Polynomial Approximation Schemes for Smoothed and Random Instances of Multidimensional Packing Problems

David Karger
MIT, CSAIL
karger@mit.edu

Krzysztof Onak
MIT, CSAIL
konak@mit.edu

Abstract

The multidimensional bin packing and vector bin packing problems are known to not have asymptotic polynomial-time approximation schemes (unless P = NP). Nevertheless, we show that:

- Any *smoothed* (randomly perturbed) instance, and any instance from a class of other distributions, does have a polynomial-time *probable* approximation scheme. Namely, for any fixed $\epsilon > 0$, we exhibit a linear-time algorithm that finds a $(1+\epsilon)$-approximate packing with probability $1-2^{-\Omega(n)}$ over the space of random inputs.
- There exists an oblivious algorithm that does not know from which distribution inputs come, and still asymptotically does almost as well as the previous algorithms. The oblivious algorithm outputs almost surely a $(1+\epsilon)$-approximation for every $\epsilon > 0$.
- For vector bin packing, for each considered class of random instances, there exists an algorithm that in expected linear time computes a $(1+\epsilon)$-approximation, for any fixed $\epsilon > 0$.

To achieve these results we develop a multidimensional version of the one-dimensional rounding technique introduced by Fernadez de la Vega and Lueker. Our results generalize Karp, Luby and Marchetti-Spaccamela's results on approximatibility of random instances of multidimensional bin packing to a much wider class of distributions.

1 Introduction

In the classical *bin packing* problem a finite set of items of sizes in the range $[0, 1]$ must be placed into the minimum number of bins. The only constraint is that the sum of the sizes of items in a single bin cannot exceed 1. This problem naturally generalizes to more than one dimension in at least two different ways. These two multidimensional problems are *multidimensional bin packing* and *vector bin packing*. In both, we assume that we have a constant dimensionality $d \geq 2$, and all items are vectors in $[0, 1]^d$. As before, we want to place the items into the minimum number of bins. The problems differ on the criterion whether a given set of items fits into a single bin:

- In the multidimensional bin packing problem, we treat items as d-dimensional boxes (rectangular parallelepipeds), and bins as unit cubes. A set of boxes fits into a single bin if and only if we can place boxes in the bin so that their interiors are disjoint. (Note that this definition implies that the boxes can touch each other.) Moreover, we assume that we cannot rotate boxes, all their faces must be parallel to faces of bins, and their orientation is determined by the order of coordinates in their size-vectors.

- In vector bin packing, all items are d-dimensional vectors, and a set of vectors fits into a single bin if and only if the sum of the vectors does not exceed 1 in any dimension.

Both of these problems have been proven NP-hard, using typical reductions that carefully construct input instances of very specific structure. Arguably, such specific constructs are "pathological" and unlikely to arise in practice. And indeed, many NP-hard problems seem to be very easy to solve when they are encountered in practice, presumably because these pathological hard instances do not occur. How can develop a theory that predicts the behavior or guides the development of algorithms in practice?

One popular approach has been *average case analysis*, showing that an input instance selected uniformly at random can be solved, say, in polynomial expected time (Dyer and Frieze [6]). A major objection to this approach is that uniformly random instances are "too easy", reflecting *none* of the complex structure that arises in practice and makes the problems challenging.

Spielman and Teng [12] explored a fascinating middle ground. To understand the excellent behavior of the simplex algorithm for linear programming in practice, they developed the idea of *smoothed analysis*. They showed that if *any* input instance were given a small, random perturbation, the resulting problem was probably easy for simplex. Intuitively, a small perturbation of an arbitrary instance should preserve much of the interesting structure of that instance, while simultaneously doing away with any truly pathological features of a single instance.

In this paper, we take a similar smoothed analysis perspective on NP-hard problems. Multidimensional bin packing has been proven not to have polynomial time approximation schemes [2]. We show, however, that *any* given instance with any small, random perturbation *does* have an approximation scheme with high probability.

We assume throughout the paper that $d \geq 2$ is fixed.

1.1 Basic definitions. We consider the following classes of random instances:

1. A *D-smooth instance* of n vectors is (a random variable representing) a multiset of n vectors chosen independently at random where the i-th vector, $1 \leq i \leq n$, is chosen according to ϱ_i, where each $\varrho_i : [0,1]^d \to \mathbb{R}_{\geq 0}$ is a probability density function on $[0,1]^d$ bounded by some real $D \geq 1$.

   Note that this definition includes instances created as follows. We take an arbitrary deterministically specified instance, and independently add noise to each vector, restricting it so that the size of this vector remains inside the $[0,1]^d$ cube. For typical notions of noise (Gaussian distortion or uniform distribution on some neighborhood of the original value) it turns out that each vector is drawn according to some probability density function bounded by D, where D is the same for all instances.

2. A *ϱ-random instance* of n vectors is (a random variable representing) a multiset of n vectors chosen independently at random according to ϱ, where $\varrho : [0,1]^d \to \mathbb{R}_{\geq 0}$ is a probability density function on $[0,1]^d$. We do not assume that ϱ is bounded.

Below, we use *with very high probability* to mean with probability $1 - 2^{-\Omega(n)}$ for input instances of size n.

We say that a deterministic algorithm for an optimization problem P is a *probable α-approximation algorithm* for a class of distributions, if for each distribution in the class it outputs an α-approximation with very high probability, where the probability is taken over random inputs.

We say that a family $\{\mathcal{A}_\epsilon\}_{\epsilon>0}$ is a *polynomial-time probable approximation scheme* (PTPAS) for a class of distributions if for each ϵ, A_ϵ is a probable $(1+\epsilon)$-approximation algorithm

1.2 Our results. We focus on multidimensional bin packing. Let S be a multiset of items. Denote by $\mathrm{OPT}(S)$ the size of the optimum packing of S, and by $\mathrm{Vol}(S)$ the total volume of items in the multiset S of items. Note that the following lower bound holds for any instance of multidimensional bin packing: $\mathrm{Vol}(S) \leq \mathrm{OPT}(S)$.

Let D be a fixed arbitrary positive real not less than 1. We show that the class of D-smooth instances has a PTPAS $\{\mathcal{A}_\epsilon\}_{\epsilon>0}$ such that the running time of each $\mathcal{A}_\epsilon$ is linear in n, the number of items. Moreover, for any constant $\epsilon > 0$, for instances S_n of n items, we have

$$\mathbb{E}\left[\frac{|\mathcal{A}_\epsilon(S_n)|}{\mathrm{OPT}(S_n)}\right] \leq 1 + \epsilon + o(1),$$

where again the expectation is over random instances, not the running of the algorithm. Furthermore, for each fixed fixed probability density function ϱ on $[0,1]^d$, we show that the same results hold for the class of ϱ-random instances.

We also present an oblivious algorithm $\mathcal{A}_\star$, that for any constant $\epsilon > 0$ and any of the considered classes of random instances (with fixed respectively D or ϱ) constructs in linear time a packing such that

$$\frac{|\mathcal{A}_\star(S_n)|}{\mathrm{OPT}(S_n)} \leq 1 + \epsilon$$

with probability $1 - 2^{-\widetilde{\Omega}(n)}$. The oblivious algorithm, contrary to the previous algorithms, does not know from which class of distributions inputs come, but still for any fixed class of distributions, it holds that

$$\mathbb{E}\left[\frac{|\mathcal{A}_\star(S_n)|}{\mathrm{OPT}(S_n)}\right] \to 1 \qquad \text{as } n \to \infty.$$

All results that we have presented so far hold also for vector bin packing. Moreover, for vector bin packing we will show that for any considered class of random instances, and for any $\epsilon > 0$, there exists an algorithm that in expected linear time finds a $(1+\epsilon)$-approximate packing, where the expectation is taken over random instances.

A main role in our algorithms is played by a multidimensional rounding technique. It is based on partitioning a set of large items into small multisets such that all elements in each of them are smaller then all elements in another. Using this fact, we round smaller items to larger ones, reducing the number of distinct types of items, at which point we are able to compute an exact optimum solution essentially by brute force, and then replace larger items by the corresponding smaller, achieving a packing of the original set.

1.3 Related work. One-dimensional bin packing has been studied extensively. Fernandez de la Vega and Lueker [7] gave the first asymptotic polynomial-time approximation scheme. They introduced a rounding technique that allowed them to reduce (at a cost of epsilon times optimum) the problem of packing large items to finding an optimum packing of just a constant number of different items. Their algorithm was later improved by Karmarkar and Karp [9], who gave an algorithm constructing a packing of size at most

$\mathrm{OPT}(S) + \log^2 \mathrm{OPT}(S)$, for an input S. Bansal *et al.* [2] gave an asymptotic polynomial-time approximation scheme for hypercube packing. Hypercube packing is a special case of multidimensional bin packing in which each item has all sides equal.

For 2-dimensional vector bin packing packing, Woeginger [13] proved that there is no asymptotic polynomial time approximation scheme. Chekuri and Khanna [4] showed an $O(\log d)$-approximation algorithm that runs in polynomial time for fixed d. Bansal *et al.* [1] have recently improved this result, showing an $(\ln d + 1 + \epsilon)$-approximation algorithm for any $\epsilon > 0$.

For multidimensional bin packing Bansal and Sviridenko (see [2] for a journal version) showed that the problem does not admit an asymptotic polynomial-time approximation scheme, assuming that $\mathrm{P} \neq \mathrm{NP}$. On the positive side, the best known approximation factor for 2-dimensional bin packing is due to Bansal *et al.* [1], who showed a $(\ln \Pi_\infty + 1 + \epsilon)$-approximation algorithm for any $\epsilon > 0$, where $\Pi_\infty = 1.691\ldots$ In addition, Correa and Kenyon [2] also considered the 2-dimensional case, and showed that we can pack rectangular items into the optimal number of square bins, if we allow slight enlargement of bins. For each $\epsilon > 0$, they presented a polynomial time algorithm that packs rectangles into $\mathrm{OPT}(S)$ square bins, each of size $(1+\epsilon) \times (1+\epsilon)$.

Probabilistic analysis of one-dimensional bin packing has a long history (see the survey by Coffman *et al.* [5] for details). Karp *et al.* [10] considered also multidimensional bin packing for random instances. They assume that an instance S is drawn from the uniform distribution (or some simple generalization of it), and in this setting they are able to show a polynomial-time algorithm such that the expected wasted space in the packing it constructs is of order $\widetilde{O}\left(n^{(d-1)/d}\right)$. One can show that it follows that for their specific distributions their algorithm $\mathcal{A}$ has the property that

$$\mathbb{E}\left[\frac{|\mathcal{A}(S_n)|}{\mathrm{OPT}(S_n)}\right] \to 1.$$

We prove a similar result for much more general classes of distributions, in particular, we show that it holds for smoothed instances. Unfortunately, one cannot prove that the expected wasted space is sublinear for as vast class of distributions as we consider. Suppose that we draw all items independently from the uniform distribution on $\left[\frac{4}{6}, \frac{5}{6}\right]^d$. Clearly, we can place at most one item chosen according to this distribution in a single bin, and therefore the expected waste is greater than $\left(1 - (5/6)^d\right) n$.

Spielman and Teng [12] introduced *smoothed analysis*, which is a hybrid of worst-case and average-case analysis. They tried to explain the good behavior of the simplex algorithm in practice, and showed that for any instance with every value with added Gaussian noise (which is often true, since in the real world we encounter measurement errors) the expected running time of the simplex algorithm is polynomial.

Here we also consider behavior of algorithms on perturbed instances. For both considered bin packing problems, we exhibit algorithms of a linear running time and of a low expected approximation ratio. Moreover, we show for vector bin packing (but not for multidimensional bin packing) that for any considered class of random and smoothed instances, we can always get an arbitrary low approximation ratio in an expected polynomial running time.

1.4 Our approach. Our approach builds on that of Fernandez de la Vega and Lueker for the one-dimensional bin-packing problem. Increasing dimension creates various obstacles to applying their approach, which we surmount. Their approach has three key components. Here, we review those three components and their limitations, and explain how we generalize them for the higher-dimensional case.

The first observation of Fernandez de la Vega and Lueker is that a certain instance of *restricted bin packing* (RBP) can be solved exactly. This is the case where all item sizes exceed some minimum constant δ, and the number of distinct item sizes is at most some other constant k. In this case, only a constant number $(1/\delta)$ of items can go into a bin, so there are only a constant number of distinct bin "types" that make up the optimal solution, described by the number of items of each size in the bin. It is therefore possible to write an integer linear program for the the bin packing problem, with one variable of each bin type representing the number of bins of that type in the optimal solution, and one constraint for each item size enforcing that the chosen set of bins contains in total the correct number of items of each size. This integer program can be solved in time $f(k,\delta)\,\mathrm{polylog}(n)$, for some function f. Therefore, for fixed k and δ we can solve this problem in time sublinear in n.

This portion of their approach survives nearly unchanged in our approach. We generalize RBP to higher dimensions, requiring that items sizes exceed a constant minimum length *in every dimension* and that there be only a constant number of distinct sizes of items. As before, we can then argue that there are only a constant number of multidimensional bin "types" in the optimal solution, and that the right number of each type can be found using an integer linear program. We must address one detail. In one dimension, deciding whether a given bin type (number of items of each size) is legal simply requires summing the sizes of the items. In higher dimensions, it is not immediately obvious how to perform a similar determination, since there are many degrees of freedom in how the items might be placed in a bin. We show, however, using ideas of Bansal *et al.* [2], that if a set of items fits in a bin, then they can be placed in a certain canonical position (with items touching other items on most sides). We argue that the number of such canonical placements is finite, so that we can decide which bin types are legal by enumerating all possible placements (recall that the number of items being placed, and thus the number of canonical arrangements, is a "constant" depending only on k and ϵ).

The second observation of Fernandez de la Vega and Lueker has to do with the small items (size less than ϵ). They observe that once the large items have been placed, it is possible to place the small items greedily, without significantly affecting the quality of the approximation (for if a given small item does not fit in an already-existing bin and must open a new bin, then all the existing bins must be nearly full, meaning that the solution is very close to optimal). This approach fails in higher dimensions, because an item can be small in one dimension but large in others. Such an item cannot be treated as small, and placed greedily, because it might not fit in any bin even if the bins are mostly empty due to bad usage in other dimensions. At the same time, such an item cannot be treated as large and handled in the RBP instance, because having one small dimension allows many to be placed in a single bin, violating the RBP solution's reliance on fitting only a constant number of items in each bin.

We wipe this problem away with randomization. We observe that in any "smoothed" instance of the sort described earlier, most items will be large in every dimension. This gives us a large lower bound on the optimum size, and also a small upper bound on the number of trouble-making items. Thus, we can afford to give each item that is small in any dimension its own bin, without significantly affecting the approximation ratio. This step is susceptible to the same critique as the analysis of *uniform* random instances: that randomization destroys all interesting and challenging structure in the problem. However, we will see below that the large items still have hard, interesting structure that is not erased by smoothing, so that smoothed analysis can suggest more about how to approach real-world problems.

The final and most sophisticated contribution of Fernandez de la Vega and Lueker is a *rounding* technique that they use to reduce any instance without small items (which can be set aside and handled at the end, as discussed above) to an RBP instance with only a constant number of item sizes, without significantly affecting the value of the solution. Essentially, they aim to "enlarge" each item to one of k canonical sizes. Since each item is enlarged, a solution to the new problem can immediately be transformed into a solution to the original without violating feasibility, simply by shrinking each item back to its original size. The challenge is to do this enlargement in such a way that the value of the optimum solution does not change a great deal. To do so, they order the items in decreasing sizes $s_1, \ldots, s_n$ and then break this order to a constant number of equal-sized *intervals* $(s_1, \ldots, s_m)$, $(s_{m+1}, \ldots, s_{2m})$, etc. They then argue that each item in a given interval can be enlarged to the size of the smallest item in the previous interval, without significantly changing the optimum solution. This follows from the fact that starting from the optimum solution, we could imagine moving each s_i into the space occupied by item s_{i-m} and *then* enlarging s_i to the desired size, which is at most s_{i-m}. This provides a location for each enlarged item except for those in the largest interval; however, so long as each interval contains a sufficiently small fraction of the total number of items, there are few enough in the first interval that each of them can get its own bin.

We generalize this rounding scheme to higher dimensions, by dealing with the following problem. In the one-dimensional case, there is an obvious total order on the items, and any small item will "fit" in the space occupied by a larger item. In higher dimensions, however, different items may be incomparable, with neither fitting inside the other, so it is not obvious how to create "intervals" whose items can be promoted into the "smallest" (which may not even be defined) item in the next interval. We address this problem by showing that although there is no total order, we can partition the items into a *set* of *partial* orders, each of which is close enough to a total order to allow us to do the rounding trick.

Consider the two-dimensional problem. The two numbers defining the size of each item can be represented by points in the unit square. Divide that unit square into a grid/checkerboard, and observe that all items in a particular cell of the grid will fit inside any item that is in the cell directly above and to the right (see Figure 2). More generally, we can think of the cells on a given diagonal of the grid as forming a *semichain*—items may be incomparable to items in the same cell, but will be comparable to items in all other cells of the diagonal. We can therefore hope to apply the "rounding trick"—increasing the size of all of the items in a given cell to become that of a particular item in the next large cell on the diagonal. This reduces the number of distinct item sizes to the number of cells, creating an RBP instance that we can solve as discussed above.

To make this work, we need to again invoke the smoothness of the distribution of sizes. In particular, note that every item must be bijectively paired with (promoted to the place of) another, larger item that acts as a witness that the smaller item can be enlarged to the canonical size. If there are many items within a single cell, there might not be enough larger items in the cells above on the diagonal to receive all the necessary roundings. However, if the distribution is smooth, then for a fine enough grid, it is highly unlike that many items will all be in the same cell. This lets us complete the argument.

2 Optimal packing of large items

The constructive proof of the following lemma which is a trivial extension of a lemma by Bansal *et al.* [2] shows us how to find an optimal packing of items that are large in each dimension, i.e. have sides greater than some constant δ. The volume of each such item is at least δ^d, and therefore, only at most $1/\delta^d$ of them can be placed in a single bin. For constant δ and a constant number of distinct sizes of items, we get a constant number of bin types. The only remaining problem is how to check whether a given constant number of items can be packed into a bin. Bansal *et al.* [2] observed that it suffices to consider a constant number of canonical placements of each item in the bin.

LEMMA 2.1. (Bansal *et al.* [2]) *Let S be a multiset of n items of at most m distinct sizes and with each side length at least δ. There exists an algorithm* **PackLarge** *that for constant m and δ computes an optimum packing of S in $O(\text{polylog}(n))$ time.*

3 Multidimensional rounding

3.1 Useful definitions. We introduce a natural partial order on items. For items $g = (g_1, g_2, \ldots, g_d)$ and $h = (h_1, h_2, \ldots, h_d)$ we say that $g \sqsubseteq h$ if $g_i \leq h_i$, for each coordinate i. Less formally, $g \sqsubseteq h$ means that item g fits in item h, even when no rotation is allowed. If for items g and h it holds that $g \sqsubseteq h$, it means that in any packing that contains h, we can replace the larger h with the smaller g, and get a legal packing.

Now we recall for reference the notion of a chain cover.

DEFINITION 3.1. Let $\langle S, \sqsubseteq \rangle$ be a partially ordered multiset. A partition $C_1 \cup C_2 \cup \ldots \cup C_k$ of multiset S is a *chain cover* of S if each C_i is a chain, i.e. for each pair $a, b \in C_i$ at least one of $a \sqsubseteq b$ and $b \sqsubseteq a$ holds.

If a random instance had a small chain cover, we could use the original rounding trick by Fernandez de la Vega and Lueker, to each chain separately, but unfortunately, this is very unlikely. Nevertheless, we relax the notion of a chain, and achieve a structure, that

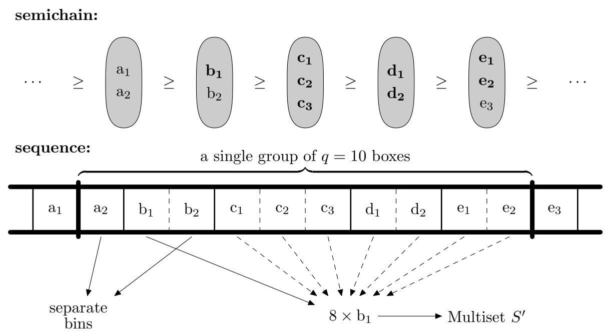

Figure 1: An example of the rounding. For each item either there is a corresponding greater or equal item in the new rounded instance, or the original item is packed separately.

we call a *semichain*, in which we are still able to conduct rounding. As we will see later, to be actually able to conduct rounding, we need a special parameter, that we call the *coarseness*, of this new structure to be small. For random instances, we can usually cover the set of items with a sufficiently small number of semichains of small coarseness.

DEFINITION 3.2. A *semichain* $T = (T_1, T_2, \ldots, T_k)$ is a set of disjoint nonempty subsets T_i of a partially ordered multiset such that if $t_1 \in T_i$, $t_2 \in T_j$, and $i < j$, then $t_1 \sqsubseteq t_2$. We call each T_i a *link*. We define the *coarseness* of T to be

$$\frac{\max_{1 \leq i \leq k} |T_i|}{\sum_{1 \leq i \leq k} |T_i|}.$$

In other words, a semichain is coarse if it has many of its items in one subset.

We also define a *semichain cover* as a partition of a partially ordered multiset S into pairwise disjoint nonempty semichains S_1, S_2, $\ldots$, S_k. The *coarseness* of the semichain cover is the maximum of coarsenesses of semichains S_i.

3.2 Reduction. Now we will present the key part of our paper, describing a multidimensional version of the rounding technique. We split each semichain into a constant number of groups, and in every group but the group of the largest items, we find a representative that is greater than most items in this group. Then we round up every item that is less than the representative of its group to this representative. If we find a packing of the rounded instance, we can replace every rounded item by its smaller progenitor. The number of remaining, unrounded items is negligible, each item can be packed in its own bin, and we still get a $(1 + \epsilon)$-approximation. See Figure 1 for an example, and the proof of Lemma 3.1 for details.

LEMMA 3.1. *Let S be a multiset of at most n items with each side length not less than δ. Suppose that we are given a semichain cover of S by at most k semichains and of coarseness at most $\epsilon^2\delta^{2d}/18$. The problem of finding a $(1+\epsilon)$-approximation can be reduced in linear time to the problem of finding an optimal packing of at most n items of at most $3k/\epsilon\delta^d$ distinct sizes with each side length not less than δ.*

Proof. We construct a multiset S' of items that dominates S. For each semichain $T = (T_1, T_2, \ldots, T_k)$ in the semichain cover we do the following (see Figure 1). We order items of T as follows. First we add (in arbitrary order) items of T_k (the set of "largest" items), then in arbitrary order items of T_{k-1}, and so on. Let

$$q = \left\lceil \frac{|T|\epsilon\delta^d}{3} \right\rceil.$$

We split the sequence of items into groups of q items so that only the last group may be smaller. There are $\lceil |T|/q \rceil$ groups. For each group except the first, we take the first T_i, i.e. the one with the greatest index i, that is completely contained by this group, and we choose an arbitrary item $c \in T_i$. Then for each item d in this group that belongs to T_j for $j < i$, we add a copy of c to S'.

Note that for each item d that we replace by an item c in S', it holds that $d \sqsubseteq c$. Therefore, once we find an optimal packing of the restricted instance S', we can replace every item in this packing by the corresponding d, without breaking any constraint. Note also that each c is less than (can be placed inside of) all items belonging to the previous larger group in the sequence that we have created, and hence $\mathrm{OPT}(S') \le \mathrm{OPT}(S)$.

For how many items do we not add a canonical item c to S'? Denote this number by r, and recall that a link is a subset of which a semichain consists. We do not add a canonical item c for the whole first group (q items), and for at most two of the links C_i in each other group. That is in total for a given semichain T, we have at most

$$\begin{aligned} r &\le \text{(maximal size of a group)} \\ &\quad + (\#\text{groups} - 1) \cdot 2(\text{maximal size of a link}) \\ &\le q + \left(\left\lceil \frac{|T|}{q} \right\rceil - 1 \right) \cdot 2|T| \frac{\epsilon^2\delta^{2d}}{18} \\ &\le \left(\frac{|T|\epsilon\delta^d}{3} + 1 \right) + \frac{|T|}{q} \cdot |T| \frac{\epsilon^2\delta^{2d}}{9} \\ &\le \frac{|T|\epsilon\delta^d}{3} + 1 + \frac{3}{\epsilon\delta^d} \cdot |T| \frac{\epsilon^2\delta^{2d}}{9} + 1 = \frac{2|T|\epsilon\delta^d}{3} + 1 \end{aligned}$$

such items. Moreover, the product of the coarseness of T and the size of T is by definition at least 1. Since the coarseness of T is upper-bounded by $\epsilon^2\delta^{2d}/18$, we have

$$1 \le |T| \cdot \frac{\epsilon^2\delta^{2d}}{18} < \frac{|T|\epsilon\delta^d}{3},$$

and therefore,

$$r \le \frac{2|T|\epsilon\delta^d}{3} + 1 < \frac{2|T|\epsilon\delta^d}{3} + \frac{|T|\epsilon\delta^d}{3} = |T|\epsilon\delta^d.$$

Focus now on all items that we do not pack, using the optimum packing for S'. Summing over all semichains, we know that there are at most $|S|\epsilon\delta^d$ of them. By packing each of them into a single bin we use at most

$$|S|\epsilon\delta^d \le \mathrm{Vol}(S) \cdot \epsilon \le \mathrm{OPT}(S) \cdot \epsilon$$

bins, as each item requires space of volume at least δ^d. This implies that we use at most

$$\begin{aligned} \mathrm{OPT}(S') + \epsilon\,\mathrm{OPT}(S) &\le \mathrm{OPT}(S) + \epsilon\,\mathrm{OPT}(S) \\ &= (1+\epsilon)\,\mathrm{OPT}(S) \end{aligned}$$

bins to pack S.

We only need to make sure that there are only a few item sizes in S'. There are at most

$$k \left(\left\lceil \frac{|C|}{T} \right\rceil - 1 \right) \le 3k/\epsilon\delta^d$$

of them. ■

Denote by **RoundAndPack**(T, ϵ) the following algorithm. Take a semichain cover T of the required coarseness of at most $\epsilon^2\delta^{2d}/18$, reduce the packing as described in Lemma 3.1, and run algorithm **PackLarge** from Lemma 2.1 on the instance consisting of a smaller number of items. Take the packing returned by **PackLarge**, and turn it into a $(1+\epsilon)$-approximate packing of T. Obviously, the running time of the algorithm is $O(n)$ plus the running time of **PackLarge**.

4 Packing smooth and random instances

4.1 A generic subroutine. We will start by presenting a generic subroutine, which summarizes the whole packing process. In next sections we will find out what parameters need to be passed to this subroutine for particular distributions.

The size of each item is described by a vector in $[0,1]^d$. Our subroutine partitions the cube $[0,1]^d$, the space of size-vectors. Figure 2, that presents a sample partition in the 2-dimensional setting, may be useful to understand what happens. The first step is to get rid of items that have some side smaller than some positive δ (the white area in the figure). We pack each of them into a separate bin. Next, we partition the

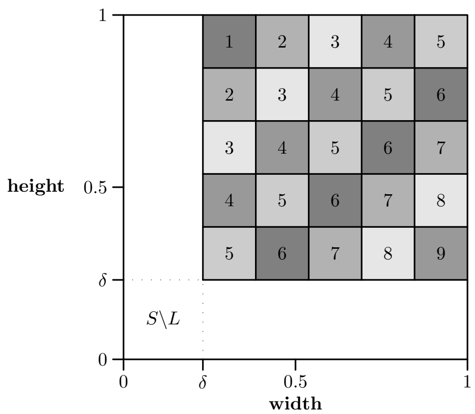

Figure 2: Partition of the space of size-vectors

remaining space of size-vectors into k^d cubes (in the picture we have $k = 5$, and 5^2 gray squares). Then if some cube contains more than P items, where P is another parameter to the subroutine, we get rid of them by packing each of them into a separate bin. Then we construct a semichain cover of the remaining items as follows. Items in each cube will constitute a link in a semichain. All cubes on a line parallel to the diagonal from $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$ contribute to the same semichain. In Figure 2 cubes that belong to the same semichain are labelled with the same number. Next, before we apply our multidimensional rounding to get a $(1+\epsilon)$-approximation, we get rid of semichains that have coarseness exceeding the $\epsilon^2\delta^{2d}/18$ demanded in Lemma 3.1, by packing each item in them into a separate bin. Then we run **RoundAndPack** algorithm on the remaining semichain covers. As one can see, we pack items in 4 steps in this procedure. A formal description of the subroutine follows.

Subroutine **GenericPacking**$(S, \delta, k, P, \epsilon)$:

1. Let L be the subset of items b in S such that $(\delta, \delta, \ldots, \delta) \sqsubseteq b$.

2. Pack each item in $S \backslash L$ (i.e. any item with some side smaller than δ) into a separate bin.

3. Let
$$C_{i_1,i_2,\ldots,i_d} = \prod_{j=1}^{d} \left[\delta + i_j \frac{1-\delta}{k}, \delta + (i_j+1)\frac{1-\delta}{k}\right],$$
where each i_j belongs to set $\{0, 1, \ldots, k-1\}$. These k^d cubes $C_{i_1,\ldots,i_d}$ cover the space of size-vectors of unconsidered items $[\delta, 1]^d$. Partition L into multisets $L_{i_1,\ldots,i_d}$, so that a multiset $L_{i_1,\ldots,i_d}$ contains items belonging to the cube $C_{i_1,\ldots,i_d}$, and if some item belongs to more than one of these cubes, assign it arbitrarily to one of them.

4. Pack items in multisets $L_{i_1,\ldots,i_d}$ such that $|L_{i_1,\ldots,i_d}| > P$, each into a separate bin.

5. Construct a semichain cover T from unpacked sets $L_{i_1,\ldots,i_d}$ as follows. Each such a set $L_{i_1,\ldots,i_d}$ will be a part of some semichain. If a semichain contains an unpacked set $L_{i_1,\ldots,i_d}$, then it contains all unpacked sets $L_{j_1,\ldots,j_d}$ such that $i_1 - j_1 = \ldots = i_d - j_d$ and only them.

6. Let T' be the semichain cover consisting of semichains in T of coarseness at most $\epsilon^2\delta^{2d}/18$. Pack the items from the remaining semichains, each into a separate bin.

7. **RoundAndPack**(T', ϵ)

4.2 Tools for analysis. In a few next lemmas we explore basic properties of random distributions. We will make use of them later, analysing our algorithms. We will skip formal proofs, which can be found in the full version of the paper.

Denote by X_ϱ a value chosen at random according to a probability density function ϱ. Our first lemma simply says that if a probability density function on $[0,1]^d$ is bounded, then the probability that a vector drawn according to this density function has a small coordinate is also small.

LEMMA 4.1. *Let ϱ be a probability density function on $[0,1]^d$ bounded by D. It holds that*
$$\mathbb{P}\left[X_\varrho \in [0,1]^d \backslash [\delta,1]^d\right] \le dD\delta.$$

Now we will show that the expected total volume of items grows linearly with very high probability. This comes from the fact that in any random model that we consider, there exists a positive constant such that with probability $1/2$ the volume of each item is greater than this constant, and on average at least half of items contribute with this value to the sum of volumes.

LEMMA 4.2. *If S_n is a D-smooth or ϱ-random instance of n items, for fixed D or ϱ respectively, there exists a constant $c > 0$ such that*
$$\mathbb{P}\left[\mathrm{OPT}(S_n) < cn\right] < 2^{-\Omega(n)}.$$
A constant c equal to at most $\frac{1}{4(2dD)^d}$ or $\delta_\star^d/4$, respectively, is such a good constant, where $\delta_\star > 0$ is such that $\mathbb{P}\left[X_\varrho \in [\delta_\star, 1]^d\right] \ge 1/2$.

The next lemma shows that there are only few items with at least one small dimension (less than some positive constant). Furthermore, by Lemma 4.2 we know that the the optimum packing size is with very high probability sufficiently large to overwhelm the number of these items, even if we pack each of them into a separate bin.

Lemma 4.3. *If S_n is a D-smooth or ϱ-random instance of n items, for fixed D or ϱ respectively, for each $\epsilon > 0$ there exists $\delta > 0$ such that*

$$\mathbb{P}\left[\left|S_n \backslash [\delta, 1]^d\right| > \epsilon \, \mathrm{OPT}(S_n)\right] < 2^{-\Omega(n)}.$$

A constant δ equal to at most $\epsilon/4(2dD)^{d+1}$ or such that

$$\mathbb{P}\left[X_\varrho \in [0,1]^d \backslash [\delta, 1]^d\right] \leq \frac{1}{8}\epsilon\delta_\star^d,$$

respectively, is such a good constant, where $\delta_\star$ is the $\delta_\star$ from Lemma 4.2.

The next lemma under a very technical formulation hides a simple fact. Suppose that we choose an element according to some probability density function. By the *average density* of a subset S of our probability space we mean the quotient of the probability that an element from S is drawn by the volume of S. For a fixed density function, taking D large enough, we can make the probability that an element belongs to a part of average density at least D arbitrarily small regardless of a partition that we have. This implies that for D large enough, the number of items in a part of large average density becomes with very high probability small, comparing it to the optimum packing size.

Lemma 4.4. *Let*

- *P_1, P_2, ..., P_k be measurable sets that constitute a partition of $[0,1]^d$,*
- *ϱ be a probability density function on $[0,1]^d$,*
- *$\delta_\star$ be the $\delta_\star$ from Lemma 4.2,*
- *η be such that $\mathbb{P}[\varrho(X_\varrho) \geq \eta] \leq \epsilon\delta_\star^d/16$,*
- *$P_\star$ be the union of P_i such that $\mathbb{P}[X_\varrho \in P_i] \geq 2\eta \, \mathrm{Vol}(P_i)$,*
- *S_n be a ϱ-random instance of n items.*

With probability $1 - 2^{-\Omega(n)}$,

$$\frac{|S_n \cap P_\star|}{\mathrm{OPT}(S_n)} \leq \epsilon.$$

4.3 D-Smooth instances. First, we present a PTPAS for D-smooth instances. An algorithm takes a multiset S of items and constants D and ϵ. Note that it passes to subroutine **GenericPacking** $P = \infty$. We may skip Step 4 of the subroutine, because in D-smooth instances the average density of a set is always bounded by D, and it is very unlikely that some cube contains too many items.

Algorithm **PackDSmooth**(S, D, ϵ)

$$\equiv \quad \textbf{GenericPacking}\left(S, \delta, \left\lceil \frac{12(2dD)^{d+1}}{\epsilon g} \right\rceil, \infty, \frac{\epsilon}{3}\right)$$

for $\delta = \frac{\epsilon}{24dD(2dD)^d}$ and $g = \frac{\epsilon^2\delta^{2d}}{162}$.

Now we will show that the algorithm has the required properties. Using previous lemmas, one can easily show that with very high probability we are not much worse then the optimum solution at each step.

Theorem 4.1. *Algorithm* **PackDSmooth** *is a PTPAS for the class of D-smooth instances. This implies in turn that for fixed D for D-smooth instances S_n of n items*

$$\mathbb{E}\left[\frac{|\textbf{PackDSmooth}(S_n, D, \epsilon)|}{\mathrm{OPT}(S_n)}\right] \leq 1 + \epsilon + o(1).$$

Proof sketch. We only pack items in Steps 2, 6 and 7 of **GenericAlgorithm**, since taking $P = \infty$, we skip Step 4. We will show that the ratios of the sizes of these packings to the optimal packing sum with very high probability to the required ratio.

By Lemma 4.3 with very high probability in Step 2 we have

$$\frac{|S \backslash L|}{\mathrm{OPT}(S_n)} \leq \frac{\epsilon}{3},$$

and by Lemma 4.2, also with very high probability, it holds that $OPT(S_n)$ is linear in n.

Because the probability density functions are bounded, with very high probability each cube contains small number of items. We want the number of items in semichains that we pack in Step 6 to be small. Obviously, if sets $C_{i_1,i_2,\ldots,i_d}$, are small, and a semichain contains many items, then its coarseness is less than g. We can show that, if a semichain is packed in step 6, then with very high probability, it contains few items. Since there are less than dk^{d-1} semichains, we can show that with very high probability

$$\frac{\#\text{items in } T \backslash T'}{\mathrm{OPT}(S_n)} \leq \frac{\epsilon}{3}.$$

In Step 7 we pack the remaining items, that constitute semichain cover T'. The coarseness of T' is as small as it is required to get a $(1 + \epsilon/3)$-approximation of the optimal packing of T'.

In total, we get with very high probability a packing that uses at most $(1+\epsilon)\,\mathrm{OPT}(S_n)$ bins. If this does not hold, we still have

$$\frac{|\textbf{PackDSmooth}(S_n, D, \epsilon)|}{\mathrm{OPT}(S_n)} \leq \frac{n}{1} = n,$$

which yields

$$\mathbb{E}\left[\frac{|\textbf{PackDSmooth}(S_n, D, \epsilon)|}{\mathrm{OPT}(S_n)}\right] \leq (1+\epsilon) + n2^{-\Omega(n)} = 1 + \epsilon + o(1).$$

By Lemma 2.1 the algorithm works in linear time for fixed d, D, ϵ. ∎

4.4 ϱ-Random instances. Now we feed the generic subroutine with parameters, constructing a PTPAS for ϱ-random instances. This time much deeper knowledge about our inputs than just a single real number is required.

Algorithm **PackRhoRandom**(S, ϱ, ϵ)

$\equiv$ **GenericPacking** $\left(S, \delta, k, \frac{4Qn}{k^d}, \frac{\epsilon}{4}\right)$, where

- $\delta_\star > 0$ is such that $\mathbb{P}\left[X_\varrho \in [\delta_\star, 1]^d\right] \geq \frac{1}{2}$,
- $\delta > 0$ is such that $\mathbb{P}\left[X_\varrho \in [0,1]^d \backslash [\delta, 1]^d\right] \leq \frac{1}{32}\epsilon\delta_\star^d$,
- Q is such that $\mathbb{P}\left[\varrho(X_\varrho) \geq Q\right] \leq \frac{1}{64}\epsilon\delta_\star^d$,
- $g = \frac{1}{288}\epsilon^2\delta^{2d}$,
- $k = \left\lceil \frac{64dQ}{g\epsilon\delta_\star^d} \right\rceil$.

It can be shown that the algorithm **PackRhoRandom** is the required PTPAS. The proof goes along the same lines as the proof of Theorem 4.1. The only significant difference is that now we may pack some items in Step 4 of **GenericPacking**. But their number can be bounded by using Lemma 4.4.

4.5 An oblivious algorithm. One can show that there exists an oblivious algorithm, than does not know any of the properties of instances that we have so far made use of. This algorithm achieves for a fixed $\epsilon > 0$ almost the same properties as the previous algorithms, and is even better in the asymptotic sense, since the expected ratio of the size of the packing that the oblivious algorithm outputs to the optimum packing size converges to 1 as the number of items goes to ∞.

The whole idea lies in slow improvement of parameters that we pass to the generic subroutine. Each parameter slowly changes with growing n. Kannan [8] showed that an integer linear program can be solved in $N^{O(N)}\,\mathrm{poly}(L)$ time, where N is the number of variables, L is the size of the program description, and $\mathrm{poly}(L)$ is a polynomial in L. Using this result, one can extend Lemma 2.1 and show that a multiset of n items of at most m distinct sizes and with each side length at least δ can be optimally packed in time

$$2^{(M+m)^{O(M)}}(\log n)^{O(1)},$$

where $M = 1/\delta^d$.

Algorithm **ObliviousPacking**(S)

$\equiv$ **GenericPacking** $\left(S, \frac{1}{\log^{(3)} n}, k, \frac{n\log^{(3)} n}{k^d}, \frac{1}{\log^{(3)} n}\right)$,
where $k = \lceil \log^{(2)} n \rceil$.

One can show that this algorithm runs in linear time, and that for any fixed considered class of random instances

$$\mathbb{E}\left[\frac{|\textbf{ObliviousPacking}(S_n)|}{\mathrm{OPT}(S_n)}\right] \xrightarrow{n\to\infty} 1.$$

5 Other packing problems

5.1 Multidimensional bin packing with rotations. Multidimensional bin packing *with rotations* is a variant of multidimensional bin packing where rotations of items by 90 degrees in each plane determined by any two dimensions are allowed. Note that all our techniques easily apply in this setting. We only need to consider each of possible $d!$ rotations of every item in subroutine **PackLarge**.

5.2 Vector bin packing. Note that all techniques that work for multidimensional bin packing can be applied to the vector bin packing problem. One only needs to work out details that differ the problems, but the general approach remains the same. For instance, volumes of items in multidimensional bin packing can be replaced with lengths of vectors in L_1-metric, and if we denote the sum of this lengths for a multiset S of vectors by $\mathrm{Len}(S)$, the following inequality holds:

$$\frac{\mathrm{Len}(S)}{d} \leq \mathrm{OPT}(S).$$

Therefore, all the results that we have proven for multidimensional bin packing also hold for vector bin packing.

Vector bin packing turns out to be even easier. For instance, we can easily check whether an arbitrary set of items fits into a bin. We sum the corresponding vectors, and check if their sum is at most 1 in each coordinate. Furthermore, an optimal vector packing can always be computed in $2^{O(n)}$ time. Note that there are at most 2^n subsets of an input multiset S of items. For each subset $A \subseteq S$ the minimum number $\mathrm{OPT}(A)$ of bins necessary to pack it is computed using dynamic programming. We consider subsets of an input multiset S in arbitrary ordering such that if $A \subset B \subseteq S$, then subset A is considered before subset B. Obviously, $\mathrm{OPT}(\emptyset) = 0$, and for any other subset B, we have

$$\mathrm{OPT}(B) = 1 + \min_{A \subset B \wedge A \in \mathcal{S}_1} \mathrm{OPT}(A),$$

where $\mathcal{S}_1$ is the set of nonempty subsets of S that fit into a single bin. Using this formula, we compute $\mathrm{OPT}(S)$, and reconstruct an optimal packing in time

$$2^n \cdot 2^n \cdot \mathrm{poly}(n) = 2^{2n+o(n)}.$$

As we already know, for each considered class of random instances, and for each $\epsilon > 0$, there exists a linear-time algorithm that with probability $1 - 2^{-\Omega(n)}$ over random inputs constructs a $(1+\epsilon)$-approximation. Moreover, it follows from the Chernoff bound, that by choosing appropriate parameters in the algorithm, the constant hidden by big-omega notation can be made arbitrary large. Specifically, this constant can be greater than 2, the constant in the exponent of the upper bound on the exact algorithm running time. The approximation algorithm can be easily modified to check if each required inequality holds. If at least one of them fails, the exact algorithm is run, and an optimum solution that it computes is returned. As we can see, the modified algorithm always returns a $(1+\epsilon)$-approximation. Moreover, its expected running time is still linear, since the probability that the exact algorithm is run surpasses the exact algorithm running time. This allows us to state the following theorem.

Theorem 5.1. *For D-smooth or ϱ-random instances of vector bin packing, for fixed D or ϱ respectively, there exists an algorithm that computes a $(1+\epsilon)$-approximation in expected linear time, where ϵ is an arbitrary positive constant.*

6 Conclusion

This paper has carried out a smoothed analysis of an NP-hard problem, showing that even adversarially constructed instances are susceptible to a polynomial time approximation scheme when given an arbitrarily small random perturbation. Such smoothed analysis seems a natural way to explain the "easiness" of some NP-hard problems in practice. Several criticisms can be leveled at our approach, each suggesting future work:

- Once we incorporate random perturbations, there is no proof that the resulting problem is NP-hard. Perhaps there is an *exact* algorithm with (probable) polynomial running time in the smoothed model?

- As we noted, our random perturbation essentially wipes out all items with a tiny length in one dimension. Arguably, this is erasing important problem structure. While our smoothing is surely more realistic then a uniform random distribution (at least in its treatment of large items), perhaps there are other, better perturbations that preserve more structure—for example, a perturbation of lengths by a *multiplicative* rather than an *additive* factor?

Even in light of these criticisms, it seems natural to consider smoothed analysis for many other NP-hard problems as well.

References

[1] N. Bansal, A. Caprara, and M. Sviridenko. Improved approximation algorithms for multidimensional bin packing problems. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, 2006. To appear.

[2] N. Bansal, J. R. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operation Research*, 31(1):31–49, 2006.

[3] A. Caprara. Packing 2-dimensional bins in harmony. In *Proceedings of the 43th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 490–499, 2002.

[4] C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM Journal on Computing*, 33(4):837–851, 2004.

[5] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Co., Boston, MA, USA, July 1996.

[6] M. E. Dyer and A. M. Frieze. The solution of some random NP-hard problems in polynomial expected time. *Journal of Algorithms*, 10(4):451–489, 1989.

[7] W. Fernandez de la Vega and G. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[8] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.

[9] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1982)*, pages 312–320, 1982.

[10] R. M. Karp, M. Luby, and A. Marchetti-Spaccamela. A probabilistic analysis of multidimensional bin packing problems. In *Proceedings of the 16th Annual ACM Symposium on the Theory of Computing (STOC 1984)*, pages 289–298, 1984.

[11] H. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[12] D. A. Spielman and S.-H. Teng. Smoothed analysis: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[13] G. J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters*, 64(6):293–297, 1997.