packing, resource augmentation is well studied. For geometric bin packing similar results were known for resource augmentation packing [12, 15]. We specifically show the following.

**Theorem 4.0.3.** There is a polynomial time algorithm for packing vectors into at most $(1 + \epsilon + \frac{\epsilon^2}{2d}) \cdot \text{Opt} + 1$ bins with $\epsilon$ resource augmentation in $(d - 1)$ dimensions (i.e., bins have length $(1 + \epsilon)$ in $(d - 1)$ dimensions and 1 in the other dimension), where $\text{Opt}$ denotes minimum number of unit vector bins to pack these vectors.

In Theorem 4.0.3 we adapt rounding techniques from [28] for $(d - 1)$ dimensions and for the last dimension we use linear grouping to round large dimensions of items.
Then we show that we can approximately separate items in type 1 and type 2 bins using a multi-objective multi-budget matching algorithm from [29]. As a corollary this gives tight \((1.5 + \xi)\) absolute approximation guarantee. This is interesting as even for 1-D bin packing, the lower bound for absolute approximation is \(3/2\). For absolute approximation, it has been an interesting open question to reach \(3/2\) absolute approximation for other generalizations of bin packing such as 2-D strip packing [93].

**Theorem 4.0.4.** There is a polynomial time algorithm with an absolute approximation ratio of \((1.5 + \xi)\) for 2-D vector packing.

Finally, we combine these structural properties, resource augmented packing and matching algorithm with R & A framework to obtain better approximation as in Theorem 4.0.1 and 4.0.2. We believe that these techniques might be useful in many similar packing or covering related problems. Our result on asymptotic approximation for 2-D vector packing also matches the present best asymptotic approximation for 2-D GBP and the result on absolute approximation is better than present best asymptotic approximation for 2-D GBP, which is also interesting to understand the hardness of different variants of bin packing.

**Organization.** The organization of the paper is as follows. In section 4.1, we discuss related previous works. In section 4.2, we present the preliminaries. In section 4.3, we consider packing when resource augmentation is allowed in \((d - 1)\) dimensions. In section 4.4, we present the \((d + 1)/2\) approximation algorithm for \(d\)-D vector packing and some related structural properties of vector packing. Also we present the algorithm with absolute approximation guarantee of \(3/2\). In section 4.5 and section 4.6, we present the improved algorithms using R & A framework for 2-D and \(d\)-dimensions respectively.
4.1 Prior Works

In this section we survey the previous work on vector packing and its variants.

4.1.1 Offline Results:

The first paper to obtain an APTAS for 1-D bin packing by Fernandez de la Vega and Lueker [47] imply a \((d+\epsilon)\) approximation for vector packing problem. Woeginger [187] showed that there exists no APTAS even for \(d = 2\) unless \(P = NP\). However some restricted class of vectors may still admit an APTAS. For example, consider the usual partial order on \(d\) dimensional vectors where \((x_1, x_2, \ldots, x_d) \prec (y_1, y_2, \ldots, y_d)\) if and only if \(x_i \leq y_i\) for all \(i \in [d]\). In Woeginger’s gadget for lower bound, the items are pairwise incompatible. The opposite extreme case, when there is a total order on all items, is easy to approximate. In fact, a slight modification of de la Vega and Lueker [47] algorithm yields an APTAS for subproblems of \(d\) dimensional vector packing with constant Dilworth number. After nearly twenty years, offline results for the general case was improved by Chekuri and Khanna [28]. They gave an algorithm with asymptotic approximation ratio of \(1+\epsilon d+H_{1/\epsilon}\) where \(H_k = 1+1/2+\cdots+1/k\), the \(k\)’th Harmonic number. Considering \(\epsilon = 1/d\), they show that for fixed \(d\), vector bin packing can be approximated to within \(O(\ln d)\) in polynomial time. Bansal et al. [11] then introduced the Round and Approx framework and the notion of subset oblivious algorithm and improved it further to \(1 + \ln d\). Both these algorithms run in time that is exponential in \(d\) (or worse). Yao [188] showed that no algorithm running in time \(o(n\log n)\) can give better than a \(d\)-approximation.

For arbitrary \(d\), Chekuri-Khanna [28] showed vector bin packing is hard to approximate to within a \(d^{1/2-\epsilon}\) factor for all fixed \(\epsilon > 0\) using a reduction from graph coloring problem.

The two dimensional case has received special attention. Kellerer and Kotov [124] designed an algorithm for 2-D vector packing with worst case absolute approximation
ratio as 2. On the other hand there is a hardness of 3/2 for absolute approximation ratio that comes from the hardness of 1-D bin packing.

4.1.2 Online Results

A generalization of First Fit algorithm gives $d + \frac{7}{10}$ competitive ratio for the online version by Garey et al. [74]. Galamobos et al. [72] showed a lower bounds on the performance ratio of online algorithms that tends to 2 as $d$ grows. The gap persisted for a long time[73], in fact it was conjectured in [60] that the lower bound is super constant, through sublinear.

Recently Azar et al. [5] settled the status by giving $\Omega(d^{1-\epsilon})$ information theoretic lower bound using stochastic packing integer programs and online graph coloring. In fact their result holds for arbitrary bin size $B \in \mathbb{Z}^+$ if the bin is allowed to grow. In particular, they show that for any integer $B \geq 1$, any deterministic online algorithm for VBP has a competitive ratio of $\Omega(d^{\frac{1}{B-\epsilon}})$. For $\{0, 1\}$ VBP the lower bound is $\Omega(d^{\frac{1}{B+1-\epsilon}})$. They also provided an improved upper bound for $B \geq 2$ with polynomial time algorithm for online VBP with competitive ratio: $O(d^{1/(B-1)} \log d^{B/(B+1)})$, for $[0, 1]^d$ vectors and $O(d^{1/B} \log d^{(B+1)/B})$, for $\{0, 1\}^d$ vectors.

4.1.3 Heuristics

Heuristics for 2-D VBP was studied in details by Spieksma [175], who mentions applications in loading, scheduling, and layout design, considers lower bounding and heuristic procedures, and uses them within a branch-and-bound scheme. Here, upper bounds are derived from a heuristic, adapted from the first fit decreasing (FFD)-rule for the bin-packing problem. To find good lower bounds, properties of pairs of objects are investigated. Han et al. [89] present heuristic and exact algorithms for a variant of 2-D VBP, where the bins are not identical. Caprara and Toth [24] also studied 2-D VBP. They analyze several lower bounds for 2-D VBP. In particular, they determine an upper bound on the worst-case performance of a class of lower bounding procedures.
derived from classical bin packing problem. They also prove that the lower bound associated with the huge linear programming relaxation dominates all the other lower bounds. They then introduce heuristic and exact algorithms, and report extensive computational results on several instance classes, showing that in some cases the combinatorial approach allows for a fast solution of the problem, while in other cases one has to resort to the huge formulation for finding optimal solutions. Chang et al. [26] had proposed a greedy heuristic named hedging. Otoo et al. [151] studied the 2-D VBP where each item has 2 distinct weights and each bin has 2 corresponding capacities and have given linear time greedy heuristics. An interesting application of the 2-D VBP problem is studied by Vercruyssen and Muller [184]. The application arises in a factory where coils of steel plates (the objects), each having a certain physical weight and height, have to be distributed over identical furnaces (the bins) with a limited capacity for height and weight. Another application of the problem is described by Sarin and Wilhelm [163], in the context of layout design. Here, a number of machines (the objects) have to be assigned to a number of robots (the bins), with each robot having a limited capacity for space, as well as a limited capacity for serving a machine. Many of these heuristics are tailor-made for 2-D.

For the general case Stillwell et al. [178] studied variants of FFD concluding that the algorithm FFDAvgSum is best in practice. They also show that genetic algorithms don’t perform well. Panigrahy et al. [152] systematically studied variants of the First Fit Decreasing (FFD) algorithm. Inspired by bad instances for FFD-type algorithms, they propose new geometric heuristics that run nearly as fast as FFD for reasonable values of $n$ and $d$.

### 4.2 Preliminaries

Let $I := \{v_1, v_2, \ldots, v_n\}$ be a set of $d$-dimensional vectors where $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ and $v_{ij} \in [0, 1]$ for $j \in [d]$. Here $[d]$ denotes the set $\{1, 2, \ldots, d\}$. We will use the
term dimension and coordinate interchangably. If $\alpha$ is a vector, we call a bin $B_i$ is $\alpha$-packed if $\sum_{v_j \in B_i} v_j^\ell \leq \alpha^\ell$ for all $\ell \in [d]$. For any bin $B_i$, it has slack $\delta$ in dimension $k$ if $\sum_{v \in B_i} v^k \leq (1-\delta)$. For a set of vector $S$, $\sigma_S$ be the vector denoting coordinate wise sum of all vectors in $S$, i.e., $\sigma_S = \sum_{v_j \in S} v_j$. We denote $\sigma_S \leq v$ if $\sigma_S^\ell \leq v^\ell$ for all dimensions $\ell \in [d]$. Thus $S$ is a feasible configuration if $\sigma_S \leq 1$ where $1 = (1, \ldots, 1)$ is the unit vector.

We now classify the items into the following two classes based on their sizes. Here $\beta$ is a parameter depending on $\epsilon$ and $d$ and will be fixed later.

- Large or big items ($\mathcal{L}$) : at least one coordinate has size $\geq \beta$, i.e., $v \in \mathcal{L}$ iff $||v||_{\infty} \geq \beta$.
- Small items ($\mathcal{S}$) : all coordinates have size $< \beta$, i.e., $v \in \mathcal{S}$ iff $||v||_{\infty} < \beta$.

Let $\mathcal{L}_B$ be the set of big items and $\mathcal{S}_B$ be the set of small items in a bin $B$. We call a configuration of big items in $B$ to be the vector $\sum_{v_i \in \mathcal{L}_B} v_i$. Similarly we call a configuration of small items to be the vector $1 - \sum_{v_i \in \mathcal{L}_B} v_i$.

Let $(c_1, c_2, \ldots, c_d)$ be a $d$-tuple of integers such that $c_i \in [0, \lceil \frac{1}{\epsilon} \rceil]$. Each such distinct tuple is called to be a capacity configuration and approximately describes how a bin is filled. A packing of a set of vectors $V$ is said to be viable for a capacity configuration $(c_1, \ldots, c_d)$ if $\sigma_V \leq \epsilon \cdot (c_1, \ldots, c_d)$. There are $r_c := (1 + \lceil \frac{1}{\epsilon} \rceil)^d$ possible types of capacity configurations.

Now we define a bin configuration to be a $r_c$-tuple of integers $(m_1, m_2, \ldots, m_{r_c})$ where $m_i \in [m]$ and $\sum_i m_i = m$. Bin configuration approximately describes the structure of all packed bins, i.e., there are $m_i$ bins with capacity configuration $c_i$ for $i \in [r_c]$. Total number of bin configurations is $O(m^{r_c})$. A packing of a set of vectors $V$ is said to be viable to a bin configuration $M = (m_1, m_2, \ldots, m_{r_c})$ if there is a packing of items in $V$ into $m$ bins such that the bins can be partitioned into sets $B_1, B_2, \ldots, B_{r_c}$ and there are $m_i$ bins in $B_i$ that are viable to capacity configuration.
4.2.1 Limitations of Round and Approx Framework

Theorem 4.2.1. Any algorithm that rounds large dimensions (with value more than \( \epsilon \), where \( \epsilon > 0 \) is a given accuracy parameter) of items to \( O(1) \) types can not achieve better than \( d \) approximation for \( d \)-D vector packing.

Proof. Let \( \mathcal{A} \) be an algorithm that rounds large dimensions of items to \( r \) (a constant) types. Let \( t \in \mathbb{N} \) be another large constant. Consider following \( n = dtrk \) items that can be packed into \( m = n/d = trk \) bins \( \mathcal{B}_1, \ldots, \mathcal{B}_m \) in the optimal packing. There are \( tr \) types of bin in the optimal packing and each type contains \( m/tr = k \) bins. Each \( i \)'th type bin contains \( d \) items \( v_{i1}, v_{i2}, \ldots, v_{id} \) such that \( v^k_{ik} = 1 - \epsilon_i \) and \( v^l_{ik} = \epsilon_i/(d-1) \) for \( k = 1, 2, \ldots, d \) and \( l \in [d] \setminus \{k\} \) where \( \epsilon_i = \frac{\xi_d}{d^{r}} \) and \( \xi \) is a small constant, i.e., in each bin of type \( i \) for each coordinate \( k \) there is exactly one item \( v_{ih} \) with value \((1 - \epsilon_i)\) in that coordinate ( \( v_{ik} \) has length \( \frac{\epsilon_i}{(d-1)} \) in all other coordinates) and \((d-1)\) other items with value \( \epsilon' \) in coordinate \( k \).

Now let the rounding algorithm rounds the large coordinates of items to constant types \( 1 - \delta_1, 1 - \delta_2, \ldots, 1 - \delta_r \) such that \( 1 - \epsilon_{\ell_h} \leq 1 - \delta_h < 1 - \epsilon_{\ell_{h+1}} \) for \( h \in [r-1] \) and \( 1 - \epsilon_{tr} \leq 1 - \delta_r \). Then apart from possibly the items in bins of type \( \ell_h \) for \( h \in [r-1] \) and of type \( tr \), for all other items large coordinates \((1 - \epsilon')\) are rounded to some other value \((1 - \bar{\epsilon})\) such that \( \epsilon' \geq d\bar{\epsilon} \). Let us call the bins that are not of type \( \ell_h \) for \( h \in [r-1] \) and of type \( tr \) to be complex bins. We claim the following.

Claim 4.2.2. No two rounded items from two complex bins, can be packed into one bin.

Proof. Let \( v_{ix} \) and \( v_{jy} \) be two items from complex bins of type \( i \) and \( j \) where \( i \leq j \). From definition of \( \epsilon_i \),

\[
\epsilon_i = d^{j-i} \epsilon_j
\]  

(20)
Let $\mathbf{v}_x$ and $\mathbf{v}_y$ be the rounded vector, then
\[
\mathbf{v}_y^y + \mathbf{v}_x^y \geq (1 - \epsilon_j) + \frac{\epsilon_i}{d-1} \\
\geq (1 - \frac{\epsilon_j}{d}) + \frac{\epsilon_i}{d-1} \quad \left[ \therefore \epsilon_j \geq d \epsilon_j \right] \\
\geq (1 - \frac{\epsilon_j}{d}) + \frac{\epsilon_j}{d-1} > 1 \quad \left[ \therefore i \leq j \quad \text{and} \quad d > 1 \right]
\]
Thus two rounded items cannot be packed together in a bin.

Hence, we need at least $(md - rkd)$ bins to pack all items. This implies a lower bound of approximation for these class of algorithms as $\frac{md-rkd}{m} = d(1-\frac{rk}{trk}) = d(1-\frac{1}{t})$.
Thus it gives asymptotic approximation hardness for rounding based algorithms as $d$.

Therefore improving $1 + \ln d$ for $d$-dimension is not possible by just using R & A framework with a $O(1)$ rounding based algorithm to pack the residual items.

### 4.2.2 Multi-objective/multi-budget matching problem

In *Multi-objective/multi-budget matching problem*, we are given a graph $G := (V, E)$, $k$ linear functions (called demands) $f_1, f_2, \ldots, f_k : 2^E \rightarrow \mathbb{R}^+$, $\ell$ linear functions (called budgets) $g_1, g_2, \ldots, g_\ell : 2^E \rightarrow \mathbb{R}^+$, and the goal is to find a matching $\mathcal{M}$ satisfying $f_i(\mathcal{M}) \geq D_i$ for all $i \in [k]$ and $g_i(\mathcal{M}) \leq B_i$ for all $i \in \ell$.

Chekuri et al. [29] gave an algorithm that solve the problem nearly optimally.

**Theorem 4.2.3.** [29] For any constant $\gamma > 0$ and any constant number of demands and budgets $k + \ell$, there is a polynomial time algorithm which for any feasible instance of multi-objective matching finds a feasible solution $S$ such that

- Each linear budget constraint is satisfied: $g_i(S) \leq B_i$.
- Each linear demand is nearly satisfied: $f_i(S) \geq (1 - \gamma)D_i$.

If such a solution is not found, the algorithm returns a certificate that the instance is not feasible with demands $D_i$ and budget $B_i$. 

76
4.3 Vector Packing with Resource Augmentation

In this section we consider packing when resource augmentation is allowed in \((d - 1)\)-dimensions. We call these dimensions to be *augmentable* dimensions. The only other dimension where we are not allowed to augment, is called *non-augmentable* dimension. Without loss of generality, we assume the last dimension to be the non-augmentable dimension. It was implicit in [28] that if we allow resource augmentation in all \(d\) dimensions, then we can pack all items into \(\text{Opt}\) number of bins of length \((1 + \epsilon)\). Here we will restrict resource augmentation in the last dimension, which will be an useful tool for improved approximation for vector packing into unit bins. Now let us prove Theorem 4.0.3, the main result in this section.

**Theorem 4.3.1.** (Restatement of Theorem 4.0.3) Let \(\epsilon > 0\) be a constant and \(I\) be an instance of \(d\)-dimensional vector packing for which \(m\) unit vector bins are required in the optimal packing, then there is a polynomial time algorithm for packing vectors in \(I\) into at most \((1 + \epsilon + \frac{\epsilon^2}{2d})m + 1\) bins with \(\epsilon\) resource augmentation in \((d - 1)\) dimensions (i.e., bins have length \((1 + \epsilon)\) in \((d - 1)\) dimensions and length 1 in the other dimension).

Given \(\epsilon\) and a guess for the optimal value \(m\), we describe a procedure that either returns a feasible packing into \((1 + \epsilon + \frac{\epsilon^2}{2d})m + 1\) bins with \(\epsilon\) resource augmentation in \((d - 1)\) dimensions or proves the guess is incorrect. First we classify the items into *big* and *small* based on \(\beta = \epsilon/2d\). Then we round the big items into a constant number of classes. We set aside few vectors and pack them separately. Thereafter, for each bin configuration \(C\), the algorithm first decide if remaining rounded vectors in \(L\) can be packed viable to \(C\). This is done using dynamic programming and contributes a major slice of the overall time complexity. Afterwards we replace the rounded items by the original ones. In the final step we pack the small items using linear programming into the residual space of the bins and in some additional bins if
needed.

4.3.1 Rounding of big items

Now let us describe the procedure to round the big items into constant number of item types. We apply different rounding on augmentable and non-augmentable coordinates. Coordinates 1, . . . , d − 1 (those with the resource augmentation allowed) are rounded to the multiples of $\alpha$, where $\alpha = \frac{\epsilon^2}{2d}$ and the $d$-th coordinate is rounded using linear grouping. Note that the small items are not rounded and will be considered separately later.

4.3.1.1 Rounding of augmentable coordinates by preprocessing:

We create an instance $\hat{Q}$ rounded in the first $d − 1$ coordinates by replacing each big item $p_i$ of $I$ with an item $\hat{q}_i$ as follows:

\[
\hat{q}_i^\ell := \begin{cases} 
\left\lceil \frac{p_i^\ell}{\alpha} \right\rceil \alpha & \text{if } \ell \in \{1, \ldots, (d − 1)\} \\
p_i^\ell & \text{if } \ell = d
\end{cases}
\]

We split $\hat{Q}$ into classes $\{W_u | u \in \{1, \ldots, \lceil \frac{1}{\alpha} \rceil \}^{d−1}\}$ where $W_u := \{\hat{q} | \hat{q}^\ell = u^\ell \cdot \alpha\}$ for each $\ell \in [d − 1]$, creating $r_A := (\lceil \frac{1}{\alpha} \rceil)^{d−1}$ classes altogether.

4.3.1.2 Rounding of the non-augmentable coordinate:

We apply rounding of the last coordinate using linear grouping on each $W_u$ separately. Let $\lambda$ be a very small constant dependent just on $\epsilon$ and $d$ which will be fixed in the proof of Lemma 4.3.2. We sort the items in $W_u$ according to nonincreasing size in the last coordinate. Let $k_u$ be the number of vectors in $W_u$, denoted by $\hat{q}_{u,1}, \ldots, \hat{q}_{u,k_u}$ in sorted order according to nonincreasing size in the last coordinate. We define at most $\lceil 1/\lambda \rceil$ subsets $W_{u,j}$ each of which consists of $\lceil \lambda|W_u| \rceil$ vectors except the last subset that might contain less than $\lceil \lambda|W_u| \rceil$ vectors. This is done in the following way. We select the first vector $\hat{q}_{u,1}$ (call it the first round vector), then
assign that vector and next $\lceil \lambda |W^u| \rceil - 1$ vectors into one subset $W^{u,1}$. Then again we select the next vector (second round vector) and assign it and next $\lceil \lambda |W^u| \rceil - 1$ vectors into the set $W^{u,2}$ and so on. Thus, the $j$th subset $W^{u,j}$ contains $\lceil \lambda |W^u| \rceil - 1$ vectors $\{q_a,(j-1)\lceil \lambda |W^u| \rceil + 1), \ldots, q_a,(j\lceil \lambda |W^u| \rceil + j)\}$ and the $j$th round vector is $\tilde{q}_a,(j-1)\lceil \lambda |W^u| \rceil + j)$. The last subset can possibly contain less than $\lceil \lambda |W^u| \rceil$ vectors.

To get the final rounded instance $\tilde{Q}$ we replace each vector $\tilde{q}_i \in W^{u,j}$ by $\tilde{q}_i$, where

$$\tilde{q}_i^d := \tilde{q}_i^d \quad \text{for } \ell \in [d-1]$$

$$\tilde{q}_i^d := \max\{q^d \mid q \in W^{u,j}\}$$

Thus we round-up $d$th dimension to the $d$th coordinate of the round vector of the group. Note that the number of groups in each $W^u$ is $\leq \lceil (|W^u|) \rceil \leq \lceil \frac{1}{\lambda} \rceil$. Thus the rounded vectors in $\tilde{Q}$ can be classified into one of $r_L := \lceil 1/\lambda \rceil \cdot r_A = \lceil \frac{1}{\lambda} \rceil \cdot \lceil \frac{1}{\alpha} \rceil ^{(d-1)}$ distinct classes. Any configuration of vectors into one bin can be described as tuple $(k_1, k_2, \ldots, k_{r_L})$ where $k_i$ indicates the number of vectors of the $i$th class. As at most $d/\beta$ vectors from $\tilde{Q}$ can be there in any bin, there are at most $(d/\beta)^{r_L}$ configurations.

Now we prove that the described rounding procedure fulfils the requirements of the following lemma:

**Lemma 4.3.2.** Let $I$ be an instance of $d$-dimensional vector packing and $I_1, \ldots, I_m$ be a packing of $I$ into $m$ unit bins. Then there exists a packing of $\tilde{Q}$, rounding of the big vectors in $I$ using the described rounding procedure, such that:

1. Except for $\frac{\epsilon m}{2}$ items, all other big items in $\tilde{Q}$ can be packed into $m$ bins $\tilde{Q}_1, \ldots, \tilde{Q}_m$ of type $(1+\epsilon, 1+\epsilon, \ldots, 1+\epsilon, 1)$.

2. For $i \in [m]$, the configuration of small items in $\tilde{Q}_i$ is the same or larger than the configuration of small items in $I_i$ in all coordinates.

**Proof.** Given a packing of $I$ into $m$ bins, we can construct a packing of $\tilde{Q}$ in the following way. We set $\lambda := \frac{\epsilon \beta}{2m}$. We keep the round vectors in their same position.
For all $u \in \lfloor \frac{1}{\alpha} \rfloor^{d-1}$ and $i \in \lfloor \frac{1}{\lambda} \rfloor - 1$, we remove the other vectors (i.e., except the $i$th round vector) from the packing of $I$ corresponding to items in $W^{u,i}$ and put rounded items (except the $(i + 1)$th round vector) from $W^{u,i+1}$ in their places to construct packing $\tilde{Q}$. This will give a feasible packing as items in $W^{u,i+1}$ are same or smaller than the items in $W^{u,i}$ for each $u \in \{1, \ldots, \lfloor \frac{1}{\alpha} \rfloor\}^{d-1}$ and $i \in \lfloor \frac{1}{\lambda} \rfloor - 1$ and all subsets of $W^u$ have same cardinality of $\lfloor \lambda |W^u| \rfloor$ (except probably the last). Let $\tilde{Q}'' := \bigcup_{u \in \{1, \ldots, \lfloor \frac{1}{\alpha} \rfloor\}^{d-1}} (W^{u,1} \setminus \{\tilde{q}_{u,1}\})$ and $\tilde{Q}' = \tilde{Q} \setminus \tilde{Q}''$. By now, all items in $\tilde{Q}'$ are packed in a feasible way and we are left with items in $\tilde{Q}''$ (i.e., the set of all items in $W^{u,1}$ for each $u \in \{1, \ldots, \lfloor \frac{1}{\alpha} \rfloor\}^{d-1}$ except the round vectors) which we will pack separately. It is clear that in any bin, we did not use more space in the last dimension to pack all items in $\tilde{Q}'$, than what the big items used in the original packing in the last dimension. Let $\tilde{Q}_i$ be a bin in the packing of $\tilde{Q}'$, that we obtained from a packing of $I_i$ using the above procedure. Let us consider the sum of sizes in any coordinate $\ell \in [d-1]$

\[
\sum_{\tilde{q} \in \tilde{Q}_i} \tilde{q}^\ell \leq \sum_{p \in I_i} \left\lfloor \frac{p^\ell}{\alpha} \right\rfloor \cdot \alpha \\
\leq \sum_{p \in I_i} (p^\ell + \alpha) \\
= \sum_{p \in I_i} p^\ell + \frac{d}{\beta} \alpha \quad \left[\because |I_i| \leq \frac{d}{\beta}\right] \\
= \sum_{p \in I_i} p^\ell + \epsilon \quad \left[\because \alpha = \frac{\epsilon^2}{2d^2} = \frac{\beta \epsilon}{d}\right]
\]

Hence,

\[
(1 + \epsilon) - \sum_{\tilde{q} \in \tilde{Q}_i} \tilde{q}^\ell \leq 1 - \sum_{p \in I_i} p^\ell
\]

Therefore, the augmentation of the bin in coordinate $\ell \in [d-1]$ by $\epsilon$ is sufficient and for $i \in [m]$, the configuration of small items (remaining space in the bin) in $\tilde{Q}_i$ is the same or larger than the configuration of small items in $I_i$ in all coordinates.

Now to conclude the proof we need to show, $|\tilde{Q}''| \leq \frac{cm}{2}$. Each big item has size at least $\beta$ in at least one of the coordinates. In each bin there can be at most $1/\beta$ items
for each coordinate which are big when restricted to this coordinate, having at most 
\( \frac{d}{\beta} \) big items in a bin in total. Therefore we have \( \frac{d}{\beta} m \geq \sum |W^u| \). Now,

\[
|\tilde{Q}'| = \sum_u (|W^{u,1}| - 1) \leq \sum_u ([|\lambda|W^u|] - 1) \leq \sum_u \lambda|W^u| \leq \frac{d}{\beta} m \leq \frac{\epsilon \beta}{2d \cdot \beta} m \leq \frac{\epsilon m}{2} \leq \left\lfloor \frac{\epsilon m}{2} \right\rfloor
\]

So, we are left with at most \( \left\lfloor \frac{\epsilon m}{2} \right\rfloor \) many items. Note that we can always pack these extra items greedily into \( \left\lfloor \frac{\epsilon m}{2} \right\rfloor \) additional bins. □

### 4.3.2 Packing of big items

From Lemma 4.3.2, it follows that if \( I \) is packable into \( m \) bins then there is a structured rounding \( \tilde{Q} \) of big items in \( I \). There is a constant number of item types \( r_L \) in \( \tilde{Q} \). Thus there are only \( r \leq (\frac{d}{\beta})^{r_L} \), number of configurations. Since \( m \leq n \), this gives us \( O(m^r) \), a polynomial number of configurations of big items for \( m \) bins and allow us to guess it (more precisely, to try each configuration until we find the right one).

**Lemma 4.3.3.** Let \( m := Opt(I) \), then the rounded instance \( \tilde{Q} \) can be packed in polynomial time into \( m' := m + \left\lfloor \frac{\epsilon m}{2} \right\rfloor \) unit bins.

**Proof.** We guess the configuration of the \( m' \) bins which would correspond to the configuration of big items in some packing of \( I \) into \( m \) bins in the sense of Lemma 4.3.2. We guess the round vectors and pack them separately into \( \left\lfloor \frac{\epsilon m}{2} \right\rfloor \) bins. If there is a feasible packing of \( \tilde{Q}' \) corresponding to the guessed configuration, we can find it using dynamic programming in time \( O((\frac{dn}{\beta})^{r_L} \cdot m) \) according to the following lemma in [28, Lemma 2.3].
Lemma 4.3.4. [28]. Let $M = (m_1, \ldots, m_r)$ be a bin configuration. There exists an algorithm with running time $O((\frac{dn}{\beta})^r \cdot m)$ to decide if there is a packing of the jobs in $\tilde{Q}'$ that is viable to $M$.

If we can’t find one, we have guessed a wrong configuration and we should try again. If we tried all the configurations and none of them worked, this is a contradiction to the Lemma 4.3.2 and we know that $I$ is not packable into $m$ bins. \hfill \square

4.3.3 Packing of small items

Lemma 4.3.5. Let $(\tilde{Q}_1, \ldots, \tilde{Q}_m)$ be a packing of large vectors in $I$ into $m$ bins of type $(1 + \epsilon, \ldots, 1 + \epsilon, 1)$ such that there is an optimal packing $(I_1, \ldots, I_m)$ into $m$ unit vector bins, so that for each $k \in [m]$ configuration of small items in $\tilde{Q}_i$ is the same or larger than the configuration of small items in $I_i$ in all coordinates. Then we can pack all the small items of $I$ into residual space of bins $\tilde{Q}_1, \ldots, \tilde{Q}_m$ and at most $\left\lceil \frac{m}{2} \cdot (\epsilon + \frac{\epsilon^2}{d}) \right\rceil$ additional bins.

Proof. To pack the small vectors, we use the idea of Chekuri and Khanna [28]. However in [28], as resource augmentation was allowed in all dimensions, small vectors could have been packed without using any additional bins. In our case we need extra bins. We formulate the following assignment LP:

\[
\begin{align*}
\sum_{j=1}^{m} x_{ij} &= 1 \quad \text{for each small item } v_i \in \mathcal{S} \\
\sum_{i=1}^{\mathcal{S}} x_{ij}v_i^\ell &\leq (1 + \epsilon) - b_j^\ell \quad \text{for each bin } j \in [m] \text{ and dimension } \ell \in [d - 1] \\
\sum_{i=1}^{\mathcal{S}} x_{ij}v_i^\ell &\leq 1 - b_j^\ell \quad \text{for each bin } j \in [m] \text{ and dimension } \ell = d \\
x_{ij} &\geq 0 \quad \text{for each item } i \in \mathcal{S} \text{ and each bin } j \in [m]
\end{align*}
\]

where $b_j := \sum_{\tilde{q} \in \tilde{Q}_i} \tilde{q}$. Since there is an optimum packing of $I$ that for each bin $j$ use a smaller or equal space for small items, the assignment LP is feasible, let $x$ be a basic
feasible solution to this LP. The integrally assigned items can be packed directly. We denote $F$ to be the set of items assigned fractionally in $x$. The number of variables in the LP is $m \cdot |\mathcal{S}|$ and thus from standard polyhedral theory any basic solution to the LP has $m \cdot |\mathcal{S}|$ tight constraints. However there are $dm + |\mathcal{S}|$ non-trivial constraints. Thus at most $dm + |\mathcal{S}|$ variables can be strictly positive. As each vector is assigned to at least one bin, the number of vectors that are fractionally assigned to more than one bin is at most $dm$. Hence, $|F| \leq dm$. We pack the items in $F$ in separate bins.

Now, from any constant $\psi < 1/2$, we have the following inequality:

\[
\frac{1}{\psi} - \left(\frac{1}{\psi}\right) \cdot (1 + 2\psi) \leq \frac{1}{\psi} - (\frac{1}{\psi} - 1) \cdot (1 + 2\psi) \\
\leq 2\psi - 1 < 0
\]

Hence,

\[
\frac{1}{\left(\left\lfloor \frac{1}{\psi} \right\rfloor\right)} < \frac{(1 + 2\psi)}{(\frac{1}{\psi})}
\]

Since the size of each item is less than $\beta$ in both dimensions, the number of additional bins are at most

\[
\left\lceil \frac{dm}{\frac{1}{\beta}} \right\rceil \leq \left\lceil \frac{dm}{\frac{2d}{\epsilon}} \right\rceil \\
\leq \left\lceil dm \cdot \frac{(1 + \epsilon/d)}{(\frac{2d}{\epsilon})} \right\rceil \\
\leq \left\lceil \frac{m}{2} \cdot (\epsilon + \frac{\epsilon^2}{d}) \right\rceil
\]

[From (21), using $\psi = \epsilon/2d$]

\[
\square
\]
4.3.4 Algorithm and proof of Theorem 4.0.3

**Guessing phase:**

Guess $m := \text{Opt}(I)$  \hspace{1cm} // $O(\log n)$ guesses

Create $\tilde{Q}$, the rounded instance of big vectors of $I$ \hspace{1cm} // $O(n \log n)$

Guess big configuration $M$ of $m + \lceil \frac{\epsilon m}{2} \rceil$ bins of type $(1 + \epsilon, \ldots, 1 + \epsilon, 1)$ \hspace{1cm} // $O(m^t)$,

**Packing phase:**

Pack items in $\tilde{Q}$ into configuration $M$, or \hspace{1cm} // $O((\frac{dn}{\beta})^{rL} \cdot m)$

return to Guessing phase

Replace rounded items of $\tilde{Q}$ by original items \hspace{1cm} // $O(n)$

Pack small items into the free space of the packing and additional

$\left\lceil \frac{m}{2} \cdot (\epsilon + \frac{\epsilon^2}{d}) \right\rceil$ bins using the assignment LP, or

return to Guessing phase

**Algorithm 1:** Algorithm for $d$-dimensional vector packing with resource augmentation in $(d - 1)$ dimensions

of Theorem 4.0.3. If there is a packing of $I$ into $m$ unit bins, from Lemma 4.3.2 we know that there is a packing of rounded instance $\tilde{Q}$ of big vectors of $I$ into $m' := m + \lceil \frac{\epsilon m}{2} \rceil$ unit bins. Moreover, Lemma 4.3.2 assures, that there is still enough space to pack all the small vectors except at most $dm$ small items. Thus if we can guess the bin configuration of the optimal packing, We can pack the small items of $I$ using Lemma 4.3.5, getting packing of $I$ into $m'' := m + \lceil \frac{\epsilon m}{2} \rceil + \lceil \frac{m}{2} \cdot (\epsilon + \epsilon^2/d) \rceil \leq m + m \cdot (\epsilon + \frac{\epsilon^2}{2d}) + 1$ bins.

Since items in $Q$ are rounded to a constant number of classes, there is a polynomial number of configurations of $m$ bins for $Q$ and we can, in the worst case, enumerate them all. If we fail in the following steps in packing $I$ into $m'' := m + m \cdot (\epsilon + \frac{\epsilon^2}{2d}) + 1$ bins for every choice of bin configurations, due to Lemma 4.3.2, and 4.3.5, we can claim that $\text{Opt}(I) > m$. Otherwise $\text{Opt}(I) \leq m''$. 

84
This enables us to guess the value of \( m \) using binary search between \( \| \sum_{v \in I} v \|_\infty \) and \( n \). We find the smallest \( m \) such that we are able to pack \( I \) into \( m'' \) bins using Algorithm 1. Then from Lemma 4.3.2 and 4.3.5 we know that \( m \leq \text{Opt}(I) \leq m'' = m + m \cdot (\epsilon + \frac{\epsilon^2}{2d}) + 1 \) and the statement of the theorem follows.

### 4.4 Vector Packing without resource augmentation

In this section we prove following theorem:

**Theorem 4.4.1.** [\( d \)-Dimensional Vector Packing] Let \( \epsilon \) be a small positive constant and \( I \) be an instance of \( d \)-dimensional vector packing for which \( m := \text{Opt}(I) \) unit vector bins are required in the optimal packing, then there is a polynomial time algorithm for packing vectors in \( I \) into at most \( \left\lceil \left( \frac{d+1}{2} \right) \cdot (1 + \epsilon + \frac{\epsilon^2}{2d}) + 1 \right\rceil \) \( m' \) unit bins where \( \xi' = \left( \frac{d+1}{2} \right) \cdot (1 + \epsilon + \frac{\epsilon^2}{2d}) \) and \( c_2 = 2 + \epsilon + \frac{\epsilon^2}{2d} \).

Thus it gives \((1.5 + \xi')\) and \((2 + \xi')\) asymptotic approximation for for 2-D and 3-D vector packing and improves upon the present best asymptotic approximation ratios of \((1.693 + \xi')\) and \((2.099 + \xi')\) respectively.

For an instance \( I \), if there exists a packing into small number of bins having a \( \delta \) slack in all dimensions, we could use the the algorithm of Chekuri and Khanna [28] with resource augmentation in all dimensions to get an almost optimal packing. However, \( \text{Opt}_{(1-\delta)} \), the minimum number of bins of size \((1-\delta)^d\) to pack all items in \( I \), can be quite large compared to \( m \). Consider 2-D instance of \( n \) identical vectors of size \((\frac{1}{2}, \frac{1}{2})\). It is easy to see, that \( \text{Opt} = n/2 \), while \( \text{Opt}_{(1-\delta)} = n \). Similarly an example can be constructed, showing that in \( d \)-dimensional case \( \text{Opt}_{(1-\delta)} \) can be \( d \)-times larger than \( \text{Opt} \). Consider the set of \( md \) vectors that can be packed into \( m \) bins in optimal packing such that in optimal packing in each bin \( B \), there are \( d \) vectors \( v_j \) for \( j \in [d] \) having coordinate \( j \) equal to \( 1 - \delta \) and the rest equal to \( \frac{\delta}{d-1} \). Then \( \text{Opt} = m \), while \( \text{Opt}_{(1-\delta)} = md \).

We show, that allowing packings of more sophisticated structure, we can achieve
better results. First, we define what we call \textit{structured packing}. We call a packing to be \textit{structured} if all the bins fit into one of the following categories:

- $\mathcal{B}_S$: bins containing only a single big item $v$,
- $\mathcal{B}_T$: bins containing two big items $v_i, v_j$,
- $\mathcal{B}_\ell$ for $\ell \in [d]$: bins containing slack $\delta$ in all dimensions except for $\ell$, meaning that $\ell$ is the only dimension where the bin can contain smaller slack than $\delta$.

If a bin has a slack at least $\delta$ in all dimensions, we assign it to any of the classes $\mathcal{B}_1, \ldots, \mathcal{B}_d$ arbitrarily. Let us fix $\delta := \frac{\epsilon}{1+\epsilon}$. We claim the following:

\textbf{Lemma 4.4.2.} Let $\epsilon > 0$ be a constant and $I$ be an instance of $d$-dimensional vector packing having a structured packing into $m'$ bins, then there is a polynomial time algorithm that pack items in $I$ into at most $(1 + \epsilon + \frac{\epsilon^2}{2d})m' + 1$ bins.

Algorithm proving Lemma 4.4.2 is given at the end of this section. Together with Lemma 4.4.7 (proving existence of relatively small structured packings) it already implies Theorem 4.4.1.

\subsection*{4.4.1 Small structured packings:}

In the following part of this section we prove that there indeed exist small structured packing.

\textbf{Lemma 4.4.3.} Let $\mathcal{B}$ be a bin of 2-D vector packing that does not correspond to any of the structured packing types. Then at least one of the following holds:

1. There is a big item $p \in \mathcal{B}$ such that $p \leq \frac{1}{2}$ and $p$ is $\geq \delta$ in at least one coordinate.

2. There is a subset $S$ of items in $\mathcal{B}$ such that $\sum_{p \in S} p \leq 2\delta$ in both dimensions and
• either $B \setminus S$ contains slack in at least one dimension

• or $|B \setminus S| \leq 2$ with $\sum_{v \in B \setminus S} v \geq 1 - \delta$.

Proof. Let $\mathcal{S}$ denote the set of items $< \delta$ in $B$. Note that in any bin there can be at most two items bigger than $1/2$ in some dimension (for each dimension there can be at most one). Therefore if there is no $p$ satisfying requirements of (1), we know that $|B \setminus \mathcal{S}| \leq 2$.

If $\sum_{p \in B \setminus \mathcal{S}} p > 1 - \delta$ in both dimensions, we set $S := \mathcal{S}$. Then $\sigma_S \leq \delta$ and $|B \setminus \mathcal{S}| \leq 2$ with $\sum_{v \in B \setminus S} v \geq 1 - \delta$.

Otherwise, $\sigma_\mathcal{S} > \delta$ in at least one of the dimensions. So we greedily add items of $\mathcal{S}$ to $S$ until $B \setminus S$ contains $\delta$ slack in at least one dimension. Since $\mathcal{S}$ contains items ($< \delta$) only, the items of $S$ cannot sum to more than $\delta + \delta \leq 2\delta$ in any coordinate. \qed

If we consider $\epsilon < 1/4$, then $\delta < 1/5$. Thus $4\delta < (1 - \delta)$ and $1/2 + \delta < (1 - \delta)$. Thus from each bin we remove either $p$ or $S$ and for any two bins, the removed elements can be packed into bins that are $\leq (1 - \delta)$-packed or contains two elements. Thus 2-D case of Lemma 4.4.7 already follows fairly easily as we can remove corresponding $p$ or $S$ from $m$ bins and pack them in arbitrary pairs into $\lceil m/2 \rceil$ additional bins of type $B_S$, $B_T$ or $B_\ell$ for $\ell \in [d]$. After removal of $p$ or $S$, remaining bins are are of type $B_S$, $B_T$ or $B_\ell$ for $\ell \in [d]$.

Lemma 4.4.4. Let $I$ be an instance of 2-dimensional vector packing. Any optimal packing of $I$ into $m$ bins can be transformed into a packing into $\lceil \frac{3}{2}m \rceil$ bins of types $B_S, B_T, B_1, B_2$.

Before proving the $d$-dimensional structural lemma, we make the following observation.

Lemma 4.4.5. Let $B$ be a set of $d$-dimensional vectors that can be feasibly packed into one unit vector bin. Then there is a subset $V$ of big items and another subset $S$
of items of $B$ such that the bin packed with all the items in $B := B \setminus (V \cup S)$ contains slack of $\geq \delta$ in at least $d - 1$ dimensions and one of the following holds.

1. $|V| \leq (d - 1)$, $S = \emptyset$,

2. $|V| \leq (d - 2)$, $S \neq \emptyset$, such that $\sum_{p \in S} p \leq \kappa \delta$ in all coordinates for a small constant $\kappa$.

Furthermore, removal of each item in $V$ from $B$, creates slack of $\geq \delta$ in at least one distinct coordinate.

Proof. We start with $C = [d], V = \emptyset$ and then we update $C$ after removal of items in the following steps by considering the dimensions one by one. In each iteration we select a coordinate $k \in C$ that we have not considered already. Let $v$ be the vector with highest coordinate along $k$, i.e., $v_k \geq p_k$ for all $p \in B$. If removal of $v$ from $B$ creates $\delta$ slack along dimension $k$, then remove $v$ from $B$ and update $B = B \setminus \{v\}$, $C = C \setminus \{k\}, V = V \cup \{v\}$. We continue this while $|C| > 1$ and while such a removable item $v$ exists. Thus by definition of $V$, removal of each item in $V$ from $B$, creates slack in at least one distinct coordinate in $B$ and $|V| \leq d - 1$.

As element in $V$ creates slack in at least one distinct coordinate, if $|V| = d - 1$, $B \setminus V$ already has slack in $(d - 1)$ dimensions and we are done by taking $S = \emptyset$.

Otherwise let $|C| \geq 2$, where $C$ denotes the set of coordinates where $B$ does not contain $\delta$ slack. So, $|V| \leq (d - 2)$ Hence after removing the items in $V$, all items in $(B \setminus V)$ are $< \delta$ in all coordinates of $C$. Note that they can still be relatively big in coordinates of $\overline{C} := [d] \setminus C$. Now we prove the following general claim.

Claim 4.4.6. Let $B'$ be the set of $d$-dimensional vectors that can be feasibly packed into a unit vector bin and $C$ be the set of coordinates where the sum of vectors in the bin is $\geq (\theta - \delta)$ for some constant $\theta \leq 1$. If all items in a set $B'$ are smaller than $\delta$ in all coordinates of $C$ and $\delta < \frac{\theta}{6}$, then there exists a subset $S$ such that $|S| \geq 2$ and
\[ \sum_{p \in S} p \leq \kappa \delta \text{ in all coordinates for some constant } \kappa \text{ and } \sum_{p \in S} p \geq \delta \text{ in all coordinates in } C. \]

**Proof of Claim:** Let \( B'' \) be set of all items of \( B' \) which are \( < \frac{3\delta d}{\theta} \) in all coordinates of \( \overline{C} \). We formulate the following LP:

\[
\begin{align*}
\min & \sum_{p_i \in B''} x_i \\
\text{s.t.} & \delta \leq \sum_{p_i \in B''} x_i p_i^{\ell} \leq \frac{2\delta}{\theta} \quad \forall \ell \in C \\
& \sum_{p_i \in B''} x_i p_i^{\ell} \leq \frac{2\delta}{\theta} \quad \forall \ell \in \overline{C} \\
& 0 \leq x_i \leq 1 \quad \forall i \in B'', \ell \in [d]
\end{align*}
\]

To show the feasibility of this LP, let us consider \( x' = \left( \frac{2\delta}{\theta}, \ldots, \frac{2\delta}{\theta} \right) \). Let \( \hat{B} = B' \setminus B'' \). There could be at most \( \frac{d\theta}{3\delta d} \) items in \( B' \) bigger than \( \frac{3\delta d}{\theta} \) in some coordinate in \( C \). As these items are \( \leq \delta \) in all coordinates in \( C \), \( \sum_{p_i \in B} p_i \leq \delta \cdot \frac{d\theta}{3\delta d} \leq \theta/3 \) in all coordinates in \( C \). Now \( \sum_{p_i \in B'' \cup \hat{B}} p_i \geq (\theta - \delta) \) in all coordinates in \( C \) by taking \( \delta < \theta/6 \). Therefore, we have \( \sum_{p_i \in B''} x_i p_i \geq \frac{2\delta}{\theta} \cdot \frac{\theta}{2} \geq \delta \) in coordinates of \( C \). Since the items of \( B'' \) were packed into a single bin, we know that \( \sum_{p_i \in B''} p_i \leq 1 \) and therefore \( \sum_{p_i \in B''} x_i p_i \leq \frac{2\delta}{\theta} \) in all coordinates and thereby \( x' \) is a feasible solution of (22).

Let us fix a basic optimal solution \( x \) to (22) and set \( S = \{ p_i \mid x_i > 0 \} \). We claim that the sum \( \sum_{p_i \in S} p_i \) is at most \( \kappa \delta \) in each dimension for some constant \( \kappa \) dependent just on \( \delta \) and \( d \). Let \( F \) denote the set of fractional variables \( x_i \). We know that each basic solution to the LP (22) fulfills at least \( |B''| \) constraints with equality. Apart from constraints \( 0 \leq x_i^{\ell} \leq 1 \), which would make the variable integral if fulfilled with equality, there are at most \( (2|C| + |\overline{C}|) \leq 2d \) nontrivial constraints and they allow us \( |F| \leq 2d \). Since each item is at most \( 3\delta d/\theta \) in all dimensions we get \( \sum_{p_i \in S} p_i \leq 2\delta + 2d \cdot \frac{3\delta d}{\theta} \leq 7\delta d^2/\theta \) as \( d \geq 2 \). Hence, the claim follows by taking \( \kappa = 7d^2/\theta \).
This completes the proof.

If we consider \( \epsilon < \frac{\theta}{28d^2} \), then \( \delta < \frac{\theta}{28d^2} \). Thus \( 2 \cdot \kappa \delta < (1 - \delta) \) and \( 1/2 + \kappa \delta < (1 - \delta) \). Thus from each bin we remove \( S \) and for any two bins, the removed elements can be packed into bins that have slack \( \geq \delta \) in at least \( d - 1 \) coordinates or contains at most two items.

**Lemma 4.4.7.** Let \( I \) be an instance of \( d(> 2) \)-dimensional vector packing. Any optimal packing of \( I \) into \( m \) bins can be transformed into a packing into \( \lceil \frac{d+1}{2}m \rceil \) bins of types \( \mathcal{B}_S, \mathcal{B}_T, \mathcal{B}_1, \ldots, \mathcal{B}_d \).

**Proof.** For each bin \( B_i \) in optimal packing, let \( V_i \) and \( S_i \) denote the sets guaranteed by Lemma 4.4.5. So, the bin with items in \( B_i \setminus (V_i \cup S_i) \) is already of type \( \mathcal{B}_\ell \) for some \( \ell \in d \). We have to pack items in \( V_i \cup S_i \) into bins of types \( \mathcal{B}_S, \mathcal{B}_T, \mathcal{B}_1, \ldots, \mathcal{B}_d \).

Now there are two cases.

**Case A.** If \( d \) is odd, i.e., \( d - 1 \) is even. If \( |V_i| \leq d - 3 \) we can pack all items in \( |V_i| \) in pairs into at most \( \frac{d-3}{2} \) bins. Otherwise if \( |V_i| \geq d - 2 \), there is an item \( v \in V_i \) such that \( v \) is \( < 1/2 \) in all coordinates except one (say \( d_y \)). Pack \( v \) with \( S_i \) (if \( S_i \neq \emptyset \)) or an arbitrary other element from \( V_i \) (if \( S = \emptyset \)) separately in another bin \( \mathcal{B} \). Note that \( \mathcal{B} \) either has at most two items (if \( S_i = \emptyset \)) or has slack in all dimensions except \( d_y \), satisfying condition of \( \mathcal{B}_\ell \) bins. We can pack all remaining items in \( |V_i| \) in pairs into at most \( \frac{d-3}{2} \) bins. Total bins needed \( \leq (\frac{d-3}{2} + 1) + \frac{m}{2} \leq \lceil \frac{d+1}{2}m \rceil \).

**Case B.** If \( d \) is even, i.e., \( d - 1 \) is odd.

If \( |V_i| \leq (d-2) \) and \( S_i \neq \emptyset \), then for any two bins \( B_i \) and \( B_j \) we can pack corresponding sets \( S_i \) and \( S_j \) together and we either get a bin with at least \( \delta \) slack in all dimensions. We pack items in \( V_i \) into pairs. Total bins needed \( \leq \frac{d-2}{2} \cdot m + m + \lceil \frac{m}{2} \rceil \leq \lceil \frac{d+1}{2}m \rceil \) bins.
Only remaining case is when \(|V_i| = (d - 1)| S_i = \emptyset\). Now if there is a set \(R_i \subseteq B_i\) such that either \(R_i = \{p_i\}\) with \(p_i \leq 1/2\) in all coordinates or \(R_i\) is a set \(S_i'\) with \(\sigma_{S_i'} \leq \frac{1}{2} - \delta\) in all dimensions except one, then all sets of type \(R_i\) from different bins can be paired together into bins of type \(B_T\) or \(B_\ell\). So, all \(R_i\) items from \(m\) bins can be packed into at most \(\leq \lceil \frac{m}{2} \rceil\) bins of type \(B_S, B_T, B_\ell\). For each bin we need to show existence of such \(R_i\) and the remaining items are packable in \(d/2\) bins. Note that as each element in \(V_i\) creates slack in at least one distinct coordinate, \(B_i \setminus V_i\) has slack in all coordinates except at most one, thus already in one of the structured types. Now there are two cases.

Case B1. There is an element \(p_i \in V_i\) with \(p_i \leq 1/2\). We can set \(R_i = \{p_i\}\), pack \((d - 2)\) other elements in \(V_i \setminus \{p_i\}\) in pairs, \(B_i \setminus V_i\) into a bin of type \(B_\ell\).

Case B2. There is no element \(p_i \in V_i\) with \(p_i \leq 1/2\). Thus all elements in \(V_i\) are \(> 1/2\) in at least one distinct coordinate. Then there exists a vector \(v_x\) that is \(> 1/2\) in only one coordinate and \(\leq 1/2\) in all other coordinates as \(d > 2\). W.l.o.g. assume \(V_i = \{v_{V,1}, \ldots, v_{V,d-1}\}\) and \(v_{V,i} > 1/2\) in dimension \(i\). Now consider the vector \(v_{V,d}\) in \(B_i \setminus V_i\) with highest value on \(d\)-dimension. There are three cases:

Case B1a. \(v_{V,d} > 1/2\).

In this case the sum of all vectors in \(B_i \setminus (V_i \cup \{v_{V,d}\})\) is \(< 1/2\) in all dimensions. Let \(Q_i := \{q_1, \ldots, q_{|Q_i|}\}\) be the set of dimensions where \(\sigma_{B_i \setminus (V_i \cup \{v_{V,d}\})}\) is \(> \frac{1}{2} - \delta\). If \(Q_i \leq 1\), take \(S_i' = B_i \setminus (V_i \cup \{v_{V,d}\})\) and pack remaining \(d\) vectors in pairs in \(d/2\) bins. Otherwise if \(Q_i \geq 2\), pack \(P_i := \{v_{V,q_1}, \ldots, v_{V,q_{|Q_i|}}, v_{V,l_1}\}\) in one bin where \(l_1 \notin Q_i\). For dimensions in \(Q_i\), \(P_i\) has slack \(\geq \frac{1}{2} - \delta\) and in all other dimensions except possibly \(l_1\), \(P_i\) has slack \(> 1/2\). Thus this set has slack in all dimensions except possibly in dimension \(l_1\). Pack elements in \(B_i \setminus (V_i \cup \{v_{V,d}\})\) with another remaining vector \(v_{V,l_2}\). Clearly this set
has slack in all dimensions except dimension $l_2$. Remaining $(d - 1 - |Q| - 1)$ vectors are packed in pairs. Total number of bins needed $\leq \lceil \frac{(d-1-|Q|-1)}{2} \rceil + 1 + 1 \leq \lceil \frac{d}{2} \rceil$ as $|Q| \geq 2$ and $d \geq 4$ (\because d > 2 and d is even).

**Case B1b.** $1/2 \geq v_{d,d} \geq \delta$. Simply take $p_i = \{v_{V,d}\}$. Pack $v_x$ with $B_i \setminus (V_i \cup \{p_i\})$ in a bin that will have slack in all dimensions except possibly in dimension $x$. Remaining $(d - 2)$ vectors in $V_i$ can be packed in pairs.

**Case B1c.** $v_{d,d} < \delta$. If $B_i \setminus V_i \leq \frac{1}{2} - \delta$ in dimension $d$, pack $v_x$ with $B_i \setminus V_i$ in a bin that will have slack in all dimensions except possibly in dimension $x$. pack remaining vectors in pair. Last case is when $B_i \setminus V_i > \frac{1}{2} - \delta$ in dimension $d$. Let $H_i$ be the set of coordinates where $B_i \setminus V_i > \frac{1}{2} - \delta$. Apply the Claim in the proof of Lemma 4.4.5 restricted to coordinates in $H_i$ and take $C = \{d\}$, $\theta = \frac{1}{2}$ to get a set $S'_i$ that is $> \delta$ in $C$ and $< \kappa \delta$ in all coordinates in $H_i$. Thus, $v_x$ can be packed with $B_i \setminus (V_i \cup S'_i)$ that have slack in all dimensions except possibly dimension $x$. Remaining vectors can be packed in pairs. 

4.4.2 Algorithm for $d$-Dimensional Vector Packing:

From Lemma 4.4.7, we know that any optimal packing of $I$ can be transformed into a packing of $m' := \lceil \frac{d+1}{2} \cdot m \rceil$ bins of types $\mathcal{B}_S, \mathcal{B}_T, \mathcal{B}_1, \ldots, \mathcal{B}_d$. Let $m_S, m_T, m_\ell$ be the number of bins of types $\mathcal{B}_S, \mathcal{B}_T, \mathcal{B}_\ell$ respectively for $\ell \in [d]$. So, $\sum_\ell m_\ell + m_S + m_T \leq \lceil \frac{d+1}{2} \cdot m \rceil$. Assume we can partition instance $I$ into $d + 2$ sets $I_S, I_T, I_1, \ldots, I_d$ such that they correspond to elements of bins of type $\mathcal{B}_S, \mathcal{B}_T, \mathcal{B}_1, \ldots, \mathcal{B}_d$ respectively in some structured packing of at most $\lceil \frac{d+1}{2} \cdot m \rceil$ bins. Now let us consider a graph $G$ with vertices as items in $\mathcal{B}_T$ and there is an edge between two item $v_i, v_j$ if they can be packed together in a bin, i.e., $\sigma_{\{v_i, v_j\}} \leq 1$. Thus if we find the maximum matching in $G$, we get an optimal packing of items in $\mathcal{B}_T$. Then we can pack each item in
*I* separately in *m*\(_S\) bins, pack items in *I*\(_T\) into *m*\(_T\) bins using matching and pack items in *I*\(_\ell\) into \((1 + \epsilon + \frac{\delta^2}{2\delta})m_\ell + 1\) unit vector bins using Algorithm 1 with resource augmentation in all coordinates except \(\ell\), since we have \(\delta\) slack in all dimensions which could be overpacked by factor \((1 + \epsilon)\) and \((1 - \delta)(1 + \epsilon) = 1\).

However in general there are exponentially many such partition of *I*, thus we will try to get a good approximate partition. If we simply guess the number of items in bins of type \(B_S\) and \(B_T\) bins and rounding size classes used in Algorithm 1 for each \(\ell \in [d]\), then the above approach might not work. For example, if we separate *I*\(_S\) and subinstances *I*\(_\ell\) first, since Algorithm 1 recognizes the items just up to the rounding, it can happen that the \(2m_T\) items which are leftover for the matching type bins are not matchable together. On the other hand, if we try to pack the matching bins first, the standard matching can possibly select too small items which would result in moving a smaller item from the bins packed using Algorithm 1 and putting a larger one back. This way we loose guaranties for the performance of Algorithm 1, since we are in fact packing different items. Therefore, to get a better control over the items which will be matched, we guess for each rounding class how many items with similar or slightly smaller sizes will be placed in bins of type \(B_T\) or \(B_S\), and we find a matching respecting these requirements. Though matching will be found on the items with original sizes, this will ensure that that even though we might not pack the same items into the matching bins as the structured packing did, we packed items from the same rounding class and therefore we do not affect the performance of Algorithm 1. We will also show polynomially many guesses are sufficient in this approach.

To describe the algorithm, we will use similar notations and results as in previous Section 4.3. Remember, \(r_A\) is the number of rounding classes of augmentable coordinates and there are \(\left\lceil \frac{1}{\lambda} \right\rceil\) number of linear grouping classes inside of each of them.
4.4.2.1 Preprocessing phase:

First, we separate the small items $S$ of $I$, this can be done in linear time. We just put the small items aside for now, they will be packed later. Let us denote $I' := I \setminus S$.

4.4.2.2 Guessing phase:

Our algorithm is trying to find $m'$, the minimum number of bins in a structured packing for $I$. It is clear that $m'$ is at most $n$ and at least $\lceil \sigma_\infty I \rceil$. So we can guess $m'$ by binary search between $n$ and $\lceil \sigma_\infty I \rceil$ in the following way. If the Packing phase of the algorithm finds a structured packing into $m'$ bins, we decrease its value. Otherwise we increase it. After guessing of $m'$ we have to guess the partition of $m'$ bins to bins of types $B_S, B_T$ and $B_1, \ldots, B_d$. It is clear that there is a polynomial number of choices of $m'$ and $m_S, m_T, m_1, \ldots, m_d$ and in the worst case we can enumerate them all in $O((m')^{(d+2)})$. As $m' = O(md)$ from Lemma 4.4.7, the next lemma follows.

**Lemma 4.4.8.** Let $I$ be an instance having a structured packing into $m'$ bins. The guessing phase will guess $m_S, m_T, m_1, \ldots, m_d$ in $O((md)^{(d+2)})$ time.

For each $\ell \in [d]$, we define the rounding classes $W_{\ell}^u$ for $u \in [(\frac{1}{\lambda})^d \setminus \{\ell\}]$ according to the first part of the rounding procedure in Section 4.3. For each $\ell \in [d]$ we have $r_A$ (constant) number of classes, none of them clearly containing more than $n$ items, and inside of each class the items are to be rounded to size of one of $\lceil \frac{1}{\lambda} \rceil$ round vectors (again a constant), so there are at most $r_A \cdot n(\frac{1}{\lambda})$ possible choices of round vectors. This way we can classify $d \cdot r_L$ rounding classes $W_{\ell}^{u,j}$.

For each $W_{\ell}^u$, we guess numbers $w_{\ell}^{u,1} = \cdots = w_{\ell}^{u,\lceil \frac{1}{\lambda} \rceil - 1}$ — the sizes of the regular linear grouping classes and $w_{\ell}^{u,\lceil \frac{1}{\lambda} \rceil} \leq w_{\ell}^{u,1}$ the size of the last possibly smaller group. Note, that by now we have prepared rounding instances for the resource augmented packing as in Algorithm 1. We also guess numbers $s_{\ell}^{u,j}$ and $t_{\ell}^{u,j}$ — the numbers of items of size corresponding to the class $W_{\ell}^{u,j}$ but contained in the bins of type $B_S$ and $B_T$ respectively. Note that any item $v$ in $I_S$ or $I_T$ are actually not rounded.
They are just assigned to a rounded class with size $\geq v$ and they are packed using matching using their original sizes.

As we are guessing 4 numbers for each of $dr_L$ number of $W_{\ell}^{u,j}$ classes and there are $n$ possibilities for each of these numbers, there are $n^{4dr_L}$, a polynomial number of possibilities to enumerate in total. Thus we get the next lemma.

**Lemma 4.4.9.** We can guess rounding $W_{\ell}^{u,i}$ (sizes of rounding classes) and the numbers of items $w_{\ell}^{u,j}, t_{\ell}^{u,j}, s_{\ell}^{u,j}$ for $u \in [\lceil \frac{1}{\alpha} \rceil][d\setminus \ell], \ell \in [d], j \in \lceil \frac{1}{\lambda} \rceil$ in polynomial time.

**4.4.2.3 Assignment of items into rounded size:**

We have generated some rounded instances by specifying rounding and number of items belonging to each class. However, it is not clear for an item $v$, which dimension should be the nonaugmentable dimension in rounding. To find out whether there exists an assignment of items in $I$ into the rounding classes such that it will satisfy the requirements on the numbers of items we use the techniques of [156]. We specify the following flow network $G := (V, E)$:

- we have vertices $s$ and $t$ for source and sink,
- create a vertex for each item and for each class $W_{\ell}^{u,i}$,
- add edge of capacity 1 from $s$ to each item,
- for each item add an edge of capacity 1 from the item to all $d$ possible rounding classes where the item could belong to
- we add an edge of capacity $t_{\ell}^{u,j} + s_{\ell}^{u,j} + w_{\ell}^{u,j}$ from each class $W_{\ell}^{u,j}$ to $t$.

Using the algorithm of Dinic [49] we find a maximum integral flow from $s$ to $t$ in time $O(|E| \cdot |V|^2) \leq O(n + nd + dr_L)(n + dr_L)^2 = O_{d,\epsilon}(n^3)$. If it does not saturate some edge outgoing from $s$ or incoming to $t$, this means that we either cannot assign some item to any class or some class cannot have the required number of items assigned.
In this case we know that no valid assignment exists and the rounding is not feasible. We need to go back to the guessing phase and make another guess. Otherwise, we get a legal assignment of items into the rounding classes. Then we first separate and pack $I_T$. We pack $I_S$ and $I_\ell$ afterwards.

4.4.2.4 Packing of $I_T$:

To find a packing of $I_T$ into bins of type $B_T$ we construct a graph $(I', E)$ where $(p_i, p_j) \in E$ if $p_i$ and $p_j$ fit together into a bin i.e., $\sigma_{(p_i, p_j)} \leq 1$. Here we use vectors with original sizes and they are just assigned to some class $W_{ij}^\ell$ with size greater or equal to the size of the vector. We want to find a matching such that from each class $W_{ij}^\ell$, $t_{ij}^\ell$ vectors are saturated by the matching. We formulate the following LP:

\[
\sum_{u \in W_{ij}^\ell} x_{uv} = t_{ij}^\ell \quad \text{for all classes } W_{ij}^\ell \\
0 \leq x_e \leq 1 \quad \text{for all edge } e \in E.
\]

Note that this is a matching LP with a constant number of additional linear constraints, since the number of classes depends only on the value of $\epsilon$. To solve this LP we use an algorithm of Chekuri, Vondrák, and Zenklusen [29] which either returns a matching which covers between $t_{ij}^\ell$ and $(1 - \nu)t_{ij}^\ell$ items from each class $W_{ij}^\ell$ or it returns a certificate that this LP is infeasible (See Theorem 4.2.3). If the LP is infeasible, then either the rounding classes or the numbers $t_{ij}^\ell$ are incorrect and we try another guess.

The residual $\nu t_{ij}^\ell$ vectors from each class can be selected arbitrarily and we pack these $\leq 2\nu m_T$ items into $2\nu m_T$ number of additional bins using $(1 - \nu)m_T + 2\nu m_T \leq (1 + \nu)m_T$ bins in total. We choose $\nu = \epsilon$. 

96
4.4.2.5 Packing of $I_S$:

From each class $W_{u,j}^\ell$ we select arbitrary $s_{u,j}^\ell$ items not belonging to $I_T$ and pack them into separate $m_S$ bins.

4.4.2.6 Packing of $I_1, \ldots, I_d$:

For each $\ell \in [d]$ we take the leftover instances of $W_{u,j}^\ell$ which do not belong to $I_T$ nor $I_S$. We denote $I_\ell$ the union of all classes $W_{u,j}^\ell$. Thus we get the following lemma.

**Lemma 4.4.10.** For a correct guess, we can pack $I_\ell$ into $(1 + \epsilon)m$ bins in polynomial time. Moreover, the free space configuration will satisfy requirements of Lemma 4.3.5.

**Proof.** For a correct guess we can pack rounded items of type $I_\ell$ into $m_\ell + \lfloor \frac{cm_\ell}{2} \rfloor$ bins using Lemma 4.3.3 with $\ell$ to be the nonaugmentable dimension. Hence, total number of bins needed to pack all items in $\bigcup_\ell I_\ell$ is $\leq \sum_\ell (m_\ell + \lfloor \frac{cm_\ell}{2} \rfloor)$. 

4.4.2.7 Packing of small vectors:

Finally we pack the small vectors. From Lemma 4.4.10, $\sum_\ell m_\ell$ bins obtained in the previous step have at least the same small configuration as was the small configuration in corresponding bins of structured packing. So there is a feasible fractional packing of all small vectors into the space in $\sum_\ell m_\ell$ bins. We use another assignment LP similar to Lemma 4.3.5 to get the packing in at most another additional $\left\lceil \frac{\sum_\ell m_\ell}{2} \cdot (\epsilon + \frac{\epsilon^2}{d}) \right\rceil$ bins.

**Algorithm:**

97
Preprocessing stage:
Separate $S$  \hspace{1em} // vectors $< \beta$ in all coordinates
$I' := I \setminus S$

Guessing stage:
Generate rounding classes $W_{\ell}^{u}$ and guess $m', m_S, m_T, m_1, \ldots, m_d$

Guess round vectors for each dimension and create classes $W_{\ell}^{u,j}$

Guess numbers $w_{\ell}^{u,j}, t_{\ell}^{u,j}, s_{\ell}^{u,j}$

Packing stage:
Find assignment of items into rounding classes using flow-net, or
return to Guessing stage

Find multiobjective matching and pack $I_T$ into bins $B_T$, or
return to Guessing stage

Select $I_S$ arbitrarily according to numbers $s_{\ell}^{u,j}$ and pack it into bins $B_S$

\textbf{foreach} $\ell \in [d]$ \textbf{do}
  Pack $I_{\ell}$ into $B_{\ell}$ using Algorithm 1, or
  return to Guessing stage
\textbf{end}

Pack $S$ using assignment LP as in Lemma 4.3.5, or
return to Guessing stage

\textbf{return} packing $B_S, B_T, B_1, \ldots, B_d$

\textbf{Algorithm 2:} Algorithm for $d$-D vector packing with asymptotic approximation guarantee $\frac{(d+1)}{2} + \epsilon$

of Theorem 4.4.2. From Lemma 4.4.8 and 4.4.9, the guessing stage takes polynomial time. For a correct guess, $I_T$ items are packed into $(1+\epsilon)m_T$ bins, $I_S$ items are packed into $m_S$ bins. Remaining $I_\ell$ type items are packed in $\sum_{\ell}(m_{\ell} + \lfloor \frac{em_{\ell}}{2} \rfloor)$ bins and small
items are packed into additional \( \sum_{\ell} \frac{m_\ell}{2} \cdot (\epsilon + \frac{\epsilon^2}{d}) + 1 \) bins. Hence, total number of bins

\[
\leq (1 + \epsilon)m_T + m_S + \sum_{\ell} (m_\ell + \left\lfloor \frac{em_\ell}{2} \right\rfloor) + \sum_{\ell} \frac{m_\ell}{2} \cdot (\epsilon + \frac{\epsilon^2}{d}) + 1
\]

\[
\leq (1 + \epsilon + \frac{\epsilon^2}{2d}) (m_S + m_T + \sum_{\ell} m_\ell) + 1
\]

\[
\leq (1 + \epsilon + \frac{\epsilon^2}{2d}) (m' + 1)
\]

of Theorem 4.4.1. From Lemma 4.4.7, there is a there is a packing into \( \left\lceil \frac{d+1}{2} m \right\rceil \) bins of type \( B_S, B_T, B_1, \ldots, B_d \). Then using Lemma 4.4.2, we get the number of required bins is at most \( \left\lceil \frac{d+1}{2} m \right\rceil \cdot (1 + \epsilon + \frac{\epsilon^2}{2d}) + 1 \).

### 4.4.3 Tight Absolute Approximation of \((3/2 + \xi)\) for 2-D VBP

Now we will give an algorithm with absolute approximation ratio of \((1.5 + \xi)\) for 2-D vector packing. Note that even for 1-D bin packing the lower bound of absolute approximation is 3/2.

**Theorem 4.4.11.** Algorithm 3 has an absolute approximation ratio of \((1.5 + \xi)\) for 2-D vector packing.

**Proof.** Take \( \xi' = \xi/2 \). We give the algorithm in Algorithm 3. From Theorem 4.4.1, we can get a packing into \( 3m/2 + \xi'm + c_2 \) bins in polynomial time if \( m \) is the the number of bins in the optimal packing. There are two cases.

**Case A.** \( \xi'\text{Opt} \leq c_2 \): As there are at most \( c_1 = c_2 / \xi' \) (a constant) number of bins in the optimal solution, there are at most \( c_1 d / \beta \) big items. Thus there are only constant number of big configurations. So in polynomial time, we can find the big configurations in optimal packing and small vectors can be added using assignment LP except \( dc_1 \) number of small items. We can pack these small items into one additional bin by taking small \( \beta \) such that \( \beta dc_1 \leq 1 \).
Case B. $\xi'\text{Opt} > c_2$ : From Theorem 4.4.1, we can get a packing into $3m/2 + \xi'm + c_2 \leq 3m/2 + (2\xi')m \leq (1.5 + \xi)m$.

Algorithm:

| Find if there is a feasible packing into at most $m + 1$ bins for all $m \in [c_2/\xi']$ |
| Find the big configurations in optimal packing and small vectors (except possibly at most $dc_1$ number of small items) are then added using assignment LP in the remaining space |
| Pack remaining small items into one additional bin |
| Otherwise try the next value of $m$ |
| Run Algorithm 2 to find a feasible packing into $3m/2 + \xi'm + c_2$ bins |
| **return** any feasible packing found by the algorithm |

Algorithm 3: Algorithm for 2-D vector packing with absolute approximation guarantee $\frac{3}{2} + \xi$

This is interesting as it shows vector packing is much easier to approximate than geometric packing in terms of absolute approximability. On the other hand, though 1-D BP and 2-D VBP are quite distinct in asymptotic approximability, they are arbitrarily close for absolute approximability setting.

4.5 **Improved Approximation Algorithm for 2-D Vector Packing**

4.5.1 A structural theorem

In this section we show a structural theorem for $d$-dimensional vector packing. We will use this property in the analysis of algorithms for 2-D and $d$-D vector packing.

We will need the following concentration inequality [142] in the analysis.

**Lemma 4.5.1.** [Independent Bounded Difference Inequality] [142] Let $X = (X_1, \ldots, X_n)$ be a family of independent random variables, with $X_j \in A_j$ for $j = 1, \ldots, n$, and
$f : \prod_{j=1}^n A_j \rightarrow \mathbb{R}$ be a function such that
\[
f(x) - f(x') \leq c_j
\]
whenever the vectors $x$ and $x'$ differ only in the $j$-th coordinate. Let $\mathbb{E}[f(X)]$ be the expected value of the random variable $f(X)$. Then for any $t \geq 0$,
\[
P[f(X) - \mathbb{E}(f(X)) \geq t] \leq e^{-2t^2/\sum_{i=1}^n c_j^2}
\]

**Theorem 4.5.2.** Let $I$ be a set of items and $\tilde{I}$ is a rounding of items in $I$ such that all big items are rounded to $t_1$ (constant) types. Assume that there exists an optimal packing $\mathcal{P}$ of $\tilde{I}$ into $m$ bins. If for $\lceil (\ln \rho) z^* \rceil$ iterations we select a configuration $C' \in \mathcal{C}$ at random with probability $x_{C'}^s / z^*$ as in R & A framework applied to $I$ and $J$ is the residual set of elements at the end of these iterations, then with high probability there is a *systematic packing* $\mathcal{P}_J$ of $J$ into $\frac{(1+\epsilon')m}{\rho} + c_m$ bins where all big items are rounded to $t_1$ types (thus there are also only constant types of small configurations) and $c_m$ is a constant.

**Proof.** As there are $t_1$ (constant) types of big items and there can be at most $d/\beta$ big items in a configuration, there are $(d/\beta)^{t_1}$ types of big configurations and thus $(d/\beta)^{t_1}$ types of small configurations. Let $t_2 \leq (d/\beta)^{t_1}$ be the number of types of small configurations in $\mathcal{P}$. Let $\mathcal{R}_{j_1}$ be the set of big items of $j_1$’th type and $\mathcal{S}_{j_2}$ be the set of small items belonging to $j_2$’th type of small configurations in $\mathcal{P}$ for $j_1 \in [t_1]$ and $j_2 \in [t_2]$. Let $|\mathcal{R}_{j_1}| = r_{j_1}$ for $j_1 \in [t_1]$, i.e., $r_{j_1}$ is the number of items in the class of big items $\mathcal{R}_{j_1}$. Also let $s_{j_2}$ be the number of bins with small configuration of type $j_2$. Let $h_{j_1}^C$ be the number of items of type $\mathcal{R}_{j_1}$ in configuration $C$ and $g_{j_2}^C = 1$ if configuration $C$ has small configuration of type $j_2$. Consider the following $LP(\tilde{I})$
where $\mathcal{C}$ is the set of configurations of $\tilde{I}$:

$$\min \sum_{C \in \mathcal{C}} x_C$$

$$\sum_{C \in \mathcal{C}} h^C_{j_1} x_C \geq r_{j_1} \quad \forall j_1 \in [t_1]$$

$$\sum_{C \in \mathcal{C}} g^C_{j_2} x_C \geq s_{j_2} \quad \forall j_2 \in [t_2]$$

$$x_C \geq 0 \quad \forall C \in \mathcal{C}$$

(23)

Let $t = t_1 + t_2$. As the LP has $t$ (which is a constant) number of constraints and the optimal integral solution of $LP(\tilde{I})$ has value $m$,

$$m \leq LP(\tilde{I}) + t$$

(24)

Intuitively in $LP(\tilde{I})$ we consider small configurations of each bin as a single item. Now define another instance $J'$ such that it contains $\frac{r_{j_1}(1+\epsilon_1)}{\rho}$ big items of type $j_1 \in [t_1]$ and $\frac{s_{j_2}(1+\epsilon_1)}{\rho}$ items of type $j_2 \in [t_2]$ such that the size of each item in type $j_2$ is equal to the size of $j_2$'th type of small configuration. Intuitively $J'$ is a shrunk down version of $\tilde{I}$ where each small configuration is replaced by a single item of that same size.

Now consider the following $LP(J')$:

$$\min \sum_{C \in \mathcal{C}} x_C$$

$$\sum_{C \in \mathcal{C}} h^C_{j_1} x_C \geq \frac{r_{j_1}(1+\epsilon_1)}{\rho} \quad \forall j_1 \in [t_1]$$

$$\sum_{C \in \mathcal{C}} g^C_{j_2} x_C \geq \frac{s_{j_2}(1+\epsilon_1)}{\rho} \quad \forall j_2 \in [t_2]$$

$$x_C \geq 0 \quad \forall C \in \mathcal{C}$$

(25)

As the right hand side of constraints in $LP(\tilde{I})$ and $LP(J')$ differ by a factor of $\frac{(1+\epsilon_1)}{\rho}$, we get:

$$\frac{(1+\epsilon_1)}{\rho} LP(\tilde{I}) = LP(J')$$

(26)
Let $\text{Opt}(J')$ be the optimal integral solution for the above LP.

Now consider applying the same rounding as in $\tilde{I}$ to the items in $J$. Let us denote this instance to be $(\tilde{I} \cap J)$. To conclude the proof we will show that the items in $(\tilde{I} \cap J)$ can be packed in $(1 + \epsilon_2)\text{Opt}(J')$ bins.

**Lemma 4.5.3.** $\text{Opt}(\tilde{I} \cap J) \leq (1 + \epsilon_2)\text{Opt}(J')$.

**Proof.** As $J$ is the set of residual items that are not covered by any bins selected in $\lceil (\ln \rho)z^* \rceil$ iterations in packing using randomized rounding, thus for any item $i \in I$,

$$P(i \in J) = (1 - \sum_{C \ni i} x_C^*/z^*)^{\lceil (\ln \rho)z^* \rceil}$$

$$\leq (1 - \sum_{C \ni i} x_C^*/z^*)^{(\ln \rho)z^*}$$

$$\leq (1 - 1/z^*)^{(\ln \rho)z^*}$$

$$\leq e^{(-\ln \rho)}$$

$$= \frac{1}{\rho}$$

Here inequality (27) follows from the fact that $\sum_{C \ni i} x_C^* \geq 1$ for all $i \in I$ and inequality (28) follows from the fact that $(1 - \frac{1}{x})^\alpha x \leq e^{-\alpha}$ for $x > 0$.

Hence from (29), $E[|\mathcal{R}_i \cap J|] \leq |\mathcal{R}_i|/\rho$ for all $i \in [t_1]$.

Now consider the function $f_i^B(x) = \mathcal{R}_i \cap J$, i.e., the number of items of type $\mathcal{R}_i$ in $J$. This is a function of $X = (X_1, \ldots, X_{\lceil (\ln \rho)z^* \rceil})$, i.e., $\lceil (\ln \rho)z^* \rceil$ independent random variables (selected configurations in randomized rounding phase of R & A framework) where $X_i$ corresponds to the random variable for the configuration selected in $i$th iteration. Now changing value of any of these random variables may lead to selection of a different configuration $C'$ in place of configuration $C$ in the corresponding iteration. Let vectors $x$ and $x'$ differ only in $j$'th coordinate, i.e., a different configuration $C'$ is selected in place of configuration $C$. This might lead to a different set of residual...
items $J'$. Then
\[
 f_B^i(x) - f_B^i(x') \leq (|\mathcal{R}_i \cap J| - |\mathcal{R}_i \cap J'|) \\
 \leq \max\{|\mathcal{R}_i \cap C|, |\mathcal{R}_i \cap C'|\} \\
 \leq 1/\beta
\]  

(30)

Here inequality (30) follows from the fact that there can be at most $\beta$ items of type $\mathcal{R}_i$ in a bin as big items are $\geq \beta$ in at least one of the dimensions.

Therefore, from independent bounded difference inequality, we get,
\[
P[f_B^i(X) - \mathbb{E}(f_B^i(X)) \geq \gamma z^*] \leq e^{-2(\gamma z^*)^2/(\ln \rho \cdot t_2)}
\]

Thus in the asymptotic case (when $z^*$ is sufficiently large, i.e., $\gg \frac{\ln \rho \cdot t_2}{z^*}$) we can take union bound over all $t_1$ cases and with high probability for all large item classes, $f_B^i(X) - \mathbb{E}(f_B^i(X)) < \gamma z^*$. We can take $\gamma = \frac{\epsilon_3^2}{t_1 t_2 \rho}$. As in the packing of $J'$, we have $|\mathcal{R}_i|/\rho$ items in $\tilde{I} \cap J$ in the slots of same type items in $J'$. Then we can pack all remaining big items in at most
\[
t_1 \gamma z^* \leq \frac{\epsilon_3^2}{t_2 \rho} \leq \frac{\epsilon_3^2 \max(I)}{t_2} \leq \frac{\epsilon_3^2 \max(J)}{t_2} \leq \frac{\epsilon_3^2 \max(J)}{t_2} \leq \frac{\epsilon_3^2 \max(J)}{t_2} \leq \frac{\epsilon_3^2 \max(J)}{t_2}
\]

extra bins.

Now let us consider the small items. Consider a small configuration $\mathcal{S}_j = (h_1, \ldots, h_d)$. Let $\mathcal{S}_j$ be the set of items in all small configurations of type $\mathcal{S}_j$ and $s_j$ be the number of small configurations of type $\mathcal{S}_j$. Let function $f_S^k$ be $\sum_{v \in \mathcal{S}_j \cap J} v^k \cdot \frac{1}{h_k}$, i.e., the length of items in $\mathcal{S}_j \cap J$ in $k$'th dimension scaled by a factor $\frac{1}{h_k}$. Intuitively the scaling makes each dimension of $\mathcal{S}_j$ to be one. This is again function of $\lceil (\ln \rho) z^* \rceil$ independent random variables (selected configurations). Thus as above, whenever the vectors $x$ and $x'$ differ only in one coordinate, we get:
\[
f_{S_j}^k(x) - f_{S_j}^k(x') \leq \sum_{v \in \mathcal{S}_j \cap J} v^k \cdot \frac{1}{h_k} - \sum_{v \in \mathcal{S}_j \cap J'} v^k \cdot \frac{1}{h_k} \\
\leq \max\{ \sum_{v \in \mathcal{S}_j \cap C} v^k \cdot \frac{1}{h_k}, \sum_{v \in \mathcal{S}_j \cap C'} v^k \cdot \frac{1}{h_k} \} \\
\leq \frac{1}{h_k} \cdot h_k \leq 1
\]  

(31)
Thus, from independent bounded difference inequality, we get,

$$\mathbb{P}[f^k_{S_j}(X) - \mathbb{E}(f^k_{S_j}(X)) \geq \gamma z^*] \leq e^{-2(\gamma z^*)^2/((\ln \rho) z^*)}$$

Now if $s_j < \frac{\epsilon z^*}{t_1 t_2}$, one can pack these small items in all $t_2$ classes of small configurations into at most $\epsilon z^*/t_1$ additional bins. Otherwise if $s_j \geq \frac{\epsilon z^*}{t_1 t_2}$, then with high probability, $f^k_{S_j}(X) \leq (\frac{\Delta}{\rho} + \gamma z^*) \leq (1 + \epsilon_3) \cdot \frac{s_j}{\rho}$. Thus, all small items in $S_j \cap J$ can be packed into the corresponding small configurations of $\frac{1 + \epsilon_3}{\rho} s_j$ bins in $J'$ fractionally by assigning each item to each bin with $\frac{\rho}{(1 + \epsilon_3)s_j}$ fraction.

So using assignment LP we can get an integral packing of all items into corresponding small configurations of $\frac{1 + \epsilon_3}{\rho} s_j$ bins except $\lceil d \cdot \frac{(1 + \epsilon_3)}{\rho} s_j \rceil$ items. However the original sizes of these $\lceil d \cdot \frac{(1 + \epsilon_3)}{\rho} s_j \rceil$ items are $\leq \beta$ in all dimensions. So, we can pack them in extra $\leq \lceil \frac{d \beta}{\rho} \rceil \leq \lceil (d \beta + \beta^2 d) \frac{(1 + \epsilon_3)}{\rho} s_j \rceil$ number of bins.

Therefore in the asymptotic case, $\text{Opt}(\tilde{I} \cap J)$, the total number of bins in the optimal packing of $(\tilde{I} \cap J)$ is:

$$(1 + \epsilon_3) \text{Opt}(J') + \frac{\epsilon_3 \text{Opt}(J')}{t_1} + [(d \beta + \beta^2 d) \frac{(1 + \epsilon_3)}{\rho} \text{Opt}(J')] + \frac{\epsilon_3^2 \text{Opt}(J')}{t_2} \leq (1 + \epsilon_2) \text{Opt}(J')$$

by choosing $\epsilon_2 > \epsilon_3 + \epsilon_3/t_1 + \epsilon_3^2/t_2 + [(d \beta + \beta^2 d)(1 + \epsilon_3)/\rho]$.

Therefore in the asymptotic case when $z^*$ is sufficiently large,

$$\text{Opt}(J) \leq \text{Opt}(\tilde{I} \cap J) \leq (1 + \epsilon_2) \text{Opt}(J')$$

$$\leq (1 + \epsilon_2) (LP(J') + t)$$

$$\leq \frac{(1 + \epsilon_1)(1 + \epsilon_2)}{\rho} LP(\tilde{I}) + O(1)$$

$$\leq \frac{(1 + \epsilon')}{\rho} LP(\tilde{I}) + O(1)$$

$$\leq \frac{(1 + \epsilon')}{\rho} \text{Opt}(\tilde{I}) + c_m$$

Inequality (32) follows from Lemma 4.5.3. Inequality (33) follows from the fact that there are $t$ number of constraints in LP (25). Inequality (34) follows from inequality
Inequality (35) is obtained by taking $\epsilon' = (1 + \epsilon_1)(1 + \epsilon_2) - 1$ and $c_m$ is the additive constant.

This completes the proof.

Note that one can find such a (near)-optimal systematic packing in polynomial time by guessing the round vectors as we discussed in the previous section. So we need to show existence of structural packing with better guarantees.

4.5.2 Approximation Algorithm for 2-D vector packing:

Now we show a $1 + \ln(3/2)$ asymptotic approximation algorithm for any set of items $I$. Let us call a bin to be compactly packed if there are two items $a, b$ (we call them matching elements for the bin) in the bin such that $\sigma_{\{a, b\}} \geq (1 - \delta, 1 - \delta)$ or if there is only one element $a$ such that $a \geq (1 - \delta, 1 - \delta)$. We call all other bins to be noncompact. Note that compact bins can still contain other set of elements $S$ apart from $a, b$ with $\sigma_S \leq (\delta, \delta)$. Let $m_C, m_N$ be the number of compact bins and noncompact bins respectively in the optimal solution. As defined earlier, $W_{ij}^c$ is the classes corresponding to the round vectors. Let $c_{ij}^c, w_{ij}^c$ are the number of items of type $W_{ij}^c$ that are in the compact bins and noncompact bins respectively. Let $c_{ij}^c = t_{ij}^c + s_{ij}^c$ where $t_{ij}^c$ and $s_{ij}^c$ correspond to items that are in compact bins that contain two and one big item respectively. Note than we can assign any big item $v$ in compact bins to any size class $W_{ij}^c$ if $r_{ij}^c \geq v$ where $r_{ij}^c$ is the round vector corresponding to the class $W_{ij}^c$.

Given $m_C$ compact bins, we can remove the items $\leq (\delta, \delta)$ in the compact bins and greedily pack them into additional $\left\lceil \frac{m_C}{1/\delta} \right \rceil \leq (\delta + 2\delta^2)m_C + 1$ extra bins. After removing these items remaining compact bins are of type $B_S$ or $B_T$. On the other hand, given a noncompact bin $B$, from Lemma 4.4.3 we get that there is a set $S$ such that $B \setminus S$ contains slack $\geq \delta$ in at least one dimension and either $S = \{p\}$ where $p \in B$ with $p \leq 1/2$ and $p \geq \delta$ in at least one dimension, or there is a subset $S$ of
small items in $B$ such that $\sum_{p \in S} p \leq 2\delta$ in both dimensions. Thus we can round $p$
to $1/2$ and other big items using Algorithm 1 to have constant types of big items in
$\frac{3}{7}m_N + 1$ bins.

Thus there is a packing into $(1 + 2\delta + \delta^2)m_C + \frac{3}{2}(1 + \epsilon + \frac{\epsilon^2}{7})m_N + 2$ bins such that
$m_C(1 + \delta)$ bins are of type $B_S$ or $B_M$, whereas big items in $\frac{3}{7}(1 + \epsilon + \frac{\epsilon^2}{7})m_N + 1$ bins
are rounded to constant types and other $(\delta + 2\delta^2)m_C + 1$ bins contain small items.

Then as in the previous section, we can guess this rounded heights and widths of big
items and small configurations. Then using [16] we can find the structured packing
in polynomial time.

The Algorithm:
Guessing stage:

Guess $m := \text{Opt}(I)$ using binary search.

Guess $m_C, m_N$.

Guess round vectors for each dimension and create classes $W^i_{\ell}$.

Guess numbers $c^i_\ell, t^i_\ell, s^i_\ell, w^i_\ell$.

Go to Packing stage.

Packing stage:

A. Packing using matching: Use multiobjective multibudget matching on the original item sizes to pack $t^i_\ell$ items from class $W^{i}_\ell$ and pack $s^i_\ell$ items from class $W^{i}_\ell$ separately into total $(1 + \delta)m_C$ bins, or return to Guessing stage.

B. Packing using randomized rounding of configuration LP: Let $I'$ be the set of remaining items. Solve the configuration LP restricted to $I'$. Let $x^*$ be the (near)-optimal solution of the LP relaxation and let $z^* = \sum_{C \in C} x^*_C$. Let $r$ be the number of configurations in the support of $x^*$. Initialize a $|C|$-dimensional binary vector $x^r$ to be an all-0 vector. Let $\rho = 3/2$. For $\lceil (\ln \rho)z^* \rceil$ iterations repeat the following: select a configuration $C'' \in C$ at random with probability $x^r_{C''} / z^*$ and let $x^r_{C''} := 1$.

C. Packing using 3/2 approximation algorithm: Let $S$ be the remaining set of items not covered by $x^r$ i.e., $i \in S$ if and only if $\sum_{C \ni i} x^r_C = 0$. On set $S$, apply the 3/2 approximation algorithm $A$ that rounds the big items to $O(1)$ types. Remaining small items (if any) are packed using assignment LP in the guessed spaces for the small items or using additional bins. Let $x^a$ be the solution returned by $A$ for the residual instance $S$.

D. If $||x^a||_0 \leq (1 + \epsilon + \epsilon^2/d)m_N + (\delta + 2\delta^2)m_C + 2$ then return $x = x^r + x^a$ as the packing of $J$, or return to Guessing stage for the next guess.

Algorithm 4: Overview of the algorithm
4.5.3 Analysis of the Algorithm

**Theorem 4.5.4.** Algorithm 4 is a \((1 + \ln(\frac{3}{2} + \epsilon'))\) asymptotic approximation algorithm for 2-dimensional vector packing.

**Proof.** The binary search to find \(m\) takes \(O(\log n)\) time. For each guess there are \(O(n^2)\) guesses for \(m_C, m_N\). Also as there are total \(t_L\) (constant) number of round vectors, they can be guessed in \(O(n^{t_L})\) time. Thus in polynomial time we can guess the suitable values for the guessing stage in the algorithm.

Now there are three steps in the packing stage of the algorithm. In the first step we pack \((1 + \delta)m_C\) number of bins using multi-objective multi-budget matching on the original item sizes. In the second step we pack using randomized rounding of configurations and we use at most \([\ln \rho z^*]\) ≤ \(\ln \rho \cdot m + 1\) bins. We choose \(\rho = \frac{3}{2}\). In the last step we use the \(3/2\) approximation algorithm for 2-D vector packing that rounds all big items to \(O(1)\) types to pack the remaining items in \(S\) into \((1 + \epsilon + \epsilon^2/d)m_N + (\delta + 2\delta^2)m_C + 2\) number of bins. So, the total number of bins needed is \(= (1 + \delta)m_C + (\ln \rho) m + (1 + \epsilon + \epsilon^2/d)m_N + O(1) \leq (1 + \epsilon' + \ln \rho)(m_M + m_N) + O(1)\) bins. This gives \((1 + \ln(\frac{3}{2}) + \epsilon')\) asymptotic approximation if the algorithm returns a feasible solution where \(\epsilon' = \epsilon + \epsilon^2/d\).

To complete the proof we need to show that the algorithm always returns a feasible packing for a correct guess.

In the packing using matching step, the items we pack in \(m_C\) bins might not be the same items in \(B_C^{Opt}\), the matching bins in the optimal solution. However the packed items in these \(m_C\) bins are very similar to the items in \(B_C^{Opt}\). We pack \(c^i_j\) items from \(W_{ij}^\ell\) using matching. So, for a correct guess at the end of packing using matching each class \(W_{ij}^\ell\) has only \((c^i_j + w_{ij}^\ell) - c^i_j = w_{ij}^\ell\) items left.

Now \(S\) is the set of residual items that are not covered by any bins selected in \([\ln \rho z^*]\) iterations in packing using randomized rounding. Now from Lemma 4.4.7, the rounded items in the noncompact bins in the optimal packing of \(I\) can be packed.
into \( \leq \frac{3}{2}(1+\epsilon+\epsilon^2/d)m_N+1 \) bins. Thus from Theorem 4.5.2, the big items in \( S \) can be rounded to \( O(1) \) types and be packed in \( (\frac{3}{2}(1+\epsilon+\epsilon^2/d)m_N+1)/\rho \leq (1+\epsilon+\epsilon^2/d)m_N+1 \) bins. On the other hand small items in \( m_C \) bins can be greedily packed in at most \( (\delta + 2\delta^2)m_C + 1 \) bins.

This concludes the proof. \( \square \)

### 4.6 Improved Approximation Algorithm for \( d \)-Dimensional Vector Packing

First let us prove the existence of a systematic packing in \( d \)-dimensional packing.

**Theorem 4.6.1.** If there exists a feasible packing of a set of items \( I \) into \( m \) bins, there is a systematic packing into at most \( (2+\epsilon'')m + 1 \) bins such that \( m \) bins contain at most \( (d-1) \) items in each of them and all other bins have \( O(1) \) types of big items and small configurations.

**Proof.** Let \( V_i, S_i \) are the sets from Lemma 4.4.5, for bin \( B_i \) for \( i \in [m] \) and \( B_i = B \setminus (V_i \cup S_i) \). Then \( B_i \) bins can be rounded to a systematic packing using Algorithm 1. If \( |S_i| \leq 1 \), pack \( S_i \) and \( V_i \) together in one bin \( B' \). Note that \( |B'| \leq (d-1) \). Otherwise, pack \( V_i \) in one bin. As elements in \( S_i \) are \( \leq \kappa \delta \) in all dimensions, we pack them separately in additional \( \lceil m/((\frac{1}{\kappa \delta 2})) \rceil \leq \kappa \delta (1 + 2\kappa \delta) \cdot m + 1 \) bins. Take \( \epsilon'' = \kappa \delta \cdot (1 + \kappa 2\delta) \). \( \square \)

Let \( I_D \) be the set of items in those \( m \) bins containing at most \( (d-1) \) items. Let \( I_N = I \setminus I_D \). For the bins that have \( O(1) \) types of big items, we can guess their round vectors and find the (near)-optimal packing of the rounded vectors. As previously let \( W_i^{ji} \)’s correspond to the classes of the rounded vectors. We can arbitrarily assign vectors in \( I_D \) to any class with size dominating the size of the vector.
Guessing stage:

Guess $m := \text{Opt}(I)$ using binary search

Guess round vectors for each dimension and create classes $W_{ij}^\ell$

Initialize packed bins $\mathcal{B} = \emptyset$. Go to Packing stage.

Packing stage:

A. Solve the configuration LP. Let $x^*$ be the (near)-optimal solution of the LP relaxation and let $z^* = \sum_{C \in \mathcal{C}} x_C^*$. Let $r$ be the number of configurations in the support of $x^*$.

B. Packing using randomized rounding of configuration LP: Take $\rho = d/2$. For $\lceil (\ln \rho) z^* \rceil$ iterations repeat the following: select a configuration $C'' \in \mathcal{C}$ at random with probability $x_{C''}^*/z^*$ and $\mathcal{B} := \mathcal{B} \cup \{C''\}$.

C. Packing using matching: Let $S$ be the set of items left at the end of randomized rounding. Guess numbers

$$\mu_{ij}^\ell = |(I_D \cap S) \cap W_{ij}^\ell|, \nu_{ij}^\ell = |(I_N \cap S) \cap W_{ij}^\ell|.$$ 

Use multiobjective multibudget matching to pack $\mu_{ij}^\ell$ items from class $W_{ij}^\ell$ into $\lceil \frac{3(1+3\epsilon)}{2} m \rceil$ bins, or return to Guessing stage

D. Packing using $O(1)$ rounding based approximation algorithm: Let $J$ be the remaining set of items. On set $J$, apply the approximation algorithm $\mathcal{A}$ that guesses the correct rounding of the big items to $O(1)$ types. Remaining small items (if any) are packed using assignment LP in the guessed spaces for the small items. If all remaining items in $J$ are packed in $2(1 + \epsilon)(m/d) + \epsilon''$ bins, add the returned packing in $\mathcal{B}$ and return $\mathcal{B}$ as the final packing or return to Guessing stage

Algorithm 5: Overview of the algorithm
4.6.1 Analysis

**Lemma 4.6.2.** If there exists a feasible packing of $2n_1$ items into a packing $P_1$ of $n_1$ bins, one can pack them into a packing $P_2$ of $3n_1/2$ bins in polynomial time.

**Proof.** We will show there exists $3n_1/2$ bins such that each bin contain one or two items. Then we can find a packing of $2n_1$ items into $3n_1/2$ bins using maximum matching. Assume $x$ be number of bins in $P_1$ that contain odd number of items. Pack one item from each bin with odd number of items, separately into $x$ bins. Remaining bins contain even number of items. So, they can be packed in $(2n_1 - x)/2$ bins using matching. Hence total $x + \left(\frac{2n_1-x}{2}\right) \leq n_1 + \frac{x}{2} \leq 3n_1/2$ bins are sufficient.

**Theorem 4.6.3.** Algorithm 5 returns a solution with asymptotic approximation ratio $\ln(d/2) + 1.5 + o(1) + \epsilon' \approx \ln(d) + 0.807 + o_d(1) + \epsilon'$.

**Proof.** In packing using randomized rounding we use at most $\ln(d/2)(\text{Opt}) + 1$ bins. If there is a feasible solution, packing with matching takes $\frac{3(1+3\epsilon)}{2}m + 1$ and packing using $O(1)$ rounding based algorithm takes $2(1 + \epsilon)(m/d) + \epsilon''$ bins.

Now we show the algorithm always returns a feasible packing. From (29), for any item $i \in I$, $P(i \in S) = 1/\rho = \frac{2}{d}$. Thus with high probability $|I_D \cap S| \leq (1 + \epsilon)dm/(d/2) = 2(1 + \epsilon)m$ when $m$ is sufficiently large. Using Lemma 4.6.2 we can pack these $2(1 + \epsilon)m$ items into $\frac{3(1+\epsilon)}{2} \cdot (1+\epsilon)m \leq \frac{3(1+3\epsilon)}{2} \cdot m$ bins using multi-objective multi-budget matching.

The big items in $I_N$ can be rounded to $O(1)$ types and packed into $m$ bins. Thus from Theorem 4.5.2, big items in $J$ can be rounded to $O(1)$ types and be packed in $\frac{(1+\epsilon)m}{(d/2)}$ bins. Additional small items can be packed using assignment LP in the remaining space and at most $\epsilon''m$ additional bins. Thus in the asymptotic case total number of bins needed $\leq (\ln(d/2) + \frac{3}{2} + \frac{9\epsilon}{2} + 2(1+\epsilon))\text{Opt} + 2 \leq (\ln(d/2) + 1.5 + o_d(1) + \epsilon') + 1$ where $\epsilon' = \epsilon'' + \frac{9\epsilon}{2}$. This concludes the proof.

Note that one can derandomize the algorithm using standard arguments in [11].
4.7 Conclusion:

We remark that our bounds for small values of $d > 2$, can possibly be improved using multi-objective multi-budget variant of $d$-dimensional matching. However this approach may not be useful for large $d$ as there is already $\Omega(d/\ln d)$ lower bound for $d$-dimensional matching.

As there is still no known explicit hardness for $d$-D VBP as function of $d$, it will be interesting to show a $f(d)$ hardness using some reduction from $d$-dimensional matching or other related problems.
Chapter V

WEIGHTED BIPARTITE EDGE COLORING

In this chapter we discuss our results in weighted bipartite edge coloring which have been inspired from study of Clos networks [33] and also generalize classical coloring problems.

Clos Networks. Clos network were introduced by Clos [33] in the context of designing interconnection networks with small number of links to route multiple simultaneous connection requests such as telephone calls. Since then it has found various applications in data communications and parallel computing systems [113, 102]. The symmetric 3-stage Clos network is generally considered to be the most basic multistage interconnection network. Let $C(m, \mu, r)$ denote a symmetric 3-stage Clos network, where the input (first) stage consists of $r$ crossbars of size $m \times \mu$, the center (second) stage consists of $\mu$ crossbars of size $r \times r$ and the output (third) stage consists of $r$ crossbars of size $\mu \times m$. Moreover, there exists one link between every center switch and each of the $r$ input or output switch. No link exists between other pair of switches. An example $C(2, 3, 4)$ network is shown in Figure 11.

A request frame is a collection of connection requests between inlets and outlets in the network such that each inlet or outlet is associated with at most one request. A request frame is *routable* if all requests are routed through a middle switch such that no two requests share same link. An interconnection network is called to be *rearrangeably nonblocking* if all request frames are routable. In the classic switching environment all connection requests fully use a link and all have same bandwidth. However in present networks, different requests might have different bandwidths (due
Figure 11: A Clos network with $m = 2, \mu = 3, r = 4$ [54]

to wide range of traffic such as voice, video, facsimile etc.) and may be combined in a given link if the combined request does not exceed the link capacity. In this multirate setting, a connection request is a triple $(i, j, w)$ where $i, j, w$ are inlet, outlet and demand of the connection respectively, and all links have capacity one. Here a request frame is a collection of connection requests between inlets and outlets in the network such that the total weight of all requests in the frame for any particular inlet or outlet is at most one. The central question in 3-stage Clos networks is finding the minimum number of switches (crossbars) $\mu (= \mu(m, r))$ in the middle stage such that all request frames are routable. It is more interesting to obtain bounds independent of $r$.

**Weighted Bipartite Edge Coloring.** Nonblocking rearrangeable properties of 3-stage Clos network $C(m, \mu, r)$ can be translated to the following graph theoretic
problem. Formally, in weighted bipartite edge coloring problem, we are given an edge-weighted bipartite (multi)-graph $G := (V, E)$ with bipartitions $A, B$ ($|A| = |B| = r$) and edge weights $w : E \rightarrow [0, 1]$. Let $w_e$ denote the weight of edge $e \in E$. The goal is to obtain a proper weighted coloring of all edges with minimum number of colors. An edge coloring of the weighted bipartite graph is called a proper weighted coloring if the sum of the same color edges incident to a vertex is at most one for any color and any vertex. Here the sets $A$ and $B$ correspond to the input and output switches, edge $(u, v)$ corresponds to a request between input switch $u$ and output switch $v$. A routable request frame translates into the condition that weights of all incident edges to any vertex can be proper weighted colored using $m$ colors (or packed into $m$ unit sized bins) and the switches in the middle stage correspond to the colors (or bins). We refer the reader to Correa and Goemans [41] for more connection with Clos networks.

**Connection with Other Classical Problems.** Weighted bipartite edge coloring also generalizes two classical problems: bin packing and edge coloring problem. If all edge weights are one, this problem reduces to classical edge coloring problem. König’s Theorem [127] gives an optimal coloring in this case where the number of colors used is exactly the maximum degree of the graph. We will give an alternate proof of König’s Theorem from skew-supermodularity in Section 5.1. Edge-coloring problem has been one of the central problems in graph theory and discrete mathematics since its appearance in 1880 [180] in relation with the four-color problem. Edge coloring problems have been pivotal in the development of several algorithmic techniques. It has plentiful applications in modeling various problems related to scheduling, partitioning and computer networks. On the other hand, if there are only one vertex in each partition in the bipartite graph, this problem reduces to classical bin packing problem.
Previous Work. Three classical results on edge coloring are König’s theorem [127] for coloring a bipartite graph with $\Delta$ colors, Vizing’s theorem [185] for coloring any simple graph with $\Delta+1$ colors and Shannon’s theorem [172] for coloring any multigraph with at most $3\Delta/2$ colors where $\Delta$ is the maximum degree of the graph. The chromatic index of a graph is the number of colors required to color the edges of the graph such that no two adjacent edges have the same color. Though one can find optimal edge coloring for a bipartite graph in polynomial time using König’s theorem, Holyer [99] showed that it is even NP-hard to decide whether the chromatic index of a cubic graph is 3 or 4. We refer the readers to [177] for a survey on edge coloring.

Now let us review the literature related to weighted bipartite edge coloring. First let us introduce some notation. Let $\chi'_w(G)$ denote the minimum number of colors needed to obtain a proper weighted coloring of $G$. Given positive integers $m$ and $r$, let $\mu(m, r) = \max_G \chi'_w(G)$ where the maximum is taken over all bipartite graphs $G = (A \cup B, E)$ with $|A| = |B| = r$ and where $m$ is the maximum over all the vertices of the number of unit-sized bins needed to pack the weights of incident edges. Chung and Ross [32] made the following conjecture:

Conjecture 5.0.1. Given an instance of the weighted bipartite edge coloring problem, there is a proper weighted coloring using at most $2m - 1$ colors where $m$ denotes the maximum over all the vertices of the number of unit-sized bins needed to pack the weights of edges incident at the vertex. In other words, $\mu(m, r) \leq 2m - 1$.

When the weight function $w : E \rightarrow I$ is restricted to a subinterval $I \subseteq [0, 1]$, then we denote the minimum number of colors by $\mu_I(m, r)$. Slepian [174] showed that $\mu_{[1,1]}(m, r) = m$ using König’s theorem. Melen and Turner [144] showed that $\mu_{[0,B]}(m, r) \leq \frac{m}{1-B}$ for $B \leq 1$. In particular $\mu_{[0,1/2]}(m, r) \leq 2m - 1$. There has been a series of work improving the bounds for $\mu(m, r)$ [32, 54, 150, 41]. The best known
lower bound for $\mu(m, r)$ is $5/4$ due to Ngo and Vu [150]. Correa and Goemans introduced a novel graph decomposition result and perfect packing of an associated continuous one-dimensional bin packing instance to show $\mu(m, r) \leq 2.5480m + o(m)$. The present best algorithm is due to Feige and Singh [68] who showed $\mu(m, r) \leq 9/4$. Their result holds even if $m$ is the maximum over all the vertices of the total weight of edges incident at the vertex. For related results in general graphs we refer the readers to [68].

**Our results.** Our main result is to make progress towards resolution of Conjecture 5.0.1 by showing $\mu(m, r) \leq \frac{20m}{9} + o(m)$.

**Theorem 5.0.2.** There is a polynomial time algorithm for the weighted bipartite edge coloring problem which returns a proper weighted coloring using at most $\lceil 2.2223m \rceil$ colors where $m$ is denotes the maximum over all the vertices of the number of unit-sized bins needed to pack the weights of incident edges.

Our algorithm is purely combinatorial. It uses König’s theorem and a greedy algorithm on a subgraph created based on a threshold value for the edges. Then remaining edges are added greedily to the colors used in the algorithm. The greedy algorithm is inspired from the first fit decreasing height heuristic for bin packing. We also show $\lceil 2.2m \rceil$ colors are sufficient when all edge weights are $> 1/4$. The analysis is nontrivial and uses configuration LP and other structural property of bin packing. These techniques might be useful in other related problems in Clos networks.

**Overview of the chapter.** In Section 5.1 we present an alternate proof of König’s Theorem using skew-supermodularity. Finally in Section 5.2 we present our edge-coloring algorithm and the related analysis.
5.1 König’s edge-coloring theorem

In this section we give an alternate proof of the König’s Theorem.

**Theorem 5.1.1.** [127] Given a bipartite graph $G = (V, E)$ there exists a coloring of edges with $\Delta = \max_{v \in V} \deg_E(v)$ colors such that all edges incident at a common vertex receive a distinct color.

First let us define some preliminaries.

**Definition 5.1.2. Intersecting family:** A family $\mathcal{C}$ is called an intersecting family if for all $T, U \in \mathcal{C}$ with $T \cap U \neq \emptyset$, both $T \cap U$ and $T \cup U$ are in $\mathcal{C}$.

**Definition 5.1.3. Intersecting supermodular:** Given an intersecting family $\mathcal{C}$ and a function $f : \mathcal{C} \to \mathbb{R}$ is called intersecting supermodular if for all $T, U \in \mathcal{C}$ with $T \cap U \neq \emptyset$, we have:

$$f(T) + f(U) \geq f(T \cup U) + f(T \cap U)$$

**Definition 5.1.4. Skew-supermodular functions:** Given a finite ground set $V$ and a set function $g : 2^V \to \mathbb{Z} \cup \{\infty\}$ is called skew-supermodular if at least one of the following two inequalities holds for every $X, Y \subseteq V$:

$$g(X) + g(Y) \leq g(X \cap Y) + g(X \cup Y) \quad (37)$$

$$g(X) + g(Y) \leq g(X \setminus Y) + g(Y \setminus X) \quad (38)$$

5.1.1 Supermodular Coloring and Extension

Schrijver [165] showed the following theorem on supermodular coloring:

**Theorem 5.1.5.** [165] Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be intersecting families of subsets of a set $S$, let $g_1 : \mathcal{C} \to \mathbb{Z}$ and $g_2 : \mathcal{C} \to \mathbb{Z}$ be intersecting supermodular, and let $k \in \mathbb{Z}_+$. Then $S$ can be partitioned into classes $L_1, L_2, \ldots, L_k$ such that

$$g_i(U) \leq |\{j \in [k] : L_j \cap U \neq \emptyset\}| \quad (39)$$
for each \( i = 1, 2 \) and each \( U \in C_i \) if and only if
\[
g_i(U) \leq \min\{k, |U|\} \quad (40)
\]

König’s edge-coloring theorem for bipartite graphs is a special case of above theorem. Let \( V_1 \) and \( V_2 \) be the color classes for bipartite graph \( G \). Let \( C_i := \{\delta(v) | v \in V_i\} \) for \( i = 1, 2 \). If we define \( g_i(\delta(v)) := |\delta(v)| \) for all \( v \in V_i \) for \( i = 1, 2 \), then we get that any bipartite graph can be colored by \( \Delta(V) := \max_{v \in V} \delta(V) \) colors.

Bernáth and Király showed the following skew-supermodular coloring theorem which is an extension of Schrijver’s supermodular coloring theorem.

**Theorem 5.1.6.** Skew-supermodular coloring theorem. [17] Let \( p_1, p_2 : 2^V \rightarrow \mathbb{Z} \cup \{\infty\} \) be two skew-supermodular functions and \( k \in \mathbb{Z}_+ \). Then \( V \) can be partitioned into classes \( L_1, L_2, \ldots, L_k \) such that
\[
p_i(U) \leq |\{j \in [k] : L_j \cap U \neq \emptyset\}| \quad (41)
\]
for each \( i = 1, 2 \) and each \( U \subseteq V \) if and only if
\[
p_i(U) \leq \min\{k, |U|\} \quad (42)
\]

However this theorem is not a direct generalization of supermodular coloring as intersecting supermodular functions are not necessarily skew-supermodular. For example the above defined function \( g_i \) is not skew-supermodular.

### 5.1.2 König’s theorem from skew-supermodular coloring

In this section we prove König’s edge-coloring theorem from Theorem 5.1.6. Let \( G := (V, E) \) be a bipartite graph such that \( V_1, V_2 \) are the two partitions. Let \( X \subseteq E \) and \( \mathcal{I}_i(X) := \{v \in V_i | \delta(v) \cap X \neq \emptyset\} \), i.e., \( \mathcal{I}_i(X) \) is the set of vertices in \( V_i \) that are incident on \( X \). Let \( d(v) \) be the degree of vertex \( v \) in \( G \). Let \( S \subseteq V_i \) and \( \partial(S) := \{v \in S | d(v) \leq d(u) \forall u \in S\} \), i.e., \( \partial(S) \) is the minimum degree vertex in \( S \). If there are multiple vertices with same minimum degree we arbitrarily choose one vertex. First let us show that the function \( f_i(X) = |X| - \sum_{v \in \mathcal{I}_i(X) \setminus \partial(I_i(X))} d(v) \) is skew-supermodular.
**Lemma 5.1.7.** $f_i(X) = |X| - \sum_{v \in \{I_i(X) \cap \delta(I_i(X))\}} d(v)$ is skew-supermodular for $i = 1, 2$.

**Proof.** Let us show that $f_1$ is skew-supermodular. Proof for skew-supermodularity of $f_2$ follows similarly. Let $X, Y \subseteq E$, if $X \cap Y = \emptyset$, condition (38) of skew-supermodular functions hold trivially. So, assume $X \cap Y \neq \emptyset$ and we will show that in this case condition (37) of skew-supermodular functions is true. Let $g(X) = |X|$ and $h(X) = \sum_{v \in \{I_1(X) \cap \delta(I_1(X))\}} d(v)$. Then, $f_1(X) = g(X) - h(X)$. Clearly $g(X)$ is modular, i.e., $g(X) + g(Y) = g(X \cup Y) + g(X \cap Y)$. So we need to show $h(X) + h(Y) \geq h(X \cup Y) + h(X \cap Y)$ if $X \cap Y \neq \emptyset$. W.l.o.g. assume $d(\delta(I_1(X))) \geq d(\delta(I_1(Y)))$.

Now,

$$h(X \cup Y) + h(X \cap Y)$$

$$\leq \sum_{v \in I_1(X \cup Y)} d(v) + \sum_{v \in I_1(X \cap Y)} d(v) - d(\delta(I_1(X \cup Y))) - d(\delta(I_1(X \cap Y)))$$

$$\leq \sum_{v \in I_1(X)} d(v) + \sum_{v \in I_1(Y)} d(v) - d(\delta(I_1(X \cup Y))) - d(\delta(I_1(X \cap Y))) \quad (43)$$

$$\leq \sum_{v \in I_1(X)} d(v) + \sum_{v \in I_1(Y)} d(v) - d(\delta(I_1(X))) - d(\delta(I_1(Y))) \quad (44)$$

$$\leq h(X) + h(Y) \quad (45)$$

This is what we needed to prove. Here inequality (43) follows from the submodularity property of cut function. Inequality (44) follows from the fact that $d(\delta(I_1(X \cap Y))) \geq d(\delta(I_1(X)))$ and $d(\delta(I_1(X \cup Y))) = \min\{d(\delta(I_1(X))), d(\delta(I_1(Y)))\} = d(\delta(I_1(Y))).$

$\square$

Now we are ready to prove the edge-coloring theorem.

**Proof of Theorem 5.2.2.** Let us define $p_i(X) := |X| - \sum_{v \in \{I_i(X) \cap \delta(I_i(X))\}} d(v)$ for $i = 1, 2$. From Lemma 5.1.7, $p_i$ is skew-submodular. Also from the definition it is clear that $p_i(U) = |U| - h(U) \leq |U|$ as $h(U) \geq 0$. Now let $X_v = \{e | e \in \delta(v) \cap X\}$
for any vertex \( v \in I(X) \). Note that
\[
|X_v| \leq d(v) \leq \Delta \tag{46}
\]

Then
\[
p_i(X) = |X| - \sum_{v \in \{I_i(X) \setminus \partial(I_i(X))\}} d(v)
\]
\[
= |X_{\partial(I_i(X))}| - \sum_{v \in \{I_i(X) \setminus \partial(I_i(X))\}} (d(v) - |X_v|)
\]
\[
\leq |X_{\partial(I_i(X))}|
\]
\[
\leq \Delta \quad \text{[From (46)]}
\]

Thus \( p_i(U) \leq \min \{\Delta, |U|\} \) for any \( U \subseteq E \). Therefore using Theorem 5.1.6, we get a proper edge coloring with \( \Delta \) colors.

It will be interesting if one can extend Theorem 5.1.6 to get a better bound for weighted bipartite edge coloring.

### 5.2 Edge-Coloring Weighted Bipartite Graphs

In this section we present our main result and prove Theorem 5.0.2.

**Theorem 5.2.1.** [Restatement of Theorem 5.0.2] There is a polynomial time algorithm for the weighted bipartite edge coloring problem which returns a proper weighted coloring using at most \( \lceil 2.2223m \rceil \) colors where \( m \) is denotes the maximum over all the vertices of the number of unit-sized bins needed to pack the weights of incident edges.

**The Algorithm:**

Our complete algorithm for edge-coloring weighted bipartite graphs is given in Figure 12. In the algorithm, we set a threshold \( \gamma = \frac{1}{10} \) and consider the subgraph induced by edges with weights more than \( \gamma \) and apply a combination of König’s Theorem and
1. $F \leftarrow \emptyset$, $t \leftarrow 2.2223$.

2. Include edges with weight $> \gamma = \frac{1}{10}$ in $F$ in nonincreasing order of weight maintaining the property that $\deg_F(v) \leq \lceil tn \rceil$ for all $v \in V$.

3. Decompose $F$ into $r = \lceil tn \rceil$ matchings $M_1, \ldots, M_r$ and color them using colors $1, \ldots, r$. Let $F_i \leftarrow M_i$ for each $1 \leq i \leq r$.

4. Add remaining edges with weight $> \gamma$ in nonincreasing order of weight to any of the $F_i$’s maintaining that weighted degree of each color at each vertex is at most one, i.e., $\sum_{e \in \delta(v) \cap F_i} w_e \leq 1$ for each $v \in V$ and $1 \leq i \leq r$.

5. Add remaining edges with weight $\leq \gamma$ in nonincreasing order of weight to any of the $F_i$’s maintaining that weighted degree of each color at each vertex is at most one, i.e., $\sum_{e \in \delta(v) \cap F_i} w_e \leq 1$ for each $v \in V$ and $1 \leq i \leq r$.

Figure 12: Algorithm for Edge Coloring Weighted Bipartite Graphs

A greedy algorithm with $\lceil tn \rceil$ colors where $t = 2.2223 > 20/9$. The remaining edges of weight at most $\gamma$ are then added greedily.

Analysis:

Now we prove Theorem 5.0.2. Though the algorithm is purely combinatorial, the analysis uses configuration LP and other techniques from bin packing to prove the correctness of the algorithm.

First we state the König’s Theorem since we use it as a subroutine in our algorithm to ensure a decomposition of $F$ into $\lceil tn \rceil$ matchings.

Theorem 5.2.2. [127] Given a bipartite graph $G = (V, E)$ there exists a coloring of edges with $\Delta = \max_{v \in V} \deg_E(v)$ colors such that all edges incident at a common vertex receive a distinct color. Moreover, such a coloring can be found in polynomial time.

The following lemma from Correa and Goemans [41] (it was also implicit in [54]) ensures that if the algorithm succeeds in coloring all edges of weight at least $\gamma$, the greedy coloring will be able to color the remaining edges of weight at most $\gamma$. 

123
Lemma 5.2.3. [41, 54] Consider a bipartite weighted graph $G = (V,E)$ with a coloring of all edges of weight at least $\gamma$ using at least $\frac{2m}{1-\gamma}$ colors for some $\gamma > 0$. Then the greedy coloring will succeed in coloring the edges with weight at most $\gamma$ without any additional colors.

In our setting, we have $\gamma = \frac{1}{10}$ and the number of colors is at least $\frac{20}{9}m = \frac{2m}{1-\frac{1}{10}}$ and thus Lemma 5.2.3 applies. Hence, it suffices to show the algorithm is able to color all edges with weights $> \frac{1}{10}$ using $\lceil tm \rceil$ colors as the remaining smaller edges can be colored greedily. Thus, without loss of generality, we assume that the graph has no edges of weight less than $\frac{1}{10}$ and prove the following lemma.

Lemma 5.2.4. If all edges have weight more than $\frac{1}{10}$ and $t = 2.2223$ ($> 20/9$) then the algorithm in Figure 12 returns a coloring of all edges using $\lceil tm \rceil$ colors such that the weighted degree of each color at each vertex is at most one, i.e., $\sum_{e \in \delta(v) \cap F_i} w_e \leq 1$.

Proof. Suppose for the sake of contradiction, the algorithm is not able to color all edges. Let $e := (u, v)$ be the first edge that cannot be colored by any color in Step (3) or Step (4) of the algorithm. Let the weight of edge $e$, $w_e$, be $\alpha$. Moreover, when $e$ is considered in Step (2), degree of either $u$ or $v$ is already $\lceil tm \rceil$ else we would have included $e$ in $F$. Without loss of generality let that vertex be $u$, i.e., $deg_F(u) = \lceil tm \rceil$.

For each color $1 \leq i \leq \lceil tm \rceil$, we must have that $\sum_{f \in \delta(u) \cap F_i} w_f > 1 - \alpha$ or $\sum_{f \in \delta(u) \cap F_i} w_f > 1 - \alpha$, else we can color $e$ in Step (4). Let $H_v = \{i | \sum_{f \in \delta(v) \cap F_i} w_f > 1 - \alpha\}, \beta m = |H_v|$. Now for each color $i \notin H_v$, we have $\sum_{f \in \delta(u) \cap F_i} w_f > 1 - \alpha$. Moreover, $deg_F(u) = \lceil tm \rceil$ and each of these edges weighs at least $w_e = \alpha$. Hence, for each color $1 \leq i \leq \lceil tm \rceil$, there is an edge incident at $u$ colored with color $i$ with weight at least $\alpha$. Let us call a color $i$ tight at $u$ if $\sum_{f \in \delta(u) \cap F_i} w_f > (1 - \alpha)$ and a color $i$ open at $u$ if $\sum_{f \in \delta(u) \cap F_i} w_f \in [\alpha, 1 - \alpha]$. Let $\tau$ be the number of tight colors at
$u$ and $\theta$ be the number of open colors at $u$. Thus we have,

\[
\tau \geq (t - \beta)m \quad (47)
\]

\[
\theta = (tm - \tau) \leq \beta m \quad (48)
\]

Now consider the following lemma.

**Lemma 5.2.5.** [68] Each edge of weight at least $1/t$ is in $F$.

It gives the following inequality.

\[
\alpha < 1/t \quad (49)
\]

Now consider all edges incident on $v$. We get,

\[
m > \beta m (1 - \alpha) \quad (50)
\]

Similarly considering all edges incident on $u$. We get,

\[
m > (tm - \beta)(1 - \alpha) + (\beta m)\alpha
\]

\[
\Rightarrow 1 > t(1 - \alpha) + \beta(2\alpha - 1) \quad (51)
\]

In fact we can strengthen inequality (49) to the following:

\[
\alpha \leq 1/3 \quad (52)
\]

This follows from the fact that a unit sized bin can contain at most two items with weight $> 1/3$. As all edges incident to a vertex can be packed into $m$ unit sized bins, there can be at most $2m$ edges incident to a vertex with weight $> 1/3$. Since $t > 2$, we get that all edges with weight more than $1/3$ must have been included in $F$ in Step (2). Thus $\alpha \leq 1/3$. Moreover we also get from (50):

\[
\beta < 1/(1 - \alpha) \leq 3/2 \quad (53)
\]
Now there are two cases:

**Case A:** $\alpha \leq 1/4$. Consider the RHS of (51): $t(1 - \alpha) + \beta(2\alpha - 1)$. If we show that the expression is always greater than 1 for $\alpha \leq 1/4$, then we arrive at a contradiction. Now,

$$t(1 - \alpha) + \beta(2\alpha - 1) - 1 > t(1 - \alpha) - \frac{(1 - 2\alpha)}{(1 - \alpha)} - 1, \text{[From (50)]}$$

$$\geq \frac{20(1 - \alpha)^2 - 9(1 - 2\alpha) - 9(1 - \alpha)}{9(1 - \alpha)}$$

$$\geq \frac{(20\alpha^2 - 13\alpha + 2)}{9(1 - \alpha)}$$

$$\geq \frac{(4\alpha - 1)(5\alpha - 2)}{9(1 - \alpha)}$$

$$\geq 0, \text{as } \alpha \leq 1/4 \quad (54)$$

Thus $t(1 - \alpha) + \beta(2\alpha - 1) > 1$, which contradicts (51).

**Case B:** $1/4 < \alpha \leq 1/3$. In this case, we will show in Lemma 5.2.6 that if $\beta \leq 13/9$, then all edges incident at $u$ can not be packed in $m$ bins. On the other hand, in Lemma 5.2.13 we show that if $\beta > 13/9$, then all edges incident at $v$ can not be packed in $m$ bins. This two facts together give us the desired contradiction.

**Lemma 5.2.6.** If $\beta \leq 13/9$, then edges incident at $u$ can not be packed in $m$ bins.

**Proof.** To give a lower bound on the number of bins required, we will consider a relaxation to the bin packing problem for edges incident at vertex $u$ and show that the optimal value of the relaxation, and thus the optimal number of bins required, is greater than $m$. The lower bound will be exhibited by constructing a feasible dual solution to the relaxation to the bin packing problem.

Since $deg_F(u) = \lfloor tm \rfloor$ when edge $e$ was considered in Step (2) of the algorithm and not included in $F$, we have that all edges incident at $u$ in $F$ have weight at least
the weight of $e$. Moreover, edges are considered in the decreasing order of weight in Step (4), the weight of all edges incident at $u$ when $e$ is considered in Step (4) is $\geq w_e$. We restrict our attention to these edges incident at $u$ with weight $\geq \alpha$ and show that they cannot be packed in $m$ unit sized bins. Let us divide these edges incident at $u$ into three size classes.

- Large $L := \{ f \in \delta(u) : w_f \in (1/2, 1] \}$.
- Medium $M := \{ f \in \delta(u) : w_f \in (1/3, 1/2] \}$.
- Small $S := \{ f \in \delta(u) : w_f \in [\alpha, 1/3] \}$.

First we have the following observation.

**Observation 5.2.7.** In any bin packing solution, in any bin there can be at most one item from $L$, two items from $L \cup M$ and three items from $L \cup M \cup S$.

Now let us prove the following two claims.

**Claim 5.2.8.** Edges in $L \cup M$ are included in Step (2) of the algorithm and thus are a subset of $F$.

**Proof.** If we are unable to add an edge $f$ in Step (2), it means one of its endpoints have $\lceil tm \rceil > 2m$ edges with weight $\geq w_f$. Since all edges incident at any vertex $v \in V$ can be packed in $m$ bins, there are at most $2m$ edges incident at it with weight more than $1/3$. Thus all edges of weight more than $1/3$ i.e., all edges in $L \cup M$ must be included in $F$ in Step (2) of the algorithm.

As a corollary we get,

**Claim 5.2.9.** For any color $i$, there is at most one edge in $L \cup M$ with color $i$.

**Proof.** As all edges in $L \cup M$ must be included in $F$ in Step (2) of the algorithm. In Step (3) of the algorithm, we include at most one edge of $F$ incident at any vertex in
each \( F_i \). Thus each color class obtains at most one edge incident at each vertex from \( F \) and therefore, from \( L \cup M \).

Using the observation and the above claim, we itemize the configuration of each of the tight colors depending on the size of edges with that color. Note that tight colors must have weight \( 1 - \alpha \geq 1 - 1/3 = 2/3 \).

1. If the tight color has a single edge \( f \). Then we have that \( w_f > \frac{2}{3} \) and only possibility is \( i)(L) \); Here by \( (L) \), we denote that the bin contains only one item and that item is an item from the set \( L \).

2. If the tight color contains exactly two edges. Here \( (S, S) \) is not tight as the total weight of edges in such a bin is \( \leq 2/3 \). So, the bin can contain at most one item from \( S \). On the other hand, the bin can contain at most one item from \( L \cup M \) from Claim 5.2.9. Thus the possible size types of these edges are \( ii)(L, S); iii)(M, S) \); As above, by \( (L, S) \) we denote that the bin contains only two items: exactly one item from set \( L \) and exactly one item from \( S \).

3. If the tight color contains three edges. The bin can contain at most one item from \( L \cup M \) from Claim 5.2.9. However if it contains one \( L \) item, sum of weights of three items exceeds one. Thus the only possible size types of these edges are \( iv)(M, S, S); v)(S, S, S) \).
Now consider the following LP: $LP_{bin}(u)$:

\[
\begin{align*}
\min & \quad \sum_{i=1}^{5} y_i \\
\text{s.t.} & \quad x_1 + x_2 + x_3 + x_4 + x_5 \geq \tau \quad (55) \\
& \quad y_1 + y_2 \geq x_1 + x_2 + z_1 \quad (56) \\
& \quad y_1 + 2y_3 + y_4 \geq x_3 + x_4 + z_2 \quad (57) \\
& \quad y_2 + y_3 + 2y_4 + 3y_5 \geq x_2 + x_3 + 2x_4 + 3x_5 + z_3 \quad (58) \\
& \quad z_1 + z_2 + z_3 \geq \theta \quad (59) \\
& \quad x_j, y_k, z_l \geq 0 \quad \forall j \in [5], k \in [5], l \in [3] \quad (60)
\end{align*}
\]

**Lemma 5.2.10.** The optimal number of unit sized bins needed to pack all edges incident at $u$ is at least the optimum value of $LP_{bin}(u)$.

**Proof.** Given a feasible packing of edges incident at $u$ in at most $m$ unit sized bins, we construct a feasible solution $(\bar{x}, \bar{y}, \bar{z})$ to the linear programming relaxation whose objective is at most the number of unit sized bins needed in the packing. In the feasible solution $(\bar{x}, \bar{y}, \bar{z})$, the variables $\bar{x}$ and $\bar{z}$ are constructed using the coloring given by the algorithm. The variables $\bar{y}$ are constructed using the optimal bin packing.

We first define the variables $\bar{x}$. Let $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5$ be the number of tight colors at $u$ of type $(L), (L,S), (M,S), (M,S,S), (S,S,S)$, respectively. Since the coloring of the edges incident at $u$ is one of the five types described above and there are at least $\tau$ tight colors, we have that $\sum_{i=1}^{5} \bar{x}_i \geq \tau$ and thus the solution satisfies constraint (55).

Now we define the variables $\bar{z}_1, \bar{z}_2, \bar{z}_3$ to be the number of items in open colors from $L, M$ and $S$ respectively. There are $\theta$ open colors. Each open color contains at least one item $L \cup M \cup S$. Thus, $\bar{z}_1 + \bar{z}_2 + \bar{z}_3 \geq \theta$ and thus the solution satisfies constraint (59).

To construct the solution $\bar{y}$, we will group the bins in the optimal bin packing solutions depending on the subset of items present in them into five classes and the
number of bins in each class will define the variables \( \bar{y} \). The constraints (56)-(58) will correspond to making sure that the optimal bin packing solution has appropriate number of items of each size type.

We now have the following claim where we characterize the possible bin configurations.

**Claim 5.2.11.** Consider any feasible bin-packing of edges incident at \( u \) restricted to edges in \( L \cup M \cup S \). Then each bin must contain items which correspond to a subset of one of the following 5 configurations or subsets of these configurations.

\[
\begin{align*}
C_1 &: (L, M) \\
C_2 &: (L, S) \\
C_3 &: (M, M, S) \\
C_4 &: (M, S, S) \\
C_5 &: (S, S, S)
\end{align*}
\]

**Proof.** Observe that in any bin there can be at most one item from \( L \), two items from \( L \cup M \) and three items from \( L \cup M \cup S \). Now let us consider two cases.

1. If the bin contain an item from \( L \). In this case, the bin can not contain three items as the sum of their weights exceeds one. So, it can contain at most one item from \( L \) and one item from \( M \cup S \). Thus \( C_1 \) and \( C_2 \) cover such two cases.

2. If the bin does not contain any item from \( L \). In this case, the bin can contain three items from \( M \cup S \) and at most two of these items can be from \( M \). Thus \( C_3, C_4 \) and \( C_5 \) cover such possibilities.

We map each configuration in the optimal bin packing solution to one of types \( C_i \) where the configuration is either \( C_i \) or its subset. Let \( \bar{y}_i \) denote the number of bins mapped to type \( C_i \) for each \( 1 \leq i \leq 5 \). We now count the number of items of each type to show feasibility of the constraints of the linear program.

**Constraint (56).** Items of type \( L \) equal \( \bar{x}_1 + \bar{x}_2 + \bar{z}_1 \) and can only be contained in configuration \( C_1 \) and \( C_2 \). Thus we have \( \bar{y}_1 + \bar{y}_2 \geq \bar{x}_1 + \bar{x}_2 + \bar{z}_1 \) satisfying constraint (56).
Constraint (57). Items of type $M$ equal $\bar{x}_3 + \bar{x}_4 + \bar{z}_2$ and are contained once in configurations $C_1, C_4$ and twice in configuration $C_3$. Thus we have $\bar{y}_1 + 2\bar{y}_3 + \bar{y}_4 \geq \bar{x}_3 + \bar{x}_4 + \bar{z}_2$ satisfying constraint (57).

Constraint (58). Items of type $S$ equal $\bar{x}_2 + \bar{x}_3 + 2\bar{x}_4 + 3\bar{x}_5 + \bar{z}_3$ and occur once in configurations $C_2, C_3$, twice in configurations $C_4$ and thrice in $C_5$. Thus, we have $\bar{y}_2 + \bar{y}_3 + 2\bar{y}_4 + 3\bar{y}_5 \geq \bar{x}_2 + \bar{x}_3 + 2\bar{x}_4 + 3\bar{x}_5 + \bar{z}_3$ showing feasibility of constraint (58).

This implies that $(\bar{x}, \bar{y}, \bar{z})$ is a feasible solution to $LP_{bin}$ and its objective equals the number of bins needed to pack the edges incident at $u$ in unit sized bins. Thus we have the lemma. □

We now show a contradiction by showing the optimal value of the $LP_{bin}(u)$ is more than $m$.

**Lemma 5.2.12.** The optimal solution to $LP_{bin}(u)$ is strictly more than $m$.

**Proof.** We prove this by considering the dual linear program of the $LP_{bin}(u)$. Since every feasible solution to the dual LP gives a lower bound on the objective of the primal $LP_{bin}(u)$, it is enough to exhibit a feasible dual solution of objective strictly more than $m$ to prove the lemma. Now the dual of the $LP_{bin}$ is following:

$$\begin{align*}
\text{max} & \quad \tau \cdot v_1 + \theta \cdot v_5 \\
\text{Subject to:} & \quad v_1 - v_2 \leq 0, \quad v_1 - v_2 - v_4 \leq 0, \\
& \quad v_1 - v_3 - v_4 \leq 0, \quad v_1 - v_3 - 2v_4 \leq 0, \\
& \quad v_1 - 3v_4 \leq 0, \quad v_2 + v_3 \leq 1, \\
& \quad v_2 + v_4 \leq 1, \quad 2v_3 + v_4 \leq 1, \\
& \quad v_3 + 2v_4 \leq 1, \quad 3v_4 \leq 1, \\
& \quad v_5 - v_2 \leq 0, \quad v_5 - v_3 \leq 0, \\
& \quad v_5 - v_4 \leq 0, \quad v_5 - v_5 \leq 0, \\
& \quad v_5 - v_1 \leq 0, \quad v_i \geq 0 \quad \forall i \in [4]
\end{align*}$$

A feasible dual solution is: $v_1 = \frac{2}{3}, v_2 = \frac{2}{3}, v_3 = \frac{1}{3}, v_4 = \frac{1}{3}, v_5 = \frac{1}{3}$.
Thus dual optima $\geq \frac{2\tau}{3} + \frac{\theta}{3}$ and we need at least these many colors to color items in $\tau$ tight colors and $\theta$ open colors. Using the fact that $\theta = tm - \tau, \tau \geq (t - \beta)m$ and $t > \frac{20}{9}m, \beta \leq \frac{13}{9}$, we obtain that the number bins required to pack all items incident on $u$ is:

$$\geq \tau \cdot v_1 + \theta \cdot v_4$$

$$\geq \frac{2}{3} \tau + \frac{1}{3}(tm - \tau)$$

$$= \frac{1}{3} \tau + \frac{1}{3}(tm)$$

$$\geq \frac{1}{3} (t - \beta)m + \frac{1}{3}(tm)$$

$$\geq \frac{2t}{3}m - \frac{\beta}{3}m$$

$$> m\left(\frac{2}{3} \cdot \frac{20}{9} - \frac{1}{3} \cdot \frac{13}{9}\right)$$

$$= m$$

Thus the number bins required to pack all items incident on $u$ is strictly greater than $m$. This is a contradiction. \qed

This concludes the proof of Lemma 5.2.6. \qed

**Lemma 5.2.13.** If $\beta > 13/9$, then edges incident at $v$ can not be packed in $m$ bins.

**Proof.** Similar to the previous lemma, to give a lower bound on the number of bins required, we will consider a relaxation to the bin packing problem for edges incident at vertex $v$ and show that the optimal value of the relaxation, and thus the optimal number of bins required, is greater than $m$. Again, the lower bound will be exhibited by constructing a feasible dual solution to the relaxation to the bin packing problem.

As $\beta(1 - \alpha) < 1$, we get,

$$\alpha > 1 - 1/\beta \geq 4/13 > 0.3. \quad (61)$$

Let us call a color $i$ tight at $v$ if $\sum_{f \in \delta(v) \cap E_i} w_f > (1 - \alpha)$. Now consider any tight color $B$ at $v$. At most one edge $f$ in $B$ was colored in Step (3) of the algorithm and
remaining edges (if any) in $B$ were colored in Step (4) of the algorithm. Now, $w_f$ can be smaller than $w_e$ as it might be the case that when $e$ was considered in Step (2) then already degree of other endpoint $u$ was $\lceil tm \rceil$. However, edges are considered in the nonincreasing order of weight in Step (4), thus the weight of all edges incident at $v$ when $e$ is considered in Step (4) is also $\geq w_e$. Thus, all the remaining edges (if any) in $B$ that were colored in Step (4) of the algorithm have weights $> \alpha$.

We restrict our attention to the edges at tight colors at $v$ and show that if $\beta > \frac{13}{9}$ they cannot be packed in $m$ unit sized bins. Let us divide these edges incident at $u$ into three size classes.

- **Large** $L := \{ f \in \delta(v) : w_f \in (1/2, 1] \}$.
- **Medium** $M := \{ f \in \delta(v) : w_f \in (1/3, 1/2] \}$.
- **Small** $S := \{ f \in \delta(v) : w_f \in [\alpha, 1/3] \}$.
- **Tiny** $T := \{ f \in \delta(v) : w_f \in (1/10, \alpha) \}$.

First we have the following observation.

**Observation 5.2.14.** In any bin packing solution, in any bin there can be at most one item from $L$, two items from $L \cup M$, three items from $L \cup M \cup S$ and nine items from $L \cup M \cup S \cup T$.

Now let us prove the following claim.

**Claim 5.2.15.** For any tight color $i$ at $v$, all edges added in Step (4) of the algorithm are in $S$. As a corollary, there is at most one edge incident on $v$ with color $i$ that is in $L \cup M \cup T$ and it can only be added in Step (2) of the algorithm.

**Proof.** From Claim 5.2.8, it follows that all edges in $L \cup M$ must be included in $F$ in Step (2) of the algorithm. On the other hand, all edges colored in Step (4) have
weights $\geq \alpha$, they can not be in $T$. Hence, only edges in $S$ are colored in Step (4). Edges in $L \cup M \cup T$ are colored in Step (3). In Step (3) of the algorithm, we include at most one edge of $F$ incident at any vertex in each $F_i$. Thus each color class obtains at most one edge incident at each vertex from $F$ and therefore, from $L \cup M \cup T$. \qed

Using the observation and the claim, we itemize the configuration of each of the tight colors depending on the size of edges with that color. Note that in this case tight colors have weights $> 1 - \alpha \geq 1 - 1/3 = 2/3$.

1. If the tight color has a single edge $f$. Then we have that $w_f > 2/3$ and only possibility is $i)(L)$; Here by $(L)$, we denote that the bin contains only one item and that item is an item from the set $L$.

2. If the tight color contains exactly two edges. From Claim 5.2.15, the bin can contain at most one item from $L \cup M \cup T$. On the other hand, $(S, S)$ or $(T, S)$ has weight $\leq 2/3$. So the bin can contain at most one item from $S$ and one item from $L \cup M$. Thus the possible size types of these edges are $ii)(L, S); iii)(M, S)$; As above, by $(L, S)$ we denote that the bin contains only two items: exactly one item from set $L$ and exactly one item from $S$.

3. If the tight color contains three edges. From Claim 5.2.15, the bin can contain at most one item from $L \cup M \cup T$. However if the bin contains an item from $L$, the sum of weights of an item from $L$ and two items from $S$ exceeds one. Thus the possible size types of these edges are $iv)(M, S, S); v)(S, S, S); vi)(T, S, S)$.\[134]
Now consider the following configuration LP based on the items at $v$: $LP_{bin}(v)$:

\[
\min \sum_{i=1}^{19} y_i \\
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq \beta m \quad (62) \\
y_{14} \geq x_1 \quad (63) \\
y_8 + y_9 + y_{10} + y_{15} \geq x_2 \quad (64) \\
y_3 + y_7 + y_9 + y_{11} + 2y_{12} + y_{16} \geq x_3 \quad (65) \\
y_2 + 2y_4 + y_6 + y_{10} + y_{11} + 2y_{13} + y_{17} \geq x_4 \quad (66) \\
3y_1 + 2y_2 + 2y_3 + y_4 + 2y_5 + y_6 + y_7 + y_8 + y_{18} \geq x_2 + x_3 + 2x_4 + 3x_5 + 2x_6 \quad (67) \\
(3y_5 + 3y_6 + 3y_7 + y_8 + y_9 + y_{10} + 3y_{11} + 3y_{12} + 3y_{13} + 3y_{14} + 4y_{15} + 6y_{16} + 6y_{17} + 6y_{18} + 9y_{19}) \geq x_6 \quad (68) \\
y_j, x_k \geq 0 \quad \forall j \in [19], k \in [6] \quad (69)
\]

**Lemma 5.2.16.** The optimal number of unit sized bins needed to pack all edges incident at $u$ is at least the optimum value of $LP_{bin}(v)$.

**Proof.** Given a feasible packing of edges incident at $v$ in at most $m$ unit sized bins, we construct a feasible solution $(\bar{x}, \bar{y})$ to the linear programming relaxation whose objective is at most the number of unit sized bins needed in the packing. In the feasible solution $(\bar{x}, \bar{y})$, the variables $\bar{x}$ are constructed using the coloring given by the algorithm. The variables $\bar{y}$ are constructed using the optimal bin packing.

We first define the variables $\bar{x}$. Let $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5, \bar{x}_6$ be the number of tight colors at $v$ of type $(L), (L,S), (M,S), (M,S,S), (S,S,S), (T,S,S)$, respectively. Since the coloring of the edges incident at $v$ is one of the six types described above and there are at least $\beta m$ tight colors at $v$, we have that $\sum_{i=1}^{6} \bar{x}_i \geq \beta m$ and thus the solution satisfies constraint (74).

To construct the solution $\bar{y}$, we will group the bins in the optimal bin packing solutions depending on the subset of items present in them into nineteen classes and
the number of bins in each class will define the variables $\bar{y}$. The constraints (75)-(80) will correspond to making sure that the optimal bin packing solution has appropriate number of items of each size type.

To define the 19 different classes of bin types in the optimal solution, we need to further classify items according to size. Let $L_1, L_2 \subseteq L$ be the set of large edges that appear in the configurations of the type $(L)$ and $(L, S)$, respectively, in the tight colors.

As for any item $l_1 \in L_1$, $w_{l_1} + \alpha > 1$. We get,

$$w_{l_1} > 1 - \alpha \geq 1 - 1/3 = 2/3. \quad (70)$$

Let $M_1, M_2$ be the set of medium edges that appear in the tight colors of type $(M, S)$ and $(M, S, S)$, respectively.

We now have the following claim where we characterize the possible bin configurations. We show that each bin contains items which correspond to one of 19 possible configurations or their subsets.

**Claim 5.2.17.** Consider any feasible bin-packing of edges incident at $v$ restricted to edges in $L \cup M \cup S \cup T$. Then each bin must contain items which correspond to a subset of one of the following 19 configurations.

- $C_1 : (S, S, S)$
- $C_2 : (M_2, S, S)$
- $C_3 : (M_1, S, S)$
- $C_4 : (M_2, M_2, S)$
- $C_5 : (S, S, T, T, T)$
- $C_6 : (M_2, S, T, T, T)$
- $C_7 : (M_1, S, T, T, T)$
- $C_8 : (L_2, S, T)$
- $C_9 : (L_2, M_1, T)$
- $C_{10} : (L_2, M_2, T)$
- $C_{11} : (M_1, M_2, T, T, T)$
- $C_{12} : (M_1, M_1, T, T, T)$
- $C_{13} : (M_2, M_2, T, T, T)$
- $C_{14} : (L_1, T, T, T)$
- $C_{15} : (L_2, T, T, T)$
- $C_{16} : (M_1, T, T, T, T, T)$
- $C_{17} : (M_2, T, T, T, T, T)$
- $C_{18} : (S, T, T, T, T, T)$
- $C_{19} : (T, T, T, T, T, T)$

**Proof.** Observe that since item in $L$ have weight more than $1/2$, items in $M$ have weight more than $1/3$, items in $S$ have weight more than $1/4$ and items in $T$ have weight more than $1/10$, there can be at most one item from $L$, two items items from $L \cup M$, at most
three items in total from $L \cup M \cup S$ and at most nine items from $L \cup M \cup S \cup T$ in any feasible packing.

1. Bins with three items from $D := L \cup M \cup S$. As the sum of weights of three elements from $D$ is more than $3\alpha > 0.9$, elements from $T$ can not appear in these bins as $3\alpha + w_f > 1$ for any $f \in T$. Moreover, configurations which contain at least one item of $L$ cannot have three items from $D$ without the weight exceeding one. Thus, the packing can contain only items from $M$ and $S$. When the bin contains only $S$ items, it corresponds to configuration $C_1$. Packings which contain one item from $M_1 \cup M_2$ and two items from $S$ are exactly the configurations $C_2, C_3$.

Now let us consider the case when we have two items from $M_1 \cup M_2$. First observe that for each $h \in M_1$, there exists a $s \in S$ such that $(h,s)$ are the only edges colored with a tight color. Thus we have that $w_h + w_s > 1 - \alpha$. But then for any other $h' \in M_1 \cup M_2$ and $s' \in S$, we have that

$$w_h + w_{h'} + w_{s'} \geq w_h + w_s + \alpha > 1$$

(71)

where the inequality follows since $w_{h'} \geq w_s$ and $w_{s'} \geq \alpha$. This implies that configurations of type $(M_1, M_1, S)$ or $(M_1, M_2, S)$ are not feasible. Hence, the only possible remaining configuration is $C_4$.

2. Bins with two items from $D$. Here we consider maximal configurations which are not subsets of configurations which contain three items from $D$. When the configuration contains two items $s_1, s_2 \in S$, we have that $w_{s_1} + w_{s_2} > 0.6$ and thus the only maximal configuration is $C_5$. Configurations $C_6, C_7$ cover the case when the configuration contains one item from $S$ and one item from $M$. Let $l \in L_1$. Since $l$ appears alone in a tight color, we have that $w_l + \alpha > 1$. Since every item in $M \cup S$ has weight at least $\alpha$, there is no valid configuration with two items from $D$ such that one of them is in $L_1$. If the configuration contains
$l_2 \in L_2$ and $g \in M_1 \cup M_2 \cup S$, it can at most contain one element $t \in T$ as $w_{l_2} + w_g + w_t > 0.9$. Thus configurations $C_8$ cover the case when there is one item from $S$ and one item from $L$. Now we are left with cases when there are no $S$ items in the bin. If there is one $L$ item and one $M$ item, $C_9, C_{10}$ cover such possibilities.

Similarly if configuration contains two items from $M$, it can contain at most 3 elements from $T$. Configurations $C_{11}, C_{12}, C_{13}$ cover all such the possibilities.

3. Bins with one item from $D$. Here we consider configurations which are not subsets of configurations which contain at least two items from $D$. Note that as for any item $l_1 \in L_1$, from inequality (70), $w_{l_1} > 2/3$. Thus $(L_1, T, T, T)$ is the maximal configuration containing one $L_1$ item. $C_{14}$ is the corresponding configuration. The other four possible configurations are $C_{15}, C_{16}, C_{17}, C_{18}$ where the bins contain one item from $L_2, M_1, M_2, S$ respectively. In these case the number of $T$ items are upper bounded by 4, 6, 6, 6 respectively from the lower bound of size of items in the corresponding classes in $D$.

4. Bins with no item from $D$. Only possible maximal configuration is $C_{19}$.

We map each configuration in the optimal bin packing solution to one of types $C_i$ where the configuration is either $C_i$ or its subset. Let $\bar{y}_i$ denote the number of bins mapped to type $C_i$ for each $1 \leq i \leq 19$. We now count the number of items of each type to show feasibility of the constraints of the linear program.

Constraint (75). Items of type $L_1$ equal $\bar{x}_1$ and can only be contained in configuration $C_{14}$. Thus we have $\bar{y}_{14} \geq \bar{x}_1$.

Constraint (76). Similarly, items of type $L_2$ equal $\bar{x}_2$. They are contained in configurations $C_8, C_9, C_{10}$ and $C_{15}$. Thus we have $\bar{y}_8 + \bar{y}_9 + \bar{y}_{10} + \bar{y}_{15} \geq \bar{x}_2$. 

138
Constraint (77). Items of type $M_1$ equal $\bar{x}_3$ and are contained once in configurations $C_3, C_7, C_9, C_{11}, C_{16}$ and twice in configuration $C_{12}$. Thus we have $\bar{y}_3 + \bar{y}_7 + \bar{y}_9 + \bar{y}_{11} + 2\bar{y}_{12} + \bar{y}_{16} \geq \bar{x}_3$.

Constraint (78). Items of type $M_2$ equal $\bar{x}_4$ and are contained once in configurations $C_2, C_6, C_{10}, C_{11}, C_{17}$ and twice in configurations $C_4, C_{13}$. Thus we have $\bar{y}_2 + 2\bar{y}_4 + \bar{y}_6 + \bar{y}_{10} + \bar{y}_{11} + 2\bar{y}_{13} + \bar{y}_{17} \geq \bar{x}_4$ satisfying constraint (78).

Constraint (79). Items of type $S$ equal $\bar{x}_2 + 3\bar{x}_5 + 2\bar{x}_6$ and occur once in configurations $C_4, C_6, C_7, C_8, C_{18}$, twice in configurations $C_2, C_3, C_5$ and thrice in $C_1$. Thus, we have $3\bar{y}_1 + 2\bar{y}_2 + 2\bar{y}_3 + \bar{y}_4 + 2\bar{y}_5 + \bar{y}_6 + \bar{y}_7 + \bar{y}_8 + \bar{y}_{18} \geq \bar{x}_2 + \bar{x}_3 + 2\bar{x}_4 + 3\bar{x}_5 + 2\bar{x}_6$.

Constraint (80). Items of type $T$ equal $\bar{x}_6$ and occur once in configurations $C_8, C_9, C_{10}$, thrice in configurations $C_5, C_6, C_7, C_{11}, C_{12}, C_{13}, C_{14}$, four times in configuration $C_{15}$, six times in configurations $C_{16}, C_{17}, C_{18}$ and nine times in configurations $C_{19}$. Thus, we have $(3\bar{y}_5 + 3\bar{y}_6 + 3\bar{y}_7 + \bar{y}_8 + \bar{y}_9 + 3\bar{y}_{11} + 3\bar{y}_{12} + 3\bar{y}_{13} + 3\bar{y}_{14} + 4\bar{y}_{15} + 6\bar{y}_{16} + 6\bar{y}_{17} + 6\bar{y}_{18} + 9\bar{y}_{19}) \geq \bar{x}_6$.

This implies that $(\bar{x}, \bar{y})$ is a feasible solution to $LP_{\text{bin}}(v)$ and its objective equals the number of bins needed to pack the edges incident at $v$ in unit sized bins. Thus we have the lemma.

We now show a contradiction by showing the optimal value of the $LP_{\text{bin}}(v)$ is more than $m$.

**Lemma 5.2.18.** The optimal solution to the $LP_{\text{bin}}(v)$ is strictly more than $m$.

**Proof.** We prove this by considering the dual linear program of the $LP_{\text{bin}}(v)$. Since every feasible solution to the dual LP gives a lower bound on the objective of the primal $LP_{\text{bin}}(v)$, it is enough to exhibit a feasible dual solution of objective strictly more than $m$ to prove the lemma. Now the dual of the $LP_{\text{bin}}(v)$ is given below:
\[ \max \beta m \cdot v_1 \]

Subject to:
\[ v_1 - v_2 \leq 0, \quad v_1 - v_3 - v_6 \leq 0, \]
\[ v_1 - v_4 - v_6 \leq 0, \quad v_1 - v_5 - 2v_6 \leq 0, \]
\[ v_1 - 3v_6 \leq 0, \quad v_1 - 2v_6 - v_7 \leq 0, \]
\[ 3v_6 \leq 1, \quad v_5 + 2v_6 \leq 1, \]
\[ v_4 + 2v_6 \leq 1, \quad 2v_5 + v_6 \leq 1, \]
\[ 2v_6 + 3v_7 \leq 1, \quad v_5 + v_6 + 3v_7 \leq 1, \]
\[ v_4 + v_6 + 3v_7 \leq 1, \quad v_3 + v_6 + v_7 \leq 1, \]
\[ v_3 + v_4 + v_7 \leq 1, \quad v_3 + v_5 + v_7 \leq 1, \]
\[ v_4 + v_5 + 3v_7 \leq 1, \quad 2v_4 + 3v_7 \leq 1, \]
\[ 2v_5 + 3v_7 \leq 1, \quad v_2 + 3v_7 \leq 1, \]
\[ v_3 + 4v_7 \leq 1, \quad v_4 + 6v_7 \leq 1, \]
\[ v_5 + 6v_7 \leq 1, \quad v_6 + 6v_7 \leq 1, \]
\[ 9v_7 \leq 1, \quad v_i \geq 0 \quad \forall i \in [7] \]

A feasible dual solution is
\[ v_1 = \frac{9}{13}, v_2 = \frac{9}{13}, v_3 = \frac{7}{13}, v_4 = \frac{5}{13}, v_5 = \frac{1}{13}, v_6 = \frac{4}{13}, v_7 = \frac{1}{13}. \] (72)

Thus dual optima is at least
\[ \beta m \cdot \frac{9}{13} > m \]

Thus, we need more than \( m \) bins to pack all items incident at \( v \), a contradiction. \( \square \)

This completes the proof of Lemma 5.2.13. \( \square \)

Therefore, the proof of Theorem 5.0.2 is complete. \( \square \)

5.2.1 Better approximation when all edge weights are \( > 1/4 \)

If we assume all items have weights \( > 1/4 \), then similar analysis will attain \( 2.2m \) colors are sufficient.

**Theorem 5.2.19.** There is a polynomial time algorithm for the weighted bipartite edge coloring problem which returns a proper weighted coloring using at most \( \lceil 2.2m \rceil \) colors where \( m \) is denotes the maximum over all the vertices of the number of unit-sized bins needed to pack the weights of incident edges.
Proof. In this section we will show that Algorithm 12 with \( t > \frac{11}{5} \) is sufficient to get a proper weighted coloring when all edges are \( > 1/4 \). Similar to Section 5.2, we will show in Lemma 5.2.20 that if \( \beta \leq 7/5 \), then all edges incident at \( u \) can not be packed in \( m \) bins. On the other hand, in Lemma 5.2.21 we show that if \( \beta > 7/5 \), then all edges incident at \( v \) can not be packed in \( m \) bins. This two facts together give us the desired contradiction.

Lemma 5.2.20. If \( \beta \leq 7/5 \), then edges incident at \( u \) can not be packed in \( m \) bins.

Proof. From the analysis of Lemma 5.2.12, we get that number of bins to pack all items incident on \( u \) is :

\[
\geq \frac{2t}{3} m - \frac{\beta}{3} m
\]

\[
> \frac{2}{3} \cdot \frac{11}{5} m - \frac{1}{3} \cdot \frac{7}{5} m \quad [\because t > \frac{11}{5}]
\]

\[= m \quad (73)\]

This is a contradiction. \( \Box \)

Lemma 5.2.21. If \( \beta > 7/5 \), then edges incident at \( v \) can not be packed in \( m \) bins.

Proof. As in Lemma 5.2.13 We restrict our attention to the edges at tight colors at \( v \) and show that if \( \beta > \frac{7}{5} \) they cannot be packed in \( m \) unit sized bins. Only difference is now we have edges that are \( > 1/4 \). Let us divide these edges incident at \( u \) into three size classes.

- Large \( L := \{f \in \delta(v) : w_f \in (1/2, 1]\} \).
- Medium \( M := \{f \in \delta(v) : w_f \in (1/3, 1/2]\} \).
- Small \( S := \{f \in \delta(v) : w_f \in [\alpha, 1/3]\} \).
- Tiny \( T' := \{f \in \delta(v) : w_f \in (1/4, \alpha\} \).

141
Claim 5.2.22. Consider any feasible bin-packing of edges incident at \( v \) restricted to edges in \( L \cup M \cup S \cup T \). Then each bin must contain items which correspond to a subset of one of the following 19 configurations.

\[
\begin{align*}
C_1 &: (S, S, S) & C_2 &: (M_2, S, S) & C_3 &: (M_1, S, S) \\
C_4 &: (M_2, M_2, S) & C_5 &: (S, S, T') & C_6 &: (M_2, S, T') \\
C_7 &: (M_1, S, T') & C_8 &: (L_2, S) & C_9 &: (L_2, M_1) \\
C_{10} &: (L_2, M_2) & C_{11} &: (M_1, M_2, T') & C_{12} &: (M_1, M_1, T') \\
C_{13} &: (M_2, M_2, T') & C_{14} &: (L_1, T') & C_{15} &: (L_2, T') \\
C_{16} &: (M_1, T', T') & C_{17} &: (M_2, T', T') & C_{18} &: (S, T', T') \\
C_{19} &: (T', T', T')
\end{align*}
\]

Proof. In Claim 5.2.17, we considered maximal configurations in tight colors. Let us consider any configuration in Claim 5.2.17 and let us assume that in a tight color belonging to that configuration there are \( k \) items from class \( T \) and we can not add any more item. Then \((\sum_{i \in L \cup M \cup S} w_i + (k + 1) \cdot \frac{1}{10} + \epsilon) > 0\) for any \( \epsilon > 0 \). Hence, \((\sum_{i \in L \cup M \cup S} w_i + (k + 1) \cdot \frac{2}{5} \cdot \frac{1}{4} + \epsilon) > 0\) for any \( \epsilon > 0 \). Hence if there were \( k \) items of class \( T \) in configurations in Claim 5.2.17, then we can replace items in \( T \) with at most \([ (k + 1) \cdot \frac{2}{5} ] - 1 \) items from class \( T' \). Thus we replace 1, 2, 3, 4, 6, 9 items in \( T \) in the configurations in Claim 5.2.17 with items 0, 1, 1, 2, 3 in \( T' \) respectively. This gives us the configurations in this claim from the configurations in Claim 5.2.17. □

Thus we get the following LP similar to the configuration LP \( LP_{\text{bin}}(v) \) in the proof of Lemma 5.2.13.
\begin{equation}
\min \sum_{i=1}^{19} y_i \geq \beta m
\end{equation}
\begin{equation}
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq x_1
\end{equation}
\begin{equation}
y_14 \geq x_1
\end{equation}
\begin{equation}
y_8 + y_9 + y_{10} + y_{15} \geq x_2
\end{equation}
\begin{equation}
y_3 + y_7 + y_9 + y_{11} + 2y_{12} + y_{16} \geq x_3
\end{equation}
\begin{equation}
y_2 + 2y_4 + y_6 + y_{10} + y_{11} + 2y_{13} + y_{17} \geq x_4
\end{equation}
\begin{equation}
3y_1 + 2y_2 + 2y_3 + y_4 + 2y_5 + y_6 + y_7 + y_8 + y_{18} \geq x_2 + x_3 + 2x_4 + 3x_5 + 2x_6
\end{equation}
\begin{equation}
(y_5 + y_6 + y_7 + y_{11} + y_{12} + y_{13} + y_{14} + y_{15} + 2y_{16} + 2y_{17} + 2y_{18} + 3y_{19}) \geq x_6
\end{equation}
\begin{equation}
y_{j}, x_{k} \geq 0 \forall j \in [19], k \in [6]
\end{equation}

Note that all constraints are same as in the proof of Lemma 5.2.13 except for the last constraint as now we count the number of items in $T'$ instead of $T$. We also get the following dual which differs only in the constraints corresponding to configurations containing type $T$ items.

\begin{equation}
\max \beta m \cdot v_1
\end{equation}
\begin{equation}
\text{Subject to:}
\end{equation}
\begin{align*}
v_1 - v_2 &\leq 0, & v_1 - v_3 - v_6 &\leq 0, \\
v_1 - v_4 - v_6 &\leq 0, & v_1 - v_5 - 2v_6 &\leq 0, \\
v_1 - 3v_6 &\leq 0, & v_1 - 2v_6 - v_7 &\leq 0, \\
3v_6 &\leq 1, & v_5 + 2v_6 &\leq 1, \\
v_4 + 2v_6 &\leq 1, & 2v_5 + v_6 &\leq 1, \\
2v_6 + v_7 &\leq 1, & v_5 + v_6 + v_7 &\leq 1, \\
v_4 + v_6 + v_7 &\leq 1, & v_3 + v_6 &\leq 1, \\
v_3 + v_4 &\leq 1, & v_3 + v_5 &\leq 1, \\
v_4 + v_5 + v_7 &\leq 1, & 2v_4 + v_7 &\leq 1, \\
2v_5 + v_7 &\leq 1, & v_2 + v_7 &\leq 1, \\
v_3 + v_7 &\leq 1, & v_4 + 2v_7 &\leq 1, \\
v_5 + 2v_7 &\leq 1, & v_6 + 2v_7 &\leq 1, \\
3v_7 &\leq 1, & v_i &\geq 0 \forall i \in [7]
\end{align*}
Here a feasible dual solution is

\[ v_1 = \frac{5}{7}, v_2 = \frac{5}{7}, v_3 = \frac{4}{7}, v_4 = \frac{3}{7}, v_5 = \frac{1}{7}, v_6 = \frac{2}{7}, v_7 = \frac{1}{7}. \] (82)

Thus dual optima is at least

\[ \beta m \cdot \frac{7}{5} > m \]

This completes the proof of Lemma 5.2.21.

Therefore, the proof of Theorem 5.2.19 is complete.

5.3 Conclusion

In this chapter we have shown that \( \lceil 2.2223m \rceil \) colors are sufficient to get a polynomial time proper weighted coloring for bipartite graphs. Our techniques based on configuration LP and other structural properties of bin packing, might be useful in the analysis of other algorithms related to Clos networks. We remark that considering the case \( 1/4 \geq \alpha > 1/5 \) separately, might improve the bound further by more case analysis and the fact that at most \( 4m \) items are there. However we can at most attain \( 35m/16 \approx 2.19m \) by that analysis.

There is no known better existential result for even with exponential running time. Finding a better approximation algorithm (independent of \( m \)) or inapproximability, and extending our techniques to general graphs are other interesting directions.
Chapter VI

CONCLUSION

In this thesis we obtained improved approximation for three classical generalization of bin packing: geometric bin packing, vector bin packing and weighted bipartite edge coloring. We have made related concluding remarks in the corresponding chapters. Now in this chapter we conclude by listing some of the open problems related to multidimensional bin packing for future work.

Problem 1. Tight approximability of bin packing.
The present best algorithm for 1-D BP by Rothvoss [161, 97] uses $\text{Opt} + O(\log \text{Opt})$ bins. Proving one could compute a packing with only a constant number of extra bins will be a remarkable progress and is mentioned as one of the ten most important problems in approximation algorithms [186]. Consider the seemingly simple 3-Partition case in which all $n$ items have size $s_i \in (1/4, 1/2)$. Recent progress by [149] suggests that either the $\log n$ bound is best possible for 3-Partition or some fundamentally new ideas are needed to make progress.

Problem 2. Integrality gap of Gilmore-Gomory LP.
It has been conjectured in [164] that the Gilmore-Gomory LP for 1-D BP has Modified Integer Roundup Property i.e., $\text{Opt} \leq \lceil \text{Opt}_f \rceil + 1$. The conjecture has been proved true for the case when the instance contains at most 7 different item sizes [167]. Settling the status for the general case is an important open problem in optimization.

Problem 3. Tight asymptotic competitive ratio for 1-D online BP.
The present best algorithm for online bin packing is by Seiden [168] and has asymptotic performance ratio is at most 1.58889. Ramanan et al. [159] showed that Harmonic-type algorithms can not achieve better than 1.58333 asymptotic competitive ratio. In general the best known lower bound for asymptotic competitive ratio is 1.54014 [182]. Giving a stronger lower bound using some other construction is an important question in online algorithms.

**Problem 4. Improved inapproximability for multidimensional bin packing.**
There is a huge gap between the best approximation guarantee and hardness of multidimensional bin packing. There are no explicit inapproximability bounds known for multidimensional bin packing as function of $d$ apart from the APX-hardness in 2-D. Thus there is a huge gap between the best algorithm ($1.69^d-1$, i.e., exponential in $d$ for geometric packing and $O(\ln d)$ for vector packing for $d > 2$) and the hardness. Improved inapproximability as a function of $d$ will be an interesting hardness result.

**Problem 5. Tight ratio between optimal Guillotine packing and optimal bin packing.** Improving the present guarantee for 2-D GBP will require an algorithm that is not input-agnostic. In particular, this implies that it should have the property that it can round two identical items (i.e., with identical height and width) differently. One such candidate is the guillotine packing approach [14]. It has been conjectured that this approach can give an approximation ratio of $4/3$ for 2-D GBP. At present the best known upper bound on this gap is $T_\infty \approx 1.69$ [22].

**Problem 6. Tight ratio between optimal two-stage packing and optimal bin packing.** Caprara conjectured [21] that there is a two-stage packing that gives $3/2$ approximation for 2-D bin packing. As there are PTAS for 2-stage packing [22], this will give another $3/2$ approximation for 2-D BP and coupled with our R&A
method this will give another $1.405 + \epsilon$ approximation. Presently the upper bound between best two-stage packing and optimal bin packing is $T_\infty \approx 1.69$. As 2-stage packings are very well-studied, this question is of independent interest and it might give us more insight on the power of Guillotine packing.

**Problem 7. Extending R&A framework to d-D GBP and 3-D SP.**

One key bottleneck to extend R&A framework to d-D GBP or other related problems, is to find a good approximation algorithm to find the solution of the configuration LP. A $\text{poly}(d)$ asymptotic approximation for the LP will give us a $\text{poly}(d)$ asymptotic approximation for $d$-D GBP, a significant improvement over current best ratio of $2^{O(d)}$ for $d > 2$.

**Problem 8. Resolving Chung-Ross Conjecture [32].**

The first conjecture says that given an instance of the weighted bipartite edge coloring problem, there is a proper weighted coloring using at most $2m - 1$ colors where $m$ denotes the maximum over all the vertices of the number of unit-sized bins needed to pack the weights of edges incident at the vertex.

A stronger version of the conjecture is that given an instance of the weighted bipartite edge coloring problem, there is a proper weighted coloring using at most $2n - 1$ colors where $n$ is the smallest integer greater than the maximum over all the vertices of the total weight of edges incident at the vertex.

Finally, finding faster algorithms that work well in practice, is also a very important problem.
REFERENCES


[85] Grigoriadis, M. D., Khachiyan, L. G., Porkolab, L., and Villavicencio, J., “Approximate max-min resource sharing for structured concave


[161] Rothvoss, T., “Approximating bin packing within $o(\log \text{opt} \ast \log \log \text{opt})$ bins,” in FOCS, pp. 20–29, 2013.


Arindam Khan obtained his bachelors degree in Computer Science and Engineering at Indian Institute of Technology, Kharagpur, India and is currently a Phd candidate in the Algorithms, Combinatorics and Optimization program at Georgia Institute of Technology, Atlanta. His research spans several areas of theoretical computer science including approximation algorithms and inapproximability results for NP-hard optimization problems, online algorithms, algorithmic game theory and learning theory. Previously he has been a research intern at Theory group, Microsoft Research Redmond and Microsoft Research Silicon Valley and a blue scholar at Analytics and Optimization group, IBM Research, India. He welcomes your comments to akhan67@gatech.edu.
Approximation Algorithms for Multidimensional Bin Packing

Arindam Khan

169 Pages

Directed by Professor Prasad Tetali

The bin packing problem has been the corner stone of approximation algorithms and has been extensively studied starting from the early seventies. In the classical bin packing problem, we are given a list of real numbers in the range \((0, 1]\), the goal is to place them in a minimum number of bins so that no bin holds numbers summing to more than 1. In this thesis we study approximation algorithms for three generalizations of bin packing: geometric bin packing, vector bin packing and weighted bipartite edge coloring.

In two-dimensional (2-D) geometric bin packing, we are given a collection of rectangular items to be packed into a minimum number of unit size square bins. Geometric packing has vast applications in cutting stock, vehicle loading, pallet packing, memory allocation and several other logistics and robotics related problems. We consider the widely studied orthogonal packing case, where the items must be placed in the bin such that their sides are parallel to the sides of the bin. Here two variants are usually studied, (i) where the items cannot be rotated, and (ii) they can be rotated by 90 degrees. We give a polynomial time algorithm with an asymptotic approximation ratio of \(\ln(1.5) + 1 \approx 1.405\) for the versions with and without rotations. We have also shown the limitations of rounding based algorithms, ubiquitous in bin packing algorithms. We have shown that any algorithm that rounds at least one side of each large item to some number in a constant size collection values chosen independent of the problem instance, cannot achieve an asymptotic approximation ratio better than \(3/2\).

In \(d\)-dimensional vector bin packing (VBP), each item is a \(d\)-dimensional vector that need to be packed into unit vector bins. The problem is of great significance
in resource constrained scheduling and appeared also recently in virtual machine placement in cloud computing. Even in two dimensions, it has novel applications in layout design, logistics, loading and scheduling problems. We give a polynomial time algorithm with an asymptotic approximation ratio of $\ln(1.5) + 1 \approx 1.405$ for 2-D VBP. For $d$ dimensions we give a polynomial time algorithm with an asymptotic approximation ratio of $\ln(d/2) + 1.5 \approx \ln d + 0.81$. We also consider vector bin packing under resource augmentation. We give a polynomial time algorithm that packs vectors in $(1 + \epsilon) \text{Opt}$ bins when we allow augmentation in $(d - 1)$ dimensions and $\text{Opt}$ is the minimum number of bins needed to pack the vector into $(1, 1)$ bins.

In weighted bipartite edge coloring problem, we are given an edge-weighted bipartite graph $G = (V, E)$ with weights $w : E \rightarrow [0, 1]$. The task is to find a proper weighted coloring of the edges with as few colors as possible. An edge coloring of the weighted graph is called a proper weighted coloring if the sum of the weights of the edges incident to a vertex of any color is at most one. This problem is motivated by rearrangeability of 3-stage Clos networks which is very useful in various applications in interconnected networks and routing. We show a polynomial time approximation algorithm that returns a proper weighted coloring with at most $\lceil 2.2223 m \rceil$ colors where $m$ is the minimum number of unit sized bins needed to pack the weight of all edges incident at any vertex. We also show that if all edge weights are $> 1/4$ then $\lceil 2.2 m \rceil$ colors are sufficient.