Geometry Meets Vectors: Approximation Algorithms for Multidimensional Packing

Arindam Khan
Department of Computer Science and Automation,
Indian Institute of Science, Bengaluru, India

Eklavya Sharma
Department of Industrial & Enterprise Systems Engineering,
University of Illinois at Urbana-Champaign, IL, USA

K. V. N. Sreenivas
Department of Computer Science and Automation,
Indian Institute of Science, Bengaluru, India

Abstract

We study the generalized multidimensional bin packing problem (GVBP) that generalizes both geometric packing and vector packing. Here, we are given n rectangular items where the i^{th} item has width $w(i)$, height $h(i)$, and d nonnegative weights $v_1(i), v_2(i), \ldots, v_d(i)$. Our goal is to get an axis-parallel non-overlapping packing of the items into square bins so that for all $j \in [d]$, the sum of the j^{th} weight of items in each bin is at most 1. This is a natural problem arising in logistics, resource allocation, and scheduling. Despite being well-studied in practice, approximation algorithms for this problem have rarely been explored.

We first obtain two simple algorithms for GVBP having asymptotic approximation ratios $6(d+1)$ and $3(1+\ln(d+1)+\varepsilon)$. We then extend the Round-and-Approx (R&A) framework [3, 6] to wider classes of algorithms, and show how it can be adapted to GVBP. Using more sophisticated techniques, we obtain better approximation algorithms for GVBP, and we get further improvement by combining them with the R&A framework. This gives us an asymptotic approximation ratio of $2(1+\ln((d+4)/2))+\varepsilon$ for GVBP, which improves to $2.919+\varepsilon$ for the special case of $d=1$. We obtain further improvement when the items are allowed to be rotated. We also present algorithms for a generalization of GVBP where the items are high dimensional cuboids.

2012 ACM Subject Classification Theory of computation → Packing and covering problems

Keywords and phrases Bin packing, rectangle packing, multidimensional packing, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2022.23

Related Version *arXiv Version*: `https://arxiv.org/abs/2106.13951`

Funding *Arindam Khan*: Research partly supported by Pratiksha Trust Young Investigator Award, Google India Research Award, and Google ExploreCS Award.

Acknowledgements We thank Nikhil Bansal, Thomas Rothvoss, and anonymous reviewers for their helpful comments.

1 Introduction

Bin packing and knapsack problems are classical NP-hard optimization problems. Two classical generalizations of these problems: geometric packing and vector packing have been well-studied from the 1980s [14, 17]. Geometric packing considers the packing of rectangular items, whereas, in vector packing items are multidimensional vectors. However, often in practice, we encounter a mixture of geometric and vector constraints. Consider the following airlines cargo problem [42]: We have boxes to load in an airline cargo container. In addition to the geometric constraint that all the boxes must fit within the container, we also have a constraint that the total weight of the loaded boxes is within a specified capacity. Thus, in this problem, three dimensions are geometric and the weight is a vector constraint.

Weight has been an important constraint to consider for packing in logistics and supply chain management, e.g., cranes and other equipment can be damaged by the bins being too heavy [1]. When different cargoes are packed into a fleet of aircraft for transport, one needs the individual cargoes to be not too heavy to ensure stability and less fuel consumption [2]. Similar problems find applications in vehicle routing with loading constraints [8]. Many practical heuristics [48, 50] have been proposed for such problems. Many companies (such as Driw, Boxify, Freightcom) and practical packages [53] have considered the problem. Often, we also want to limit other attributes, like the amount of magnetism, radioactivity, or toxicity. Each such property can be considered an additional vector dimension.

Such multidimensional packing problems also get attention due to their connections with fair resource allocation [43]. In recent years, a considerable amount of research has focused on group fairness [31, 51] such that the algorithms are not biased towards (or against) some groups or categories. One such notion of fairness is *restricted dominance* [7], which upper bounds the number (or size) of items from a category. These different categories can be considered as dimensions. E.g., in a container packing problem for flood relief, one needs to ensure that the money spent on a container is fairly distributed among different types of items (such as medicine, food, garments). Hence, for each category, there is an upper bound on the value that can go into a container.

Formally, we are given n items $I := \{1, 2, \ldots, n\}$ that are (d_g, d_v)-dimensional, i.e., item i is a d_g-dimensional cuboid of lengths $\ell_1(i), \ell_2(i), \ldots, \ell_{d_g}(i)$ and has d_v non-negative weights $v_1(i), v_2(i), \ldots, v_{d_v}(i)$. A (d_g, d_v)-dimensional bin is a d_g-dimensional cuboid of length 1 in each geometric dimension and weight capacity 1 in each of the d_v vector dimensions. A feasible packing of items into a bin is a packing where items are packed parallel to the axes without overlapping, and for all $j \in [d_v]$, the sum of the j^{th} vector dimension of the items in the bin is at most 1 (see Definition 14 in Appendix A for a more formal definition of (d_g, d_v) packing). In the (d_g, d_v) bin packing problem (BP), we have to feasibly pack all items into the minimum number of bins. In the (d_g, d_v) knapsack problem (KS), each item i also has an associated nonnegative profit $p(i)$, and we have to feasibly pack a maximum-profit subset of the items into a single bin (also called "knapsack"). (d_g, d_v) packing problems generalize both d_g-dimensional geometric packing (when $d_v = 0$) and d_v-dimensional vector packing (when $d_g = 0$). Already for vector bin packing, if d_v is part of the input, there is an approximation hardness of $d_v^{1-\varepsilon}$, unless NP=ZPP [5]. Thus, throughout the paper we assume both d_g and d_v to be constants.

1.1 Our Results

We study the first approximation algorithms for general (d_g, d_v) BP, with a focus on $d_g = 2$. We give two simple algorithms for $(2, d)$ BP, called `simplePack` and `betterSimplePack`, having asymptotic approximation ratios (AARs) of $6(d + 1)$ and $3(1 + \ln(d + 1)) + \varepsilon$, respectively, for any $\varepsilon > 0$. Getting an AAR better than $O(\log d)$ for d-D VBP is NP-hard [46], so `betterSimplePack`'s AAR as a function of d is tight up to a constant factor. For $d = 1$, `betterSimplePack`'s AAR improves to $\approx 4.216 + \varepsilon$.

Next, we modify the Round-and-Approx (R&A) framework [6] so that it works for (d_g, d_v) BP. We combine R&A with the `simplePack` algorithm to get an AAR of $2(1+\ln(3(d+1)))+\varepsilon$ for $(2, d)$ BP. This improves upon the AAR of `betterSimplePack` for $d \geq 3$.

In Section 5, we obtain a more sophisticated algorithm for $(2, d)$ BP, called `cbPack`, that fits into the R&A framework and has an even better AAR. Table 1 lists the AARs of all our algorithms for $(2, d)$.

Table 1 Asymptotic approximation ratios of our algorithms for $(2, d)$ BP.

Algorithm	AAR for $(2, d)$ BP	AAR for $(2, 1)$ BP
`simplePack`	$6(d+1)$	12
`betterSimplePack`	$3(1+\ln(d+1))+\varepsilon$	$3(1+\ln(\frac{3}{2}))+\varepsilon \approx 4.216+\varepsilon$
`simplePack` with R&A	$2(1+\ln(3(d+1)))+\varepsilon$	$2(1+\ln 6)+\varepsilon \approx 5.5835+\varepsilon$
`cbPack` with R&A (without rotation)	$2(1+\ln(\frac{d+4}{2}))+\varepsilon$	$2(1+\ln(\frac{19}{12}))+\varepsilon \approx 2.919+\varepsilon$
`cbPack` with R&A (with rotation)	$2(1+\ln(\frac{d+3}{2}))+\varepsilon$	$2(1+\ln(\frac{3}{2}))+\varepsilon \approx 2.811+\varepsilon$

We also show how to extend `simplePack` and `betterSimplePack` to (d_g, d_v) BP to obtain AARs $2b(d_v+1)$ and $b(1+\ln(d_v+1)+\varepsilon)$, respectively, where $b := 9$ when $d_g = 3$, and $b := 4^{d_g} + 2^{d_g}$ when $d_g > 3$. We also give a similar algorithm for (d_g, d_v) KS, having an approximation ratio $b(1+\varepsilon)$.

1.2 Related Work

The bin packing problem (BP) has been the cornerstone of approximation algorithms [29]. The standard performance measure for BP algorithms is the asymptotic approximation ratio (AAR). An asymptotic polynomial time approximation scheme (APTAS) for BP was given by Fernandez de la Vega and Lueker [17], using linear grouping. Note that 1-D BP can be considered as (1, 0) BP as well as (0, 1) BP. The present best approximation algorithm for 1-D BP returns a packing in at most $\mathrm{opt}+O(\log \mathrm{opt})$ bins, where opt is the optimal number of bins [25]. Knapsack problem (KS) is one of Karp's 21 NP-complete problems. Lawler gave an FPTAS [40] for KS. For surveys on BP and KS, see [13, 33].

In [17], a $(d+\varepsilon)$-asymptotic approximation algorithm was given for the d-dimensional vector bin packing (d-D VBP). Chekuri and Khanna gave a $\ln d + O(1)$ approximation for d-D VBP [11]. The study of 2-D geometric bin packing (2-D GBP) was initiated by [14]. Caprara gave a T_∞^{d-1}-asymptotic approximation Harmonic-Decreasing-Height (HDH) algorithm for d-D GBP [9], where $T_\infty \approx 1.6901$. This is still the best known approximation for d-D GBP for $d \geq 3$. Both 2-D GBP and 2-D VBP do not admit an APTAS [4, 52, 45]. For large d, getting better than $O(\log d)$ asymptotic approximation for d-D VBP is NP-hard [46].

Bansal, Caprara, and Sviridenko [3] introduced the *Round-and-Approx (R&A)* framework to obtain improved approximations for both 2-D GBP and d-D VBP. The R&A framework is a two stage process. First, a (possibly exponential-sized) set covering LP relaxation (called configuration LP) is solved approximately. Then, a randomized rounding procedure is applied for a few steps to pack a subset of items, after which only a "small" fraction of items (called the *residual instance*) remain unpacked. In the second step, the residual instance is packed using a *subset-oblivious* algorithm. Intuitively, given a *random* subset S of I where each element occurs with probability about $1/k$, a ρ-approximate subset-oblivious algorithm produces a packing of S in approximately $\rho\,\mathrm{opt}(I)/k$ bins. In the R&A framework, one can obtain a $(1+\ln\rho)$-approximation algorithm using a ρ-approximate subset oblivious algorithm. Two algorithms, 1-D BP APTAS [17] and HDH [9] were shown to be subset-oblivious based on various properties of dual-weighting functions. This led to an AAR of $(1+\ln(1.69))$ and $(1+\ln d)$ for 2-D GBP and d-D VBP, respectively. However, it was cumbersome to extend subset-obliviousness to wider classes of algorithms.

Bansal and Khan [6] later extended the R&A framework for 2-D GBP to *rounding-based* algorithms, where the large dimensions are rounded up to $O(1)$ values and the packing of items is container-based, i.e., each bin contains a constant number of rectangular regions

called containers and items are packed into containers. For 2-D GBP, they used an algorithm with an AAR of 1.5 [27, 44] to obtain the present best AAR of $(1 + \ln 1.5) \approx 1.405$. For d-D VBP, Bansal et al. [5] used the R&A framework combined with a multi-objective budgeted matching problem, to obtain the present best AAR of $(0.81 + o_d(1) + \ln d)$.

Multidimensional knapsack is also well-studied. For d-D vector knapsack (d-D VKS), Frieze and Clarke gave a PTAS [18]. For 2-D geometric knapsack (GKS), Jansen and Zhang [28] gave a $(2 + \varepsilon)$-approximation algorithm, while the present best approximation ratio is $\frac{17}{9} + \varepsilon$ [20]. It is not even known whether 2-D GKS is APX-hard or not. There are many other related important geometric packing problems, such as strip packing [21, 19] and maximum independent set of rectangles [36, 22, 23]. For surveys on multidimensional packing, see [12, 35].

1.3 Technical Contribution

One of our main contributions is the enhancement of R&A framework [6] to wider applications.

First, R&A framework now also works with (d_g, d_v)-dimensional items, unifying the approach for geometric and vector packing. To use R&A, we need to solve the configuration LP of the corresponding bin packing problem. All previous applications (d-D VBP and 2-D GBP) of R&A solved the configuration LP within $(1+\varepsilon)$ factor using a $(1+O(\varepsilon))$-approximate solution to (a variant of) KS. Due to the unavailability of a PTAS for (a variant of) $(2, d)$ KS, we had to use a different linear programming algorithm [47] that uses an η-approximation algorithm for KS to $(1 + \varepsilon)\eta$-approximately solve the configuration LP of the corresponding BP problem, for any constants $1 < \eta$, $0 < \varepsilon < 1$.

Second, we introduce more freedom in choosing the packing structure. Unlike the R&A framework in [6] that worked only for container-based packing, we allow either relaxing the packing structure to non-container-based (like in `simplePack`) or imposing packing constraints in addition to being container-based (like in `cbPack`). This generalization can help in obtaining improved algorithms for other problems related to bin packing.

Finally, we allow rounding items in ways other than rounding up, if we can find a suitable way of *unrounding* a packing of rounded items. In `cbPack`, we round down the width and height of some items to 0, and in `simplePack`, we round each $(2, d)$-dimensional item i to an item of width 1, height x and each vector dimension x, where x is a value depending on the original dimensions of i. As shown in [35], if the large coordinates of items are rounded up to $O(1)$ types, we cannot get an AAR better than d and 4/3 for d-D VBP and 2-D GBP, respectively. However, as we now allow rounding down, the R&A framework may now work with algorithms having better AARs.

We also fix a minor error in the R&A framework of [35]. See Appendix C.1 for details.

In [3], it was mentioned: "One obstacle against the use of R&A for other problems is the difficulty in deriving subset-oblivious algorithms (or proving that existing algorithms are subset oblivious)." We expect that our progress will help in understanding the power of R&A to extend it to other set-cover type problems, e.g. round-SAP [32] and round-UFP [16, 32].

Our another major contribution is handling of the $(2, d)$ BP problem. This problem presents additional challenges over pure geometric BP, and our algorithm `cbPack` demonstrates how to circumvent them. For example, in geometric packing, items of low total area can be packed into a small number of bins using the NFDH algorithm [14]. This need not be true when items have weights, since the geometric dimensions can be small but the vector dimensions may be large. To handle this, we divide the items into different classes based on density (i.e., weight/area). We use the facts that items of low density and low total area can be packed into a small number of bins, and items of high density that fit into a bin have low

total area. Also, in geometric packing, we can sometimes move items with similar geometric dimensions across different bins (like in *linear grouping* [17, 44]). Vector dimensions again create problems here. To handle this, we only move items of similar geometric dimensions and density. This leads to a more *structured* packing and we show how to find such a near-optimal structured packing efficiently. Due to space limitations, we defer the description and analysis of `cbPack` to Section 5 and the full version's Appendix F [38].

2 Preliminaries and Notation

Let $\mathcal{I}$ be the set of all valid inputs to a minimization problem $\mathcal{P}$. For any input $I \in \mathcal{I}$, let $\text{opt}(I)$ be the cost of the optimal solution and $|\mathcal{A}(I)|$ be the cost of algorithm $\mathcal{A}$'s output on I. Define the *approximation ratio* $\rho_{\mathcal{A}}$ and the *asymptotic approximation ratio* (AAR) $\rho^{\infty}_{\mathcal{A}}$ as $\rho_{\mathcal{A}} := \sup_{I\in\mathcal{I}} \{|\mathcal{A}(I)|/\text{opt}(I)\}$, and $\rho^{\infty}_{\mathcal{A}} := \limsup_{z\to\infty} \sup_{I\in\mathcal{I}} \left\{ |\mathcal{A}(I)|/\text{opt}(I) \,\middle|\, \text{opt}(I) = z \right\}$, respectively. Intuitively, AAR is $\mathcal{A}$'s performance for inputs with large opt.

Let $[n] := \{1, 2, \ldots, n\}$. Let $\text{poly}(n)$ be the set of polynomial and sub-polynomial functions of n. Define $v_{\max}$, vol, and span as follows: $v_{\max}(i) := \max_{j=1}^{d_v} v_j(i)$, $\text{vol}(i) := \prod_{j=1}^{d_g} \ell_j(i)$, $\text{span}(i) := \max(\text{vol}(i), v_{\max}(i))$. $\text{span}(i)$ is, intuitively, the measure of *largeness* of item $i \in [n]$. For convenience, let $v_0(i) := \text{vol}(i)$. Assume w.l.o.g. that $\text{vol}(i) = 0$ implies $(\forall j \in [d_g], \ell_j(i) = 0)$. For a set I of items, given a function $f : I \mapsto \mathbb{R}$, for $S \subseteq I$, define $f(S) := \sum_{i\in S} f(i)$. This means, e.g., $\text{vol}(S) := \sum_{i\in S} \text{vol}(i)$. For any bin packing algorithm $\mathcal{A}$, let $\mathcal{A}(I)$ be the resulting bin packing of items I, and let $|\mathcal{A}(I)|$ be the number of bins in $\mathcal{A}(I)$. Define $\text{opt}(I)$ as the minimum number of bins needed to pack I. The following lemma relates span with opt.

▶ **Lemma 1.** *For (d_g, d_v) items I, $\lceil \text{span}(I) \rceil \le (d_v + 1)\,\text{opt}(I)$.*

Proof. Let $m = \text{opt}(I)$. In an optimal packing, let J_j be the items in the j^{th} bin. Then

$$\lceil \text{span}(I) \rceil = \left\lceil \sum_{k=1}^{m} \sum_{i\in J_k} \max_{j=0}^{d_v} v_j(i) \right\rceil \le \left\lceil \sum_{k=1}^{m} \sum_{j=0}^{d_v} v_j(J_k) \right\rceil \le (d_v + 1)m. \qquad \blacktriangleleft$$

For $d_g = 2$, let $w(i) := \ell_1(i)$, $h(i) := \ell_2(i)$ be the width and height of item i, respectively. The area of item i is $a(i) := w(i)h(i) = \text{vol}(i)$. The items in (2, 0) BP are called "rectangles".

2.1 Configuration LP

For a (d_g, d_v) bin packing instance I containing n items, a configuration of I is a packing of a subset of items of I into a bin. Let $\mathcal{C}$ be the set of all configurations of I. The *configuration matrix* of I is a matrix $A \in \{0, 1\}^{n\times|\mathcal{C}|}$ where $A[i, C]$ is 1 if configuration C contains item i and 0 otherwise. To solve the bin packing problem, it is sufficient to decide the number of bins of each configuration. This gives us the following linear programming relaxation, called a configuration LP:

$$\min_{x\in\mathbb{R}^{|\mathcal{C}|}} \sum_{C\in\mathcal{C}} x_C \quad \text{where} \quad Ax \ge \mathbf{1} \text{ and } x \ge 0.$$

Even though $|\mathcal{C}|$ can be exponential in n, any feasible solution x to the configuration LP may have a polynomial-sized representation if the size of $\text{support}(x)$ is polynomially bounded in n.

3 Simple Algorithms

In this section, we look at simple algorithms for (2, d) BP. They are based on the following simple corollary of Steinberg's algorithm [49].

▶ **Lemma 2** (Section 3 in [28]). *Let I be a set of rectangles where $a(I) \le 1$. Then I can be packed into 3 bins in $O(n \log^2 n / \log\log n)$ time.*

Let I be a (2, d) BP instance. Let $\widehat{I} := \{\text{span}(i) : i \in I\}$, i.e., $\widehat{I}$ is a 1-D BP instance. The algorithm `simplePack`(I) first runs the Next-Fit algorithm [30] on $\widehat{I}$. Let $[\widehat{J}_1, \widehat{J}_2, \ldots, \widehat{J}_m]$ be the resulting bin packing of $\widehat{I}$ into m bins. For each $\widehat{J}_j \subseteq \widehat{I}$, let J_j be the corresponding items from I. Then $\forall k \in [d_v]$, $v_k(J_j) \le 1$ and $\text{vol}(J_j) \le 1$. `simplePack` then uses the algorithm of Lemma 2 to pack each J_j into at most 3 bins, giving a packing of I into at most $3m$ bins. By the property of Next-Fit [30], we get that $m \le \lceil 2\,\text{size}(\widehat{I})\rceil = \lceil 2\,\text{span}(I)\rceil$. By Lemma 1, we get $3\lceil 2\,\text{span}(I)\rceil \le 6(d+1)\,\text{opt}(I)$. This gives us the following theorem.

▶ **Theorem 3.** *For (2, d) BP, `simplePack` uses at most $3\lceil 2\,\text{span}(I)\rceil$ bins, so it is a $6(d+1)$-approximation algorithm. It runs in $O(nd + n\log^2 n/\log\log n)$ time.*

The algorithm `betterSimplePack` first computes $\widetilde{I}$, which is a $(d+1)$-D VBP instance obtained by replacing the geometric dimensions of each item $i \in I$ by a single vector dimension $a(i)$. It computes a bin packing of $\widetilde{I}$ using any algorithm $\mathcal{A}$. It then uses the algorithm of Lemma 2 to pack I into at most $3|\mathcal{A}(\widetilde{I})|$ bins.

Note that $\text{opt}(\widetilde{I}) \le \text{opt}(I)$. If $\mathcal{A}$ has AAR α, then $|\mathcal{A}(\widetilde{I})| \le \alpha\,\text{opt}(\widetilde{I}) + O(1)$. Therefore, `betterSimplePack` has AAR 3α. The $(d+1)$-D VBP algorithm by [3] (parametrized by a constant $\varepsilon > 0$) gives $\alpha = 1 + \ln(d+1) + \varepsilon$ and the algorithm by [5] gives $\alpha = 1.5 + \ln((d+2)/2) + \varepsilon$ (improves to $\alpha = 1 + \ln(1.5) + \varepsilon$ for $d = 1$).

Similarly, we can get a $3(1+\varepsilon)$-approximation algorithm for (2, d) KS (see Appendix D of the full version [38]).

Although `simplePack`'s AAR is worse than `betterSimplePack`, `simplePack`'s output is upper-bounded in terms of span, which is a useful property. Hence, we will use it as a subroutine in other algorithms (like `cbPack`).

The algorithms for (2, d) packing can be extended to (d_g, d_v) packing. We just need an algorithm for the following problem: *given a set J of d_g-dimensional cuboids where $\text{vol}(J) \le 1$, pack J into a small number of bins.* We used Lemma 2 when $d_g = 2$. When $d_g = 3$, we can use the algorithm of Diedrich et al. (Section 2 of [15]) to pack J into at most 9 bins. For $d_g > 3$, we can pack J into at most $4^{d_g} + 2^{d_g}$ bins using a variant of the HDH_4 algorithm [10] (see Appendix C of the full version [38]). Hence, `simplePack` will use $b\lceil 2\,\text{span}(I)\rceil$ bins, where $b := 3$ when $d_g = 2$, $b := 9$ when $d_g = 3$, and $b := 4^{d_g} + 2^{d_g}$ when $d_g > 3$. Therefore, `simplePack` is $2b(d_v+1)$-approximate. Similarly, `betterSimplePack` has AAR $b(1 + \ln(d_v+1) + \varepsilon)$, and we can get a $b(1+\varepsilon)$-approximation algorithm for (d_g, d_v) KS.

4 Round-and-Approx Framework

We enhance the R&A framework as a general outline for designing approximation algorithms for bin packing and its variants. We denote the algorithm for the R&A framework as `rnaPack`(I, β, ε), which takes as input a set I of (d_g, d_v)-dimensional items and parameters $\beta \ge 1$ and $\varepsilon \in (0, 1)$. The steps of the algorithm are as follows. (See Algorithm 1 in Appendix C for a more formal description).

1. **Solve the Configuration LP** of I: Let $\widehat{x}$ be a μ-asymptotic-approximate solution to the configuration LP. Note that each index of $\widehat{x}$ corresponds to a configuration. In all previous applications of R&A, $\mu = 1 + \varepsilon$, but in our work, μ can be a large constant.
2. **Randomized rounding of configuration LP**: For $T := \lceil (\ln \beta) \|\widehat{x}\|_1 \rceil$ steps do the following: select a configuration C with probability $\widehat{x}_C / \|\widehat{x}\|_1$. Pack T bins according to each of these selected T configurations. Let S be the remaining items which are not packed, called the *residual instance*.
3. **Rounding of items**: We define a subroutine `round` that takes items I and parameter ε as input[1]. It *discards* a set $D \subseteq I$ of items such that $\text{span}(D) \leq \varepsilon \, \text{span}(I)$ and then *modifies* each item in $I - D$ to get a set $\widetilde{I}$ of items. We say that the output of $\texttt{round}(I, \varepsilon)$ is $(\widetilde{I}, D)$, where items in $\widetilde{I}$ are called *rounded items*. Intuitively, after rounding, the items in $\widetilde{I}$ are of $O(1)$ types, which makes packing easier. Also, since $\text{span}(D)$ is small, $D \cap S$ can be packed into a small number of bins using `simplePack`.

   We impose some restrictions on `round`, which we denote as conditions C1 and C2, which we describe in Section 4.2. Previous versions of R&A only allowed modifications where items' dimensions were rounded up. We don't have this restriction; we also allow rounding down some dimensions. We also allow `round` to output a poly(n)-sized list of guesses for $(\widetilde{I}, D)$.
4. **Pack rounded items**: Let $\widetilde{S}$ be the rounded items corresponding to $S \setminus D$. Pack $\widetilde{S}$ into bins using any bin packing algorithm that satisfies "condition C3", which we describe in Section 4.3. Let us name this algorithm `complexPack`.
5. **Unrounding**: Given a bin packing of $\widetilde{S}$, let `unround` be a subroutine that computes a bin packing of $S \setminus D$. `unround` is trivial in previous versions of R&A, because they only increase dimensions of items during rounding. In our applications, we may round down items, so `unround` can be non-trivial. `unround` can be any algorithm that satisfies "condition C4", which we describe in Section 4.3.

We can think of the R&A framework as a *meta-algorithm*, i.e., we give it the algorithms `round`, `complexPack` and `unround` as inputs and it outputs the algorithm `rnaPack`. The R&A framework requires that `round`, `complexPack` and `unround` satisfy four conditions C1, C2, C3, C4, which we describe in Sections 4.2 and 4.3. Prospective users of the R&A framework need to design these three subroutines and prove that they satisfy these four conditions.

Intuitively, `rnaPack` first packs some items into T bins by randomized rounding of $\widehat{x}$. We can prove that $\Pr(i \in S) \leq 1/\beta$ (using standard techniques from randomized rounding, see Lemma 16 in Appendix C), so S contains a small fraction of the items in I. We will then try to prove that if the rest of the algorithm (`round` + `complexPack` + `unround`) packs I into m bins, then it will pack S into roughly m/β bins. This notion was referred to in [3] as *subset-obliviousness*. We will use subset-obliviousness to bound the AAR of `rnaPack`. Section 4.5 shows how to break `simplePack` into `round`, `complexPack`, and `unround` and use it with R&A.

[1] The input to `round` is I instead of S because S is random and we want to round items deterministically, i.e., the rounding of each item $i \in S$ should not depend on which other items from I lie in S. In fact, this is where the old R&A framework [35] introduced an error. See Appendix C.1 for details.

4.1 Fractional Structured Packing

Let $(\widetilde{I}, D)$ be an output of `round`(I) and let $\widetilde{X}$ be an arbitrary subset of $\widetilde{I}$. Our analysis of `rnaPack` is based around a concept called *fractional structured packing* of $\widetilde{X}$. Note that the notion of fractional structured packing only appears in the analysis of `rnaPack`. It is not needed to describe any algorithm.

First, we discuss the notion of structured packing. Several types of packings are used in bin packing algorithms, such as *container-based* [27], *shelf-based* [14, 10], *guillotine-based* [24], *corridor-based* [20], *$\mathcal{N}$-box & $\mathcal{V}$-box* based [26], etc. A type of packing is called a *structured packing* if it satisfies *downward closure*, i.e., *a structured packing remains structured even after removing some items from the packed bins.* For example, Jansen and Prädel [27] showed that given any packing of a 2-D GBP instance into m bins, we can slice some of the items and repack them into $(1.5 + \varepsilon)m + O(1)$ bins such that the resulting packing is *container-based*. *Container-based* roughly means that in each bin, items are packed into rectangular regions called containers, and containers' heights and widths belong to a fixed set of $O(1)$ values. Hence, *container-based* is an example of a structured packing as a container-based packing remains container-based even after removing some items from the packed bins. Also note that the set of all possible packings is trivially a structured packing. Our R&A framework gives algorithm designers the freedom to use or define structured packing in any way they want, as long as they satisfy downward closure. Typically, the choice of the definition of structured packing will depend on the ease of proving Conditions C2 and C3 for that definition. This helped us go beyond Bansal and Khan's R&A framework [6], which only considered container-based packings.

Intuitively, a fractional structured packing is one where we slice each item of $\widetilde{X}$ into pieces and then find a structured packing of the pieces. Let $\text{fsopt}(\widetilde{X})$ be the number of bins in the optimal fractional structured packing of $\widetilde{X}$. To analyze the AAR of `rnaPack`, we will bound `complexPack`(S) in terms of $\text{fsopt}(\widetilde{S})$, and then bound $\text{fsopt}(\widetilde{S})$ in terms of $\text{opt}(I)$.

To define fractional structured packing, we first define what it means to slice an item. From a geometric perspective, slicing an item perpendicular to the k^{th} dimension means cutting the item into 2 parts using a hyperplane perpendicular to the k^{th} axis. The vector dimensions get split proportionately across the slices. E.g., for $d_g = 2$, if $k = 1$ for item i, then we slice i using a vertical cut, and if $k = 2$, we slice i using a horizontal cut.

▶ **Definition 4** (Slicing an item). *Let i be a (d_g, d_v)-dimensional item. Slicing i perpendicular to geometric dimension k with proportionality α (where $0 < \alpha < 1$) is the operation of replacing i by two items i_1 and i_2 such that: (i) $\forall j \neq k, \ell_j(i) = \ell_j(i_1) = \ell_j(i_2)$, (ii) $\ell_k(i_1) = \alpha\ell_k(i)$ and $\ell_k(i_2) = (1-\alpha)\ell_k(i)$, (iii) $\forall j \in [d_v], v_j(i_1) = \alpha v_j(i)$ and $v_j(i_2) = (1-\alpha)v_j(i)$.*

▶ **Definition 5** (Fractional packing). *Let $\widetilde{I}$ be (d_g, d_v)-dimensional items, where for each item $i \in \widetilde{I}$, we are given a set $X(i)$ of axes perpendicular to which we can repeatedly slice i ($X(i)$ can be empty, which would mean that the item cannot be sliced). If we slice items as per their given axes and then pack the slices into bins, then the resulting packing is called a* fractional *bin packing. (See Figure 1 for an example.)*

4.2 Properties of `round`

▶ **Definition 6.** *The density vector of a (d_g, d_v) item i is the vector $v_{\text{span}} := [v_0(i)/\text{span}(i), v_1(i)/\text{span}(i), \ldots, v_{d_v}(i)/\text{span}(i)]$. Recall that $v_0(i) := \text{vol}(i)$.*

The subroutine `round`(I) returns a set of pairs of the form $(\widetilde{I}, D)$. **Condition C1** is defined as the following constraints over each pair $(\widetilde{I}, D)$:

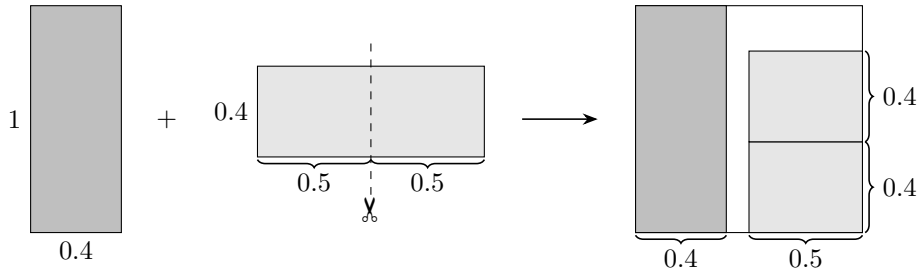

Figure 1 Example of a fractional packing of two items into a bin.

- **C1.1.** *Small discard:* $D \subseteq I$ and $\text{span}(D) \leq \varepsilon \, \text{span}(I)$.
- **C1.2.** *Bijection from $I - D$ to $\widetilde{I}$:* Each item in $\widetilde{I}$ is obtained by modifying an item in $I - D$. Let π be the corresponding bijection from $I - D$ to $\widetilde{I}$.
- **C1.3.** *Homogeneity properties:* `round` partitions items in $\widetilde{I}$ into a constant number of classes: $\widetilde{K}_1, \widetilde{K}_2, \ldots, \widetilde{K}_q$. These classes should satisfy the following properties, which we call *homogeneity* properties:
  - All items in a class have the same density vector.
  - For each class $\widetilde{K}_j$, we decide the set X of axes perpendicular to which we can slice items in $\widetilde{K}_j$. If items in a class $\widetilde{K}_j$ are not allowed to be sliced perpendicular to dimension k, then all items in that class have the same length along dimension k. (For example, if $d_g = 2$ and only vertical cuts are forbidden, then all items have the same width. However, they can have different heights.)
- **C1.4.** *Bounded expansion:* Let C be any configuration of I and $\widetilde{K}$ be any one of the constant number of classes of $\widetilde{I}$. Let $\widetilde{C} := \{\pi(i) : i \in C - D\}$. Then we need to prove that $\text{span}(\widetilde{K} \cap \widetilde{C}) \leq c_{\max}$ for some constant $c_{\max}$.

Intuitively, the homogeneity properties allow us to replace (a slice of) an item in a fractional packing by slices of other items of the same class. Thus, while trying to get a fractional packing, we can focus on the item classes, which are constant in number, instead of focusing on the n items. Intuitively, bounded expansion (C1.4) ensures that we do not round up items too much.

Condition C2 (also called *structural theorem*): For some constant $\rho > 0$ and for some $(\widetilde{I}, D) \in \texttt{round}(I)$, $\text{fsopt}(\widetilde{I}) \leq \rho \, \text{opt}(I) + O(1)$.

Intuitively, the structural theorem says that allowing slicing as per `round` and imposing a structure on the packing does not increase the minimum number of bins by too much. We will see that `rnaPack`'s AAR increases with ρ, so we want ρ to be small.

4.3 `complexPack` and `unround`

- **Condition C3**: For some constant $\alpha > 0$ and for any $(\widetilde{I}, D) \in \texttt{round}(I)$ and any $\widetilde{X} \subseteq \widetilde{I}$, $\texttt{complexPack}(\widetilde{X})$ packs $\widetilde{X}$ into at most $\alpha \, \text{fsopt}(\widetilde{X}) + O(1)$ bins.
- **Condition C4**: For some constant $\gamma > 0$, if $\texttt{complexPack}(\widetilde{S})$ outputs a packing of $\widetilde{S}$ into m bins, then `unround` converts that to a packing of $S - D$ into $\gamma m + O(1)$ bins.

Intuitively, Condition C3 says that we can find a packing of the rounded items that is close to the optimal fractional structured packing. Condition C4 says that unrounding does not increase the number of bins by too much. We will see that `rnaPack`'s AAR increases with α and γ, so we want α and γ to be small. If `round` only increases the dimensions of items, then unrounding is trivial and $\gamma = 1$.

4.4 AAR of R&A

Recall that `simplePack` is a $2b(d_v+1)$-approximation algorithm for (d_g, d_v) BP (see Section 3). Our key ingredient in the analysis of R&A is the following lemma. We give a very brief outline of the proof here and defer the full proof to Appendix C.

▶ **Lemma 7.** *Let $\widetilde{S}$ be as computed by* `rnaPack`(I, β, ε)*. Then with high probability, we get* $\text{fsopt}(\widetilde{S}) \leq \text{fsopt}(\widetilde{I})/\beta + 2b\mu\varepsilon \,\text{opt}(I) + O(1/\varepsilon^2)$.

Proof sketch. Our proof of Lemma 7 is inspired by the analysis in [35]. We prove it by analyzing the *fractional structured* configuration LP of $\widetilde{I}$.

▶ **Definition 8.** *Let $(\widetilde{I}, D) \in$* `round`(I)*. Suppose* `round` *partitioned $\widetilde{I}$ into classes $\widetilde{K}_1, \ldots \widetilde{K}_q$. Let $\mathcal{C}_f$ be the set of all structured configurations of items in $\widetilde{I}$ that allow items to be sliced as per* `round`*. For any $\widetilde{S} \subseteq \widetilde{I}$, the fractional structured configuration LP of $\widetilde{S}$, denoted as* $\text{fsLP}(\widetilde{S})$*, is*

$$\min_{x \in \mathbb{R}^{|\mathcal{C}_f|}_{\geq 0}} \sum_{C \in \mathcal{C}_f} x_C \quad \textit{where} \quad \sum_{C \in \mathcal{C}_f} \text{span}(C \cap \widetilde{K}_j) x_C \;\geq\; \text{span}(\widetilde{S} \cap \widetilde{K}_j) \quad \forall j \in [q]$$

The integer version of this program is called $\text{fsIP}(\widetilde{S})$*. The optimal objective values of* $\text{fsLP}(\widetilde{S})$ *and* $\text{fsIP}(\widetilde{S})$ *are denoted as* $\text{fsLP}^*(\widetilde{S})$ *and* $\text{fsIP}^*(\widetilde{S})$*, respectively.*

Intuitively, fsIP is the same as the structured fractional bin packing problem because of the downward closure property and homogeneity, so $\text{fsIP}^*(\widetilde{S}) \approx \text{fsopt}(\widetilde{S})$ (see Lemma 17 in Appendix C for the proof). By homogeneity (C1.3), the number of constraints in this LP is a constant q. So, by Rank Lemma[2], we can show that $|\,\text{fsopt}(\widetilde{S}) - \text{fsLP}^*(\widetilde{S})| \in O(1)$ (see Lemma 18 in Appendix C for the proof). Now to prove Lemma 7, roughly, we need to show that $\text{fsLP}^*(\widetilde{S}) \lessapprox \text{fsLP}^*(\widetilde{I})/\beta$.

The RHS in the j^{th} constraint of $\text{fsLP}(\widetilde{S})$ is a random variable $\text{span}(\widetilde{S} \cap \widetilde{K}_j)$. The RHS in the j^{th} constraint of $\text{fsLP}(\widetilde{I})$ is $\text{span}(\widetilde{K}_j)$. Now using properties of randomized rounding, one can show $\forall i \in I$, $\Pr(i \in S) \leq 1/\beta$. (see Lemma 16 in Appendix C for the proof). Using this, we obtain $\mathbb{E}(\text{span}(\widetilde{S} \cap \widetilde{K}_j)) \leq \text{span}(\widetilde{K}_j)/\beta$. In fact, we can harness the randomness of $\widetilde{S}$, the bounded expansion property (C1.4), and McDiarmid's inequality [41] to show that $\text{span}(\widetilde{S} \cap \widetilde{K}_j) \lessapprox \text{span}(\widetilde{K}_j)/\beta$. Therefore, if x^* is an optimal solution to $\text{fsLP}(\widetilde{I})$, then x^*/β is *roughly* a solution to $\text{fsLP}(\widetilde{S})$, which implies $\text{fsLP}^*(\widetilde{S}) \lessapprox \text{fsLP}^*(\widetilde{I})/\beta$ (see Lemma 21 in Appendix C for details). ◀

▶ **Theorem 9.** *With high probability, the number of bins used by* `rnaPack`(I, β, ε) *is at most* $((\ln \beta)\mu + (\gamma\alpha\rho)/\beta + 2b(d_v + 1 + \gamma\alpha\mu)\varepsilon)\,\text{opt}(I) + O(1/\varepsilon^2)$.

Proof sketch. (See Appendix C for full proof)

- We use at most $T \leq (\ln \beta)\mu \,\text{opt}(I) + O(1)$ bins to pack $I - S$.
- We use at most $b\lceil 2\,\text{span}(D)\rceil \leq 2b(d_v+1)\varepsilon \,\text{opt}(I) + b$ bins to pack $S \cap D$ using `simplePack`.
- $S - D$ occupies at most $\gamma\alpha \,\text{fsopt}(\widetilde{S}) + O(1) \leq \gamma\alpha\,(\rho/\beta + 2b\mu\varepsilon)\,\text{opt}(I) + O(1/\varepsilon^2)$ bins by Lemma 7 and Conditions C2, C3, and C4. ◀

Thus, the AAR of `rnaPack`(I) is roughly $\mu \ln \beta + \gamma\alpha\rho/\beta$. This is minimized for $\beta = \gamma\alpha\rho/\mu$ and the minimum value is $\mu\,(1 + \ln(\alpha\gamma\rho/\mu))$. As we require $\beta \geq 1$, we get this AAR only when $\gamma\alpha\rho \geq \mu$. If $\mu \geq \gamma\alpha\rho$, the optimal β is 1 and the AAR is roughly $\gamma\alpha\rho$.

[2] Rank Lemma: the number of non-zero variables in an extreme point of the set $\{x : Ax \geq b, x \geq 0\}$ is at most $\text{rank}(A)$. See Lemma 2.1.4 in [39].

4.5 Example: `simplePack`

We will show how to use `simplePack` with the R&A framework. Recall that `simplePack` is a $2b(d_v+1)$-approximation algorithm for (d_g, d_v) BP (see Section 3). Using the R&A framework on `simplePack` will improve its AAR from $2b(d_v+1)$ to $b(1+\ln(2(d_v+1)))+O(\varepsilon)$. To do this, we need to show how to implement `round`, `complexPack`, and `unround`.

1. `solveConfigLP`(I): Using the (d_g, d_v) KS algorithm of Section 3 and the LP algorithm of [47], we get a $b(1+\varepsilon)$-approximate solution to configLP(I). Therefore, $\mu = b(1+\varepsilon)$.
2. `round`(I): returns just one pair: $(\widetilde{I}, \{\})$, where $\widetilde{I} := \{\pi(i) : i \in I\}$ and $\pi(i)$ is an item having height (i.e., d_g^{th} geometric dimension) equal to span(i), all other geometric dimensions equal to 1, and all vector dimensions equal to span(i). There is just one class in $\widetilde{I}$, and we allow all items to be sliced perpendicular to the height, so the homogeneity properties are satisfied. Also, $c_{\max} = d_v + 1$ by Lemma 1 (since for any configuration C, we have $\text{span}(\pi(C)) = \text{span}(C) \le (d_v+1)\,\text{opt}(C) = d_v + 1$).
3. **Structural theorem**: We take structured packing to be the set of all possible packings. We can treat $\widetilde{I}$ as the 1-D instance $\{\text{span}(i) : i \in I\}$, so $\text{fsopt}(\widetilde{I}) = \lceil \text{span}(I) \rceil \le (d_v+1)\,\text{opt}(I)$, where the inequality follows from Lemma 1. So, $\rho = d_v + 1$.
4. `complexPack`$(\widetilde{S})$: We can treat $\widetilde{S}$ as the 1-D instance $\{\text{span}(i) : i \in S\}$ and pack it using Next-Fit [30]. Hence, $|\texttt{complexPack}(\widetilde{S})| \le \lceil 2\,\text{span}(S) \rceil \le 2\lceil \text{span}(S) \rceil = 2\,\text{fsopt}(\widetilde{S})$. So, $\alpha = 2$.
5. `unround`$(\widetilde{J})$: We are given a packing $\widetilde{J}$ of items $\widetilde{S}$. For each bin in $\widetilde{J}$, we can pack the corresponding unrounded items into b bins. Therefore, $\gamma = b$.

Hence, we get an AAR of $\mu(1+\ln(\gamma\alpha\rho/\mu)) + O(\varepsilon) \approx b(1+\ln(2(d_v+1))) + O(\varepsilon)$.

For $d_g = 2$, we can slightly improve the AAR by using the $(2+\varepsilon)$-approximation algorithm of [37] for $(2, d_v)$ KS. This gives us an AAR of $2(1+\ln(3(d_v+1))) + O(\varepsilon)$. This is better than the AAR of `betterSimplePack` for $d_v \ge 3$.

The above example is presented only to illustrate an easy use of the R&A framework. It doesn't exploit the full power of the R&A framework. The algorithm `cbPack`, which we describe in Section 5, uses more sophisticated subroutines `round`, `complexPack` and `unround`, and uses a more intricate definition of fractional structured packing to get an even better AAR of $2(1+\ln(\frac{d+4}{2})) + \varepsilon$ (improves to $2(1+\ln(19/12)) + \varepsilon \approx 2.919 + \varepsilon$ for $d = 1$).

5 Improved Approximation Algorithms

In this section, we give an overview of the `cbPack` algorithm for $(2, d)$ BP, which is inspired from the 1.5 asymptotic approximation algorithm for 2-D GBP [44]. `cbPack` is based on the following two-step procedure, as is common in many packing algorithms.

In the first step (*structural step*, Appendices F.1–F.5 and F.7 in [38]), we show the existence of a good *structured* solution. Formally, we show that if items I can be packed into m bins, then we can round I to get a new instance $\widetilde{I}$ such that $\text{fsopt}(\widetilde{I}) \le \rho m + O(1)$ for some constant ρ, where $\text{fsopt}(\widetilde{I})$ is the number of bins in the optimal *structured fractional* packing of I. Roughly, the notion of *structured* packing that we use here, which we call *compartmental packing*, imposes the following additional constraints over the *container-based* packing of [44]: (i) An item i is called *dense* iff $v_{\max}(i)/a(i)$ is above a certain threshold. If a bin contains dense items, then we reserve a sufficiently-large rectangular region exclusively for them. (ii) For a constant ε, for every $j \in [d]$, the sum of v_j of items in each bin is at most $1-\varepsilon$. The proof of our structural result differs significantly from that of [44] because the presence of vector dimensions inhibit a straightforward application of their techniques.

In the second step (*algorithmic step*), we show how to find a near-optimal structured solution. `cbPack` first rounds I to $\widetilde{I}$ and then uses brute-force and LP to pack the items into containers, similar to [44] or [34]. Then we convert that packing to a non-fractional packing of I with only a tiny increase in the number of bins (see Appendix F.8 of [38] for the algorithmic step).

Then we show that `cbPack` fits into the R&A framework (i.e., components from `cbPack` can be used as `round`, `complexPack` and `unround`) and gives an AAR of roughly $2(1+\ln(\rho/2))$. To (approximately) solve the configuration LP, we use the LP algorithm from [47] and the $(2+\varepsilon)$-approximation algorithm for $(2,\, d)$ KS from [37] (see Appendix F.9 of [38] for the details of fitting `cbPack` into the R&A framework).

5.1 Overview of Structural Result

We now give a brief overview of some key ideas used in our structural result. Due to space limitations, the details of the structural result and the algorithm can be found in Appendix F of [38]. Like [44], we start with a packing of input I into m bins, and transform it into a structured fractional packing of $\widetilde{I}$ into $\rho m + O(1)$ bins. To do this, just like many other papers on packing, we first classify the items into different classes based on their geometric and vector dimensions, and densities as follows. First, we identify two constants (which depend on ε) $\varepsilon_1, \varepsilon_2 \in (0,1)$ such that $\varepsilon_1 > \varepsilon_2$ and the set of medium items defined as

$$I_{\text{med}} := \left\{ i \in I : w(i) \in (\varepsilon_2, \varepsilon_1] \vee h(i) \in (\varepsilon_2, \varepsilon_1] \vee \left(\bigvee_{j=1}^{d} v_j(i) \in (\varepsilon_2, \varepsilon_1] \right) \right\}$$

has a very small span, i.e., $\text{span}(I_{\text{med}}) \leq \varepsilon\, \text{span}(I)$. Hence, we can treat the items in I_{med} separately as they can be packed in a very small number of bins. Also, $\varepsilon_1, \varepsilon_2$ satisfy that $\varepsilon_2 \leq \varepsilon_1^2 \varepsilon/2$. Informally, ε_2 is very small when compared to ε_1. Hence, every item in $I \backslash I_{\text{med}}$ has the property that both width and height are big ($> \varepsilon_1$) or both width and height are small ($\leq \varepsilon_2$) or it is skewed. Based on $\varepsilon_1, \varepsilon_2$, we classify every item $i \in I \backslash I_{\text{med}}$ as follows. First, we classify by geometric dimensions as follows.

- Big item: $w(i) > \varepsilon_1$ and $h(i) > \varepsilon_1$.
- Wide item: $w(i) > \varepsilon_1$ and $h(i) \leq \varepsilon_2$.
- Tall item: $w(i) \leq \varepsilon_2$ and $h(i) > \varepsilon_1$.
- Small item: $w(i) \leq \varepsilon_2$ and $h(i) \leq \varepsilon_2$.

Then, we perform another classification depending on vector dimensions.

▶ **Definition 10** (Dense items). *Item i is dense iff either $a(i) = 0$ or $v_{\max}(i)/a(i) > 1/\varepsilon_1^2$.*

▶ **Definition 11** (Heavy and light items). *A dense item i is said to be heavy in vector dimension j iff $v_j(i) \geq \varepsilon_1$. Otherwise i is said to be light in dimension j. If i is heavy in some dimension, then i is said to be heavy, otherwise i is light.*

More formal details of this classification are provided in Appendix F.1 of [38].

Then, the structural result is obtained in three steps:

(i) In the first step, we round up one geometric dimension of each item and pack the items into roughly $\rho m + O(1)$ bins. We call these bins *quarter-structured* (see Appendices F.2 and F.3 of [38]).

(ii) In the second step, we round the remaining dimensions of items and partition them into classes such that they satisfy the *homogeneity properties* (see Section 4.2). We allow slicing and repack the items into almost the same number of bins. We call the resulting bin packing *semi-structured* (see Appendices F.4 and F.5 of [38]).

(iii) In the third and final step, we transform the packing into a compartmental packing (see Appendices F.6 and F.7 of [38]). Compartmental packings have nice properties which help us find them efficiently. Roughly, a compartmental packing is a semi-structured packing that is also container-based. This step is very similar to [44].

In steps (i), (ii) and (iii), [44] uses the Next-Fit-Decreasing-Height (NFDH) algorithm [14] to pack items of $O(\varepsilon m)$ area into $O(\varepsilon m)$ bins. This does not work when vector dimensions are present as an item of low area can have large weights. In step (ii), [44] uses *linear grouping*, i.e., each item is moved in place of a geometrically larger item so that it can be rounded up. Vector dimensions make such cross-bin movement difficult, since that can violate bins' weight capacities. [44] uses cross-bin movement in step (i) too.

We first observe that most difficulties associated with vector dimensions disappear if items' *density* is upper-bounded by a constant. Density of item i is defined as $v_{\max}(i)/a(i)$. Specifically, if items of bounded density (we call them *non-dense* items) have small area, then `simplePack` can pack them into a small number of bins. Linear grouping can also be made to work for such items with some more effort. Hence, we segregate items as *dense* and *non-dense*. We reserve a thin rectangular region in bins for dense items, and the rest is reserved for non-dense items.

Furthermore, dense items in a bin must have low total area, due to their high density. If we reserve enough space for them in the bin, we can always pack them in their reserved region using NFDH (see Lemma 15 in Appendix B). Such a guarantee means that we can essentially ignore their geometric dimensions and simply treat them as vectors.

In step (ii), we want to round up vector dimensions with only a marginal increase in the number of bins. To do this, we require each quarter-structured bin to be ε-slacked. ε-slackness roughly means that for a set J of items in a bin, $\forall j \in [d], v_j(J) \leq 1 - \varepsilon$ (see Appendix F.3 of [38] for a formal description). ε-slackness also helps us use existing algorithms for resource-augmented vector bin packing as subroutines. Also, during the rounding step, we round down the weight of some dense items, and ε-slackness allows us to *unround* with no increase in the number of bins.

The observations above guide our definition of *quarter-structured*. Roughly, a packing is quarter-structured if items having large width have their width and x-coordinate rounded to a multiple of $\varepsilon_1^2/4$ and each bin is ε-slacked. We reserve a thin rectangular region of width $\varepsilon_1/2$ for packing dense items (only if the bin contains dense items).

In step (i), [44] uses a standard cutting-strip argument: They create a strip of width ε_1 next to an edge of the bin (see Figure 2). Items completely inside the strip (called dark items), have small area and are packed separately using NFDH. Items intersecting the strip's boundary (called shaded items) are removed. This creates an empty space of width ε_1 in the bin. Using this empty space, items outside the strip (called dotted items) can then have their width and x-coordinate rounded to a multiple of $\varepsilon_1^2/2$. Their key idea is how to pair up most bins so that shaded items from two bins can be rounded and packed together into a new bin. This is roughly why they get an AAR of $1.5 + \varepsilon$.

We use the cutting-strip argument too, but with some differences. We cannot freely mix shaded items from different bins if they have large weight, and we cannot simply pack dark items into a small number of bins. We also need bins to be slacked. So, we get a larger AAR of $d + 4 + \varepsilon$. For $d = 1$, however, we allow mixing items using more sophisticated techniques, which improves the AAR to $19/6 + \varepsilon$. Also, we round dotted items to a multiple of $\varepsilon_1^2/4$ instead of $\varepsilon_1^2/2$, which leaves an empty strip of width $\varepsilon_1/2$ in the bin even after rounding, and we reserve this space for dense items. This gives us a quarter-structured packing.

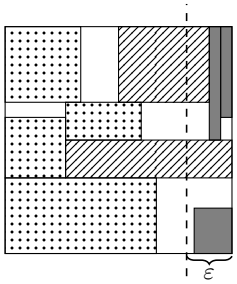

Figure 2 Example of classifying items as dark, shaded and dotted based on an ε-strip.

Based on the broad ideas above, we make more changes to the quarter-structured packing to get a compartmental packing. Finally, in the algorithmic step, we use a combination of brute-force and LP to pack items into compartments and efficiently find a *good* compartmental packing. An overview is given in the section below.

5.2 Overview of the Algorithmic Step

Note that in any bin, the number of big items and heavy items can be at most a constant in number. The nice structure obtained in the structural step has the property that in each bin there are constant number of compartments in which all the light items (which are not big) are packed. With these arguments, in any bin of this nice structure, the number of big items, heavy items, and compartments is constant in number. Moreover, the sizes of the compartments also belong to a polynomial sized set. There are other things to be considered (e.g., the slack type of the bin), but one of the main implications of the structural result is that the number of possible *configurations* of each bin is at most a constant. Also, note that the number of bins itself is bounded by the number of items n. Thus, in polynomial time (by brute force), we can guess the number of bins in the nice structure, and the configuration of each bin (for full details, see Appendix F.8.1 of [38]).

Once the configurations have been guessed, we are left with the task of packing the wide, tall, small, and light items in the compartments. First, as an intermediate step, we obtain a fractional packing of these items into the compartments by solving a linear program (if the linear program is infeasible, we realize that our guess of the bin configurations is incorrect). See Appendix F.8.2 of [38] for the details of the linear program. The main purpose of solving the linear program (if it turns out to be feasible) is to further divide compartments into what are called *containers* (see Appendix F.8.3 of [38]). Finally, the wide, tall, small, and light items are non-fractionally packed into the containers greedily. There can be some items which can't be packed by our greedy strategy, but we can prove that their span is so small that we can pack them using very few extra bins. See Appendices F.8.5, F.8.6, and F.8.6 of [38] for the details of packing the wide, tall, small, and light items.

In Theorem 66 of Appendix F.8.7 of [38], we compute the approximation factor of this algorithm:

▶ **Theorem 12.** *For general d, when rotations are forbidden, the above algorithm packs the input set I into at most*

$$(1 + 23\varepsilon)(d + 4)\,\mathrm{opt}(I) + O_\varepsilon(1)$$

number of bins. Here $O_\varepsilon(1)$ hides a constant that depends on ε.

In the special case of $d = 1$, we obtain a better approximation ratio.

▶ **Theorem 13.** *If $d = 1$ and rotations are forbidden, we can pack the input set I into at most*

$$(1 + 23\varepsilon)\left(3 + \frac{1}{6} + \frac{\varepsilon}{1-\varepsilon}\right)\mathrm{opt}(I) + O_\varepsilon(1)$$

number of bins. Here $O_\varepsilon(1)$ hides a constant that depends on ε.

Using similar techniques, we can design an algorithm for the case with rotations too. Using the Round&Approx framework on top of this algorithm, and scaling ε appropriately, we get the following approximation ratios.

- For general d, with no rotations, we get an approximation ratio of $2\left(1 + \ln\left(\frac{d+4}{2}\right)\right) + \varepsilon$.
- When $d = 1$ and rotations are forbidden, we get an approximation ratio of $\approx 2.919 + \varepsilon$.
- For general d, with rotations allowed, we get an approximation ratio of $2\left(1 + \ln\left(\frac{d+3}{2}\right)\right) + \varepsilon$.
- When $d = 1$ and rotations are allowed, we get an approximation ratio of $\approx 2.811 + \varepsilon$.

References

1. M. T. Alonso, R. Alvarez-Valdes, Manuel Iori, F. Parreño, and J. M. Tamarit. Mathematical models for multicontainer loading problems. *Omega*, 66:106–117, 2017. doi:10.1016/j.omega.2016.02.002.
2. Samir V. Amiouny, John J. Bartholdi III, John H. Vande Vate, and Jixian Zhang. Balanced loading. *Operations Research*, 40(2):238–246, 1992. doi:10.1287/opre.40.2.238.
3. Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2009. doi:10.1137/080736831.
4. Nikhil Bansal, Jose R. Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31:31–49, 2006. doi:10.1287/moor.1050.0168.
5. Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1561–1579. SIAM, 2016. doi:10.1137/1.9781611974331.ch106.
6. Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *SODA*, pages 13–25, 2014. doi:10.1137/1.9781611973402.2.
7. Suman Kalyan Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 4955–4966, 2019.
8. Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013. doi:10.1016/j.ejor.2012.12.006.
9. Alberto Caprara. Packing 2-dimensional bins in harmony. In *FOCS*, pages 490–499, 2002. doi:10.1109/SFCS.2002.1181973.
10. Alberto Caprara. Packing d-dimensional bins in d stages. *Mathematics of Operations Research*, 33:203–215, 2008. doi:10.1287/moor.1070.0289.
11. Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004. doi:10.1137/S0097539799356265.
12. Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi:10.1016/j.cosrev.2016.12.001.
13. Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: Survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. Springer New York, 2013. doi:10.1007/978-1-4419-7997-1_35.
14. Edward G. Coffman, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808–826, 1980. doi:10.1137/0209062.

15 Florian Diedrich, Rolf Harren, Klaus Jansen, Ralf Thöle, and Henning Thomas. Approximation algorithms for 3D orthogonal knapsack. *Journal of Computer Science and Technology*, 23(5):749, 2008. doi:10.1007/s11390-008-9170-7.

16 Khaled M. Elbassioni, Naveen Garg, Divya Gupta, Amit Kumar, Vishal Narula, and Arindam Pal. Approximation algorithms for the unsplittable flow problem on paths and trees. In *FSTTCS*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 267–275, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.267.

17 Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981. doi:10.1007/BF02579456.

18 A. M. Frieze and M. R. B. Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. *EJOR*, 15:100–109, 1984.

19 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight $(3/2+\varepsilon)$ approximation for skewed strip packing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.44.

20 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 260–271. IEEE, 2017. Full version available at http://www.dii.uchile.cl/~awiese/2DK_full_version.pdf. doi:10.1109/FOCS.2017.32.

21 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *FSTTCS*, pages 9:1–9:14, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.9.

22 Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. A 3-approximation algorithm for maximum independent set of rectangles. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 894–905. SIAM, 2022. doi:10.1137/1.9781611977073.38.

23 Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy, and Andreas Wiese. A $(2+\varepsilon)$-approximation algorithm for maximum independent set of rectangles, 2021. arXiv:2106.00623.

24 Paul C Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120, 1965. doi:10.1287/opre.13.1.94.

25 Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *SODA*, pages 2616–2625, 2017. doi:10.1137/1.9781611974782.172.

26 Klaus Jansen, Arindam Khan, Marvin Lira, and K. V. N. Sreenivas. A PTAS for packing hypercubes into a knapsack. In *49th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 229 of *LIPIcs*, pages 78:1–78:20, 2022.

27 Klaus Jansen and Lars Prädel. New approximability results for two-dimensional bin packing. *Algorithmica*, 74(1):208–269, 2016. doi:10.1007/s00453-014-9943-z.

28 Klaus Jansen and Guochuan Zhang. On rectangle packing: maximizing benefits. In *SODA*, pages 204–213, 2004.

29 D. S. Johnson. Approximation algorithms for combinatorial problems. In *STOC*, pages 38–49, 1973.

30 David S Johnson. *Near-Optimal Bin Packing Algorithms*. PhD thesis, Massachusetts Institute of Technology, USA, 1973.

31 Matthew Joseph, Michael J. Kearns, Jamie H. Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *NeurIPS*, pages 325–333, 2016.

32 Debajyoti Kar, Arindam Khan, and Andreas Wiese. Approximation algorithms for round-ufp and round-sap, 2022. doi:10.48550/ARXIV.2202.03492.

33 Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

34 Claire Kenyon and Eric Rémila. Approximate strip packing. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 31–36, 1996. doi:10.1109/SFCS.1996.548461.

35 Arindam Khan. *Approximation algorithms for multidimensional bin packing*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2016.

36 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.47.

37 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack, 2021. arXiv:2102.05854.

38 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenvas. Geometry meets vectors: Approximation algorithms for multidimensional packing, 2021. arXiv:2106.13951v1.

39 Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

40 Eugene L Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979. doi:10.1287/moor.4.4.339.

41 Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

42 Célia Paquay, Michael Schyns, and Sabine Limbourg. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, 23(1-2):187–213, 2016. doi:10.1111/itor.12111.

43 Deval Patel, Arindam Khan, and Anand Louis. Group fairness for knapsack problems. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1001–1009, 2021.

44 Lars Dennis Prädel. *Approximation Algorithms for Geometric Packing Problems*. PhD thesis, Kiel University, 2012. URL: https://macau.uni-kiel.de/servlets/MCRFileNodeServlet/dissertation_derivate_00004634/dissertation-praedel.pdf?AC=N.

45 Arka Ray. There is no APTAS for 2-dimensional vector bin packing: Revisited, 2021. arXiv:2104.13362.

46 Sai Sandeep. Almost optimal inapproximability of multidimensional packing problems. In *Symposium on Foundations of Computer Science (FOCS)*, pages 245–256, 2022. doi:10.1109/FOCS52979.2021.00033.

47 Eklavya Sharma. An approximation algorithm for covering linear programs and its application to bin-packing, 2020. arXiv:2011.11268.

48 Knut Olav Brathaug Sørset. A heuristic approach to the three-dimensional bin packing problem with weight constraints. Master's thesis, Høgskolen i Molde-Vitenskapelig høgskole i logistikk, 2019.

49 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.

50 Gregory S. Taylor, Yupo Chan, and Ghulam Rasool. A three-dimensional bin-packing model: exact multicriteria solution and computational complexity. *Annals of Operations Research*, 251(1-2):397–427, 2017. doi:10.1007/s10479-015-2048-5.

51 Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. Group-fairness in influence maximization. In *IJCAI*, pages 5997–6005, 2019.

52 Gerhard J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Inf. Process. Lett.*, 64(6):293–297, 1997. doi:10.1016/S0020-0190(97)00179-8.

53 Guang Yang. *gbp: a bin packing problem solver*, 2017. R package version 0.1.0.4. URL: https://CRAN.R-project.org/package=gbp.

A Formal Definition of (d_g, d_v) Packing

▶ **Definition 14.** *A valid* packing *of (d_g, d_v)-dimensional items into a (d_g, d_v)-dimensional bin is an arrangement of the items in the bin such that all of the following hold:*

1. *All items are packed in an axis parallel manner, i.e., each item has its faces parallel to the faces of the bin. Formally, to pack item i into a bin, we need to decide its position $(x_1(i), x_2(i), \ldots, x_{d_g}(i))$ in the bin, and the item will be packed in the cuboidal region $\prod_{j=1}^{d_g}[x_j(i), x_j(i) + \ell_j(i)]$.*
2. *The items are non-overlapping, i.e., the interiors of any two items do not intersect. Formally, for any two items i_1 and i_2 in the same bin, the sets $\prod_{j=1}^{d_g}(x_j(i_1), x_j(i_1)+\ell_j(i_1))$ and $\prod_{j=1}^{d_g}(x_j(i_2), x_j(i_2) + \ell_j(i_2))$ do not intersect.*
3. *All items are packed completely inside the bin, i.e., for each item i and each $j \in [d_g]$, $x_j(i) \geq 0$ and $x_j(i) + \ell_j(i) \leq 1$.*
4. *In each bin, the total weight in each of the d_v dimensions is at most one, i.e., for each set S of items in a bin and each $j \in [d_v]$, we have $\sum_{i \in S} v_j(i) \leq 1$.*

B Next-Fit Decreasing Height (NFDH)

▶ **Lemma 15** (NFDH for strip packing [14])**.** *A set I of rectangles can be packed into a strip of height at most $2a(I) + \max_{i \in I} h(i)$ using the Next-Fit Decreasing Height (NFDH) algorithm.*

C Details of the R&A Framework

Algorithm 1 `rnaPack`(I, β, ε): Computes a bin packing of (d_g, d_v) items I, and $\beta \geq 1$.

$\widehat{x} = \texttt{solveConfigLP}(I)$
repeat $T := \lceil (\ln \beta)\|\widehat{x}\|_1 \rceil$ **times**
  Select a configuration C with probability $\widehat{x}_C / \|\widehat{x}\|_1$.
  Pack a bin according to C.
end repeat
Let S be the unpacked items from I. // *S is called the set of* residual items.
Initialize J_{best} to `null`.
for $(\widetilde{I}, D) \in \texttt{round}(I)$ **do** // `round`(I) *outputs a set of pairs.*
  $J_D = \texttt{simplePack}(S \cap D)$
  Let π be a bijection from $I - D$ to $\widetilde{I}$. Let $\widetilde{S} := \{\pi(i) : i \in S - D\}$.
  $\widetilde{J} = \texttt{complexPack}(\widetilde{S})$
  $J = \texttt{unround}(\widetilde{J})$
  if J_{best} is `null` or $|J_D \cup J| < |J_{\text{best}}|$ **then**
    $J_{\text{best}} = J_D \cup J$
  end if
end for
Pack S according to J_{best}.

▶ **Lemma 16.** $\forall i \in I$, $\Pr(i \in S) \leq \exp\left(-\frac{T}{\|\widehat{x}\|_1}\right) \leq \frac{1}{\beta}$.

Proof. Let $C_1, C_2, \ldots, C_T$ be the configurations chosen during randomized rounding (line 3 in Algorithm 1). Let $\mathcal{C}_i$ be the configurations that contain the element i.

$$\begin{aligned}
\Pr(i \in S) &= \Pr\left(\bigwedge_{t=1}^{T} (C_t \notin \mathcal{C}_i)\right) = \prod_{t=1}^{T} \Pr(C_t \notin \mathcal{C}_i) && \text{(all } C_t \text{ are independent)} \\
&= \prod_{t=1}^{T} \left(1 - \sum_{C \in \mathcal{C}_i} \Pr(C_t = C)\right) = \left(1 - \sum_{C \in \mathcal{C}_i} \frac{\widehat{x}_C}{\|\widehat{x}\|_1}\right)^T \\
&\leq \left(1 - \frac{1}{\|\widehat{x}\|_1}\right)^T && \text{(constraint in configuration LP for item } i\text{)} \\
&\leq \exp\left(-\frac{T}{\|\widehat{x}_1\|}\right) \leq \frac{1}{\beta}
\end{aligned}$$

◀

▶ **Lemma 17.** $\text{fsopt}(\widetilde{S}) \leq \text{fsIP}^*(\widetilde{S}) \leq \text{fsopt}(\widetilde{S}) + q$.

Proof. Due to the downward closure property, changing inequality constraints to equality constraints doesn't affect the optimum values of the above LP and IP. Therefore, $\text{fsIP}(\widetilde{S})$ is equivalent to the fractional structured bin packing problem.

The above definition of $\text{fsLP}(\widetilde{I})$ is problematic: the number of variables may be infinite if some classes allow slicing. We circumvent this problem by *discretizing* the configurations: Let δ be the smallest dimension of any item, i.e. $\delta := \min\left(\min_{j=1}^{d_g} \ell_j(i), \min_{j=1}^{d_v} v_j(i)\right)$.

In any optimal integral solution to $\text{fsLP}(\widetilde{I})$ that uses m bins, we can slice out some items from each class in each bin so that the span of each class in each bin is a multiple of δ^{d_g}/n. In each class, the total size of sliced out items across all bins is at most δ^{d_g}. Therefore, for each class, slices of that class can fit into a single item of that class. If each such single item is packed in a separate bin, the total number of bins used is at most $m + q$.

Therefore, we only need to consider configurations where either the span of each class is a multiple of δ^{d_g}/n or there is a single item in the configuration. This gives us a finite number of configurations and completes the proof. ◀

▶ **Lemma 18.** $\text{fsLP}^*(\widetilde{S}) \leq \text{fsIP}^*(\widetilde{S}) \leq \text{fsLP}^*(\widetilde{S}) + q$.

Proof. By Rank Lemma (Lemma 2.1.4 in [39]), the number of positive-valued variables in an extreme-point solution to a linear program is at most the number of constraints (other than the variable non-negativity constraints).

Thus, an optimal extreme-point solution to $\text{fsLP}(\widetilde{S})$ has at most q positive-valued variables. Rounding up those variables to the nearest integer will give us an integral solution and increase the objective value by at most q. Hence, $\text{fsIP}^*(\widetilde{S}) \leq \text{fsLP}^*(\widetilde{S}) + q$. ◀

Let $\text{configLP}(I)$ denote the configuration LP of items I and let $\text{configLP}^*(I)$ denote the optimal objective value of $\text{configLP}(I)$. Recall that `simplePack` is a $2b(d_v+1)$-approximation algorithm for (d_g, d_v) BP (see Section 3), where $b := 3$ when $d_g = 2$, $b := 9$ when $d_g = 3$, $b := 4^{d_g} + 2^{d_g}$ when $d_g > 3$, and $b := 1$ when $d_g = 1$.

▶ **Lemma 19.** *For a set I of (d_g, d_v)-dimensional items,* $\text{configLP}^*(I) \in \Theta(\text{span}(I)) + O(1)$.

Proof. Let A be the configuration matrix of I. Let x^* be the optimal solution to $\text{configLP}(I)$. In $\text{configLP}(I)$, the constraint for item i gives us $\sum_{C \in \mathcal{C}} A[i, C] x^*_C \geq 1$. Multiplying each constraint by $\text{span}(i)$ and adding these constraints together, we get

$$\begin{aligned}\operatorname{span}(I) &\le \sum_{C\in\mathcal{C}}\sum_{i\in I}\operatorname{span}(i)A[i,C]x^*_C = \sum_{C\in\mathcal{C}}\operatorname{span}(C)x^*_C\\ &\le (d_v+1)\sum_{C\in\mathcal{C}}x^*_C = (d_v+1)\operatorname{configLP}^*(I).\end{aligned}$$

Therefore, $\operatorname{configLP}^*(I) \ge \operatorname{span}(I)/(d_v+1)$. We also have

$$\operatorname{configLP}^*(I) \le \operatorname{opt}(I) \le |\,\texttt{simplePack}(I)| \le 2b\operatorname{span}(I) + b.$$

Therefore, $\operatorname{configLP}^*(I) \in \Theta(\operatorname{span}(I)) + O(1)$. ◀

▶ **Lemma 20** (Independent Bounded Difference Inequality [41])**.** *Let $X := [X_1, X_2, \ldots, X_n]$ be random variables with $X_j \in A_j$. Let $\phi : \prod_{i=1}^n A_j \mapsto \mathbb{R}$ be a function such that $|\phi(x) - \phi(y)| \le c_j$ whenever vectors x and y differ only in the j^{th} coordinate. Then for any $t \ge 0$,*

$$\Pr[\phi(X) - \mathbb{E}(\phi(X)) \ge t] \le \exp\left(-\frac{2t^2}{\sum_{j=1}^n c_j^2}\right).$$

▶ **Lemma 21.** *Let $\widetilde{S}$ be as computed by* `rnaPack`(I,β,ε)*. Let $\varepsilon \in (0,1)$ be a constant. When $\operatorname{span}(I)$ is large compared to $1/\varepsilon^2$, we get that with high probability*

$$\operatorname{fsLP}^*(\widetilde{S}) \le \frac{\operatorname{fsLP}^*(\widetilde{I})}{\beta} + 2b\mu\varepsilon\operatorname{opt}(I) + O(1).$$

Proof. Let $y \in \mathcal{C}^T$ be the configurations chosen during randomized rounding (on line 3 in `rnaPack`). When viewed as a vector of length T, all coordinates of y are independent. Define $\operatorname{uncovered}(y) := I - \bigcup_{t=1}^T y_t$. Let S be the unpacked items from I (see line 6 in `rnaPack`). Then $S = \operatorname{uncovered}(y)$.

Let $\widetilde{K}_1, \ldots, \widetilde{K}_q$ be the classes of $\widetilde{I}$. Let π be the bijection from $I - D$ to $\widetilde{I}$. For a set $X \subseteq I$, let $\widetilde{I}[X] := \{\pi(i) : i \in X - D\}$. For $j \in [q]$, define $\phi_j \in \mathcal{C}^T \mapsto \mathbb{R}_{\ge 0}$ as

$$\phi_j(y) := \operatorname{span}\left(\widetilde{K}_j \cap \widetilde{I}[\operatorname{uncovered}(y)]\right).$$

For any $X \subseteq I$, define $g_j(X) := \operatorname{span}(\widetilde{K}_j \cap \widetilde{I}[X])$. Then $\phi_j(y) = g_j(\operatorname{uncovered}(y))$ and g_j is a non-negative additive function.

Let $y^{(1)}, y^{(2)} \in \mathcal{C}^T$ such that $y^{(1)}$ and $y^{(2)}$ differ only in coordinate t. Let $C_1 := y^{(1)}_t$, $C_2 := y^{(2)}_t$, $S_1 := \operatorname{uncovered}(y^{(1)})$, $S_2 := \operatorname{uncovered}(y^{(2)})$, and $R := \bigcup_{t'\ne t} y^{(1)}_{t'} = \bigcup_{t'\ne t} y^{(2)}_{t'}$. Then $I - S_1 = R \cup C_1$ and $I - S_2 = R \cup C_2$. So, $S_1 - S_2 = (C_2 - C_1) - R \subseteq C_2$ and $S_2 - S_1 = (C_1 - C_2) - R \subseteq C_1$. Hence,

$$\begin{aligned}\left|\phi_j(y^{(1)}) - \phi_j(y^{(2)})\right| &= |g_j(S_1) - g_j(S_2)| = |g_j(S_1 - S_2) - g_j(S_2 - S_1)| \quad \text{(additivity of } g_j)\\ &\le \max(g_j(S_1 - S_2), g_j(S_2 - S_1)) \le \max(g_j(C_2), g_j(C_1))\\ &\le \max_{C\in\mathcal{C}} \operatorname{span}(\widetilde{K}_j \cap \widetilde{I}[C]) \le c_{\max}. \qquad \text{(by bounded expansion (C1.4))}\end{aligned}$$

By Lemma 16, $\Pr(i \in S) \le 1/\beta$. By linearity of $\mathbb{E}$ and additivity of g_j, we get

$$\mathbb{E}(\phi_j(y)) = \mathbb{E}(g_j(S)) = \sum_{i\in\widetilde{I}} g_j(\{i\})\Pr(i \in S) \le \sum_{i\in\widetilde{I}} g_j(\{i\})(1/\beta) = \frac{g_j(\widetilde{I})}{\beta} = \frac{\operatorname{span}(\widetilde{K}_j)}{\beta}.$$

$\forall j \in [q]$, define Q_j as the smallest prefix of $\widetilde{S} \cap \widetilde{K}_j$ such that either $Q_j = \widetilde{S} \cap \widetilde{K}_j$ or $\operatorname{span}(Q_j) \geq \varepsilon \left\|\widehat{x}\right\|_1 / q$. Define $Q := \bigcup_{j=1}^{q} Q_j$. Hence, $\operatorname{span}(Q) \leq \varepsilon \left\|\widehat{x}\right\|_1 + q \leq \varepsilon\mu \operatorname{opt}(I) + O(1)$.

$$\begin{aligned}
\operatorname{fsLP}^*(\widetilde{S}) &\leq \operatorname{fsLP}^*(\widetilde{S} - Q) + \operatorname{fsLP}^*(Q) \\
&\leq \operatorname{fsLP}^*(\widetilde{S} - Q) + b(2\operatorname{span}(Q) + 1) && \text{(by Section 3)} \\
&\leq \operatorname{fsLP}^*(\widetilde{S} - Q) + 2b\mu\varepsilon \operatorname{opt}(I) + O(1).
\end{aligned}$$

Now we will try to prove that with high probability, $\operatorname{fsLP}^*(\widetilde{S} - Q) \leq \operatorname{fsLP}^*(\widetilde{I})/\beta$.

If $Q_j = \widetilde{S} \cap \widetilde{K}_j$, then $\operatorname{span}(\widetilde{K}_j \cap (\widetilde{S} - Q)) = 0$. Otherwise,

$$\begin{aligned}
&\Pr\left[\operatorname{span}(\widetilde{K}_j \cap (\widetilde{S} - Q)) \geq \frac{\operatorname{span}(\widetilde{K}_j)}{\beta}\right] = \Pr\left[\operatorname{span}(\widetilde{K}_j \cap \widetilde{S}) - \frac{\operatorname{span}(\widetilde{K}_j)}{\beta} \geq \operatorname{span}(Q_j)\right] \\
&\leq \Pr\left[\phi_j(y) - \mathbb{E}(\phi_j(y)) \geq \frac{\varepsilon}{q}\left\|\widehat{x}\right\|_1\right] \leq \exp\left(-\frac{2}{Tc_{\max}^2}\left(\frac{\varepsilon}{q}\left\|\widehat{x}\right\|_1\right)^2\right) && \text{(Lemma 20)} \\
&\leq \exp\left(-\frac{2\varepsilon^2}{\ln(\beta)c_{\max}^2 q^2}\left\|\widehat{x}\right\|_1\right).
\end{aligned}$$

Therefore, by union bound, we get

$$\Pr\left[\bigvee_{j=1}^{q}\left(\operatorname{span}(\widetilde{K}_j \cap (\widetilde{S} - Q)) \geq \frac{\operatorname{span}(\widetilde{K}_j)}{\beta}\right)\right] \leq q\exp\left(-\frac{2\varepsilon^2}{\ln(\beta)c_{\max}^2 q^2}\left\|\widehat{x}\right\|_1\right).$$

Since $\operatorname{configLP}^*(I) \leq \left\|\widehat{x}\right\|_1 \leq \mu \operatorname{configLP}^*(I) + O(1)$, and $\operatorname{configLP}^*(I) \in \Theta(\operatorname{span}(I)) + O(1)$ (by Lemma 19), we get $\left\|\widehat{x}\right\|_1 \in \Theta(\operatorname{span}(I)) + O(1)$. When $\operatorname{span}(I)$ is very large compared to $1/\varepsilon^2$, we get that with high probability, $\forall j \in [q]$,

$$\operatorname{span}(\widetilde{K}_j \cap (\widetilde{S} - Q)) \leq \frac{\operatorname{span}(\widetilde{K}_j)}{\beta}.$$

Let x^* be the optimal solution to $\operatorname{fsLP}(\widetilde{I})$. Then with high probability, x^*/β is a feasible solution to $\operatorname{fsLP}(\widetilde{S} - Q)$. Therefore,

$$\operatorname{fsLP}^*(\widetilde{S}) \leq \operatorname{fsLP}^*(\widetilde{S} - Q) + 2b\mu\varepsilon \operatorname{opt}(I) + O(1) \leq \operatorname{fsLP}^*(\widetilde{I})/\beta + 2b\mu\varepsilon \operatorname{opt}(I) + O(1). \quad \blacktriangleleft$$

▶ **Lemma 7.** *Let $\widetilde{S}$ be as computed by* `rnaPack`(I, β, ε). *Then with high probability, we get* $\operatorname{fsopt}(\widetilde{S}) \leq \operatorname{fsopt}(\widetilde{I})/\beta + 2b\mu\varepsilon \operatorname{opt}(I) + O(1/\varepsilon^2)$.

Proof. When $\operatorname{span}(I)$ is very large compared to $1/\varepsilon^2$, we get

$$\begin{aligned}
\operatorname{fsopt}(\widetilde{S}) &\leq \operatorname{fsIP}^*(\widetilde{S}) + O(1) && \text{(by Lemma 17)} \\
&\leq \operatorname{fsLP}^*(\widetilde{S}) + O(1) && \text{(by Lemma 18)} \\
&\leq \operatorname{fsLP}^*(\widetilde{I})/\beta + 2b\mu\varepsilon \operatorname{opt}(I) + O(1) && \text{(by Lemma 21)} \\
&\leq \operatorname{fsopt}(\widetilde{I})/\beta + 2b\mu\varepsilon \operatorname{opt}(I) + O(1). && \text{(by Lemma 17)}
\end{aligned}$$

Otherwise, if $\operatorname{span}(I) \in O(1/\varepsilon^2)$, we get

$$\begin{aligned}
\operatorname{fsopt}(\widetilde{S}) &\leq \rho \operatorname{opt}(I) + O(1) && \text{(by structural theorem)} \\
&\leq \rho|\,\texttt{simplePack}(I)| + O(1) \\
&\leq \Theta(\operatorname{span}(I)) + O(1) \leq O(1/\varepsilon^2). && \text{(by Section 3)}
\end{aligned}$$

◀

▶ **Theorem 9.** *With high probability, the number of bins used by* `rnaPack`(I, β, ε) *is at most* $((\ln \beta)\mu + (\gamma\alpha\rho)/\beta + 2b(d_v + 1 + \gamma\alpha\mu)\varepsilon)\operatorname{opt}(I) + O(1/\varepsilon^2)$.

Proof. Let J_{LP} be the set of bins packed in the *randomized rounding of configuration LP* step (see line 4 in Algorithm 1 in Appendix C), J_D be the set of bins used to pack the discarded items $D \cap S$, J be the set of bins used to pack the rest of the items $S \setminus D$, and $\widetilde{J}$ be the set of bins used by `complexPack` to pack items in $\widetilde{S}$.

Then $|J_{\text{LP}}| \le T = \lceil (\ln \beta)\|\widehat{x}\|_1 \rceil \le (\ln \beta)\mu \operatorname{opt}(I) + O(1)$.

Now, we have $|J_D| \le b\lceil 2\operatorname{span}(D)\rceil \le 2b\varepsilon \operatorname{span}(I) + b \le 2b(d_v + 1)\varepsilon \operatorname{opt}(I) + b$. The first inequality follows from the property of `simplePack`, the second follows from C1.1 (Small Discard) and the last follows from Lemma 1. Finally,

$$\begin{aligned}
|J| &\le \gamma|\widetilde{J}| + O(1) && \text{(property of } \texttt{unround} \text{ (C4))}\\
&\le \gamma\alpha \operatorname{fsopt}(\widetilde{S}) + O(1) && \text{(property of } \texttt{complexPack} \text{ (C3))}\\
&\le \gamma\alpha \left(\operatorname{fsopt}(\widetilde{I})/\beta + 2b\mu\varepsilon \operatorname{opt}(I)\right) + O(1/\varepsilon^2) && \text{(by Lemma 7)}\\
&\le \gamma\alpha \left(\rho/\beta + 2b\mu\varepsilon\right)\operatorname{opt}(I) + O(1/\varepsilon^2)
\end{aligned}$$

Here, the last inequality follows from the structural theorem (C2), which says that $\exists(\widetilde{I}, D) \in$ `round`(I) such that $\operatorname{fsopt}(\widetilde{I}) \le \rho \operatorname{opt}(I) + O(1)$. Hence, the total number of bins is at most

$$|J_{\text{LP}}| + |J_D| + |J| \le \left((\ln \beta)\mu + \frac{\gamma\alpha\rho}{\beta} + 2b(d_v + 1 + \gamma\alpha\mu)\varepsilon\right)\operatorname{opt}(I) + O(1/\varepsilon^2). \qquad \blacktriangleleft$$

C.1 Error in Previous R&A Framework

Here we describe a minor error in the R&A framework of [35], and how it can be fixed.

We define $(\widetilde{I}, D)$ as an output of `round`(I) and for the residual instance S, we define $\widetilde{S}$ as the corresponding rounded items of $S - D$. Our proof of Lemma 7 relies on the fact that for any subset of rounded items, the span reduces by a factor of at least β if we restrict our attention to the residual instance. Formally, this means that for any $\widetilde{X} \subseteq \widetilde{I}$, we have

$$\mathbb{E}(\operatorname{span}(\widetilde{X} \cap \widetilde{S})) = \sum_{i \in \widetilde{X}} \operatorname{span}(i) \Pr(i \in \widetilde{S}) \le \operatorname{span}(\widetilde{X})/\beta.$$

The equality follows from linearity of expectation and the fact that $\operatorname{span}(i)$ is deterministic, i.e., it doesn't depend on the randomness used in the randomized rounding of the configuration LP. This is because `round` is not given any information about what S is. The inequality follows from Lemma 16, which says that $\Pr(i \in S) \le 1/\beta$.

The R&A framework of [35] used similar techniques in their analysis. In their algorithm, however, they round items differently. Specifically, they define a subroutine `round` and define $\widetilde{I} :=$ `round`(I) and $\widetilde{S} :=$ `round`(S). They, too, claim that for any subset of rounded items, the span reduces by a factor of at least β if we restrict our attention to the residual instance. While their claim is correct for input-agnostic rounding (where items are rounded up to some constant size collection values chosen independent of the problem instance), the claim is incorrect for input-sensitive rounding (where the values are chosen based on the specific problem instance). So the claim is incorrect if `round` is not deterministic, as then an item can be rounded differently depending on different residual instances.

In fact, they use their R&A framework with the algorithm of Jansen and Prädel [27], which uses *linear grouping* (along with some other techniques) for rounding. Linear grouping rounds items in an *input-sensitive* way, i.e., the rounding of each item depends on the sizes of items in S which is a random subset.