Approximation Schemes for Geometric Knapsack for Packing Spheres and Fat Objects

Pritam Acharya
Department of Mathematics, Indian Institute of Science Education and Research Pune, India

Sujoy Bhore
Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India

Aaryan Gupta
Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India

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

Bratin Mondal
Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India

Andreas Wiese
Department of Mathematics, Technical University of Munich, Germany

Abstract

We study the geometric knapsack problem in which we are given a set of d-dimensional objects (each with associated profits) and the goal is to find the maximum profit subset that can be packed non-overlappingly into a given d-dimensional (unit hypercube) knapsack. Even if $d = 2$ and all input objects are disks, this problem is known to be NP-hard [Demaine, Fekete, Lang, 2010]. In this paper, we give polynomial time $(1 + \varepsilon)$-approximation algorithms for the following types of input objects in any constant dimension d:

- disks and hyperspheres,
- a class of fat convex polygons that generalizes regular k-gons for $k \geq 5$ (formally, polygons with a constant number of edges, whose lengths are in a bounded range, and in which each angle is strictly larger than $\pi/2$),
- arbitrary fat convex objects that are sufficiently small compared to the knapsack.

We remark that in our PTAS for disks and hyperspheres, we output the computed set of objects, but for a $O_\varepsilon(1)$ of them we determine their coordinates only up to an exponentially small error. However, it is not clear whether there always exists a $(1 + \varepsilon)$-approximate solution that uses only rational coordinates for the disks' centers. We leave this as an open problem which is related to well-studied geometric questions in the realm of circle packing.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Approximation Algorithms, Polygon Packing, Circle Packing, Sphere Packing, Geometric Knapsack, Resource Augmentation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.8

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2404.03981`

Funding Arindam Khan's research is supported in part by Google India Research Award, SERB Core Research Grant (CRG/2022/001176) on "Optimization under Intractability and Uncertainty", and the Walmart Center for Tech Excellence at IISc (CSR Grant WMGT-23-0001). A part of this work was done when Aaryan Gupta, Bratin Mondal, and Pritam Acharya were undergraduate interns at Indian Institute of Science, supported by Arindam Khan's Pratiksha Trust Young Investigator Award.

© Pritam Acharya, Sujoy Bhore, Aaryan Gupta, Arindam Khan, Bratin Mondal, and Andreas Wiese;
licensed under Creative Commons License CC-BY 4.0

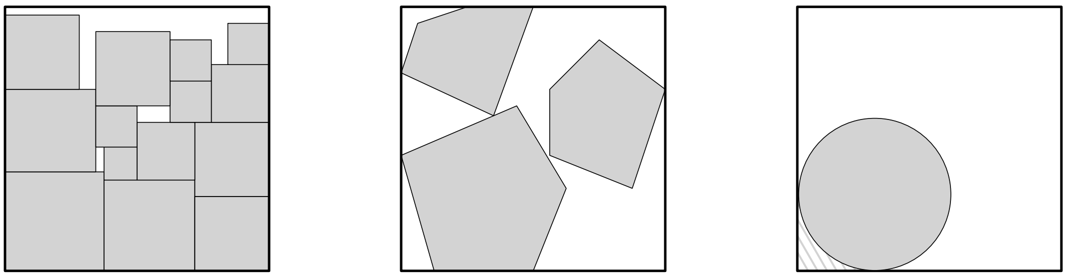

Figure 1 Left: The squares are stacked compactly inside the knapsack. Middle: The pentagons cannot be stacked as tightly inside the knapsack as the squares. Right: The space in the corner (striped area) cannot be covered by any large circle.

1 Introduction

One of the cornerstones of geometry is the problem of packing circles and spheres into a container, e.g., a square or a hypercube. It dates back to the 17th century when Kepler conjectured his famous bound on the average density of any packing of spheres in the three-dimensional Euclidean space [27]. The problem has been investigated, for example, by Lagrange [11] who solved it in the setting of two dimensions, by Hales and Ferguson [24] who proved Kepler's original conjecture, and by Viazovska [39] who studied the problem in dimension 8 and was awarded the Fields medal in 2022 for her work.

A natural corresponding optimization question is the *geometric knapsack problem*, where we are given a set of d-dimensional objects for some constant $d \in \mathbb{N}$, e.g., circles or (hyper-)spheres, but possibly also other shapes (like squares, pentagons, hexagons, etc. for the case of $d = 2$) with each of them having a given profit. The goal is to find the subset of maximum total profit that can be packed non-overlappingly into a given square or (hyper-)cube. In this work, we consider the translations of the objects but do not allow rotations.

Geometric knapsack is a natural mathematical problem and it is well-motivated by practical applications in several areas, including radio tower placement [38], origami design [30], cylinder pallet assembly [8, 17], tree plantation [38], cutting industry [38], bundling tubes or cables [40], layout of control panels [8], or design of digital modulation schemes [36].

The problem is known to be NP-hard, already for $d = 2$ and if all input objects are axis-aligned squares or disks [14, 2]. This motivates designing approximation algorithms for it. For hypercubes in any constant dimension d, there is a polynomial time $(1+\varepsilon)$-approximation algorithm known for any constant $\varepsilon > 0$ [26], i.e., a polynomial time approximation scheme (PTAS). Thus, this is the best possible approximation guarantee, unless P=NP.

However, for other classes of (fat) objects, the best known results either have approximation ratios that are (far) from their respective lower bounds or they require resource augmentation, i.e., increase in the size of the given knapsack. One intuitive reason for this is that axis-aligned squares and cubes can be stacked nicely without wasting space and the resulting coordinates are well-behaved, while for more general shapes this might not be the case, see Figure 1. For circles, the best known result is a $(3 + \varepsilon)$-approximation [35] but the best known lower bound is only NP-hardness. Still, the membership in NP is wide open for the following question: given a set of n circles of $O(1)$ number of different sizes, decide whether they can be packed into a unit square. It is also open whether packing circles into a square knapsack is $\exists\mathbb{R}$-complete or not [1]. Even for n unit circles. we do not know the exact value of the smallest size squares that can pack them. See [37] for the current status of upper and lower bounds for $n \leq 1000$.

There is a PTAS in any constant dimension d for this case, but it requires resource augmentation [9]. For triangles, there is a $O(1)$-approximation algorithm (assuming it is allowed to rotate the triangles arbitrarily) whose precise approximation ratio is not explicitly specified [33]. Also for this case, it is still possible that there is a PTAS. On the other hand, there are settings of geometric knapsack that do *not* admit a PTAS, e.g., axis-parallel cuboids in three dimensions [12].

Furthermore, in practical applications (e.g., loading cargo into a truck or cutting pieces out of raw material like cloth or metal) the objects do not necessarily all have the same shape. For example, Bennell and Oliveira [6] consider a mix of different shapes of objects (their primary objects are circles, rectangles, regular polygons, and convex polygons). However, the previous papers in the theoretical literature for geometric knapsack mostly assume that all input objects are of the same type, e.g., only squares, only circles, or only rectangles, etc. Thus, from a theoretical point of view, it is interesting to see how the problem behaves when the input objects might be of different types.

This raises the following natural question that we study in this paper:

What are the best approximation ratios we can achieve for the geometric knapsack problem, depending on the type of the input objects? For which type of objects does a PTAS exist?

1.1 Our contribution

In this paper, we present a polynomial time $(1+\varepsilon)$-approximation algorithm for geometric knapsack problem for packing d-dimensional spheres into d-dimensional hypercube knapsack, for any constant dimension $d \geq 2$. For spheres, there is a complication that possibly any (near-)optimal packing for a given instance require irrational coordinates. Therefore, our output consists of a set of spheres that can be packed non-overlappingly inside the given knapsack and whose profit is at least $(1+\varepsilon)^{-1}w(\text{OPT})$, where $w(\text{OPT})$ denotes the profit of the optimal solution OPT. Moreover, for all but at most $O_\varepsilon(1)$ spheres, our algorithm outputs the precise (rational) coordinates of the packing. [1] For the other $O_\varepsilon(1)$ spheres it outputs them up to an exponentially small error in each dimension. We remark that there are related packing problems for which it is known that irrational coordinates are sometimes necessary and that computing them is $\exists\mathbb{R}$-complete (and hence possibly even harder than NP-hardness) [1]. On the other hand, if we knew that there always exists a $(1+\varepsilon)$-approximate solution in which all coordinates are rational with only a polynomial number of bits, our algorithm would find such coordinates in polynomial time. We stress that our returned set of spheres is always guaranteed to fit into the given knapsack with appropriate (possibly irrational) coordinates but *without* resource augmentation.

Our second result is a polynomial time $(1+\varepsilon)$-approximation algorithm for the geometric knapsack problem for wide classes of convex geometric polygons. Our first result is a PTAS for a class of fat convex polygons which generalizes pentagons, hexagons, and regular k-gons for constant $k > 4$ (see Figure 2). Formally, we require for each polygon that the angle between any two adjacent edges is at least $\pi/2 + \delta$ for some constant $\delta > 0$ and that each polygon has a constant number of edges with similar lengths (up to a constant factor). Note that in contrast to many prior results, we allow that each input object has a different shape, e.g., with a different number of edges, different angles formed by them, and a different orientation. Also, the polygons may differ arbitrarily in size.

[1] The notation $O_\varepsilon(f(n))$ means that the implicit constant hidden in big-O notation can depend on ε.

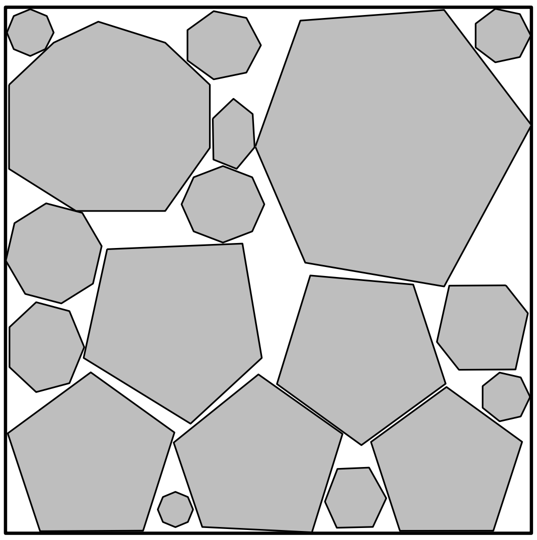

Figure 2 Packing of fat convex polygons in a knapsack.

If each input object is sufficiently small compared to the knapsack, we obtain even a polynomial time $(1+\varepsilon)$-approximation for *arbitrary* fat convex objects in any constant dimension d. We remark that for other packing problems like one-dimensional KNAPSACK or BIN PACKING, near-optimal solutions can easily be achieved via greedy algorithms if the input objects are sufficiently small. Even for sufficiently small d-dimensional axis-aligned hypercuboids, it is known that simple algorithms like NFDH [13, 4] has negligible wasted space. However, for other geometric objects this is much harder since we might not be able to place the input objects compactly without wasting space. For example, classical result by Thue [11] showed that one can pack at most $\frac{\pi}{2\sqrt{3}} \approx 0.9069$ fraction of the total area, even in the case of packing of unit circles. Furthermore, for circles and other similar convex objects, irrational coordinates may arise in the packing and the optimal solution may use a very complicated packing to minimize the wasted space.

1.2 Our techniques

We discuss now the techniques of our results, starting with our PTAS for spheres. To compute our packing, we first enumerate all the large spheres in the optimal solution, i.e., the spheres whose radius is at least a constant fraction of the side length of the knapsack. Also, we guess their placement up to a polynomially small error, which yields a small range of possible placements for each of them. Note that we cannot guess these coordinates precisely, since we cannot even exclude that they are irrational. However, we guarantee that such coordinates *exist*, by solving a system of polynomial equations *exactly* in polynomial time.

Next, we want to place small spheres into the remaining part of the knapsack. Unfortunately, we do not know precisely which part of the knapsack is available for them since we do not know the precise coordinates of the large spheres. Thus, there is some area of the knapsack that is *maybe* used by the large spheres in our packing; however, potentially, the optimal solution uses it for placing small spheres. Our key insight is that this area is small compared to the area that is for sure *not* used by large spheres in the optimal solution. Using the fact that objects are spheres, we show that some area in each corner of the knapsack cannot be covered by any large sphere in *any* solution and whose size is at least a constant fraction of the knapsack (see the bottom-left empty corner in Figure 1). We use this area to compensate the fact that we do not know the precise coordinates of our large spheres and we waste space because of this.

When we select and pack the small spheres, we define a constant number of (small) identical knapsacks that fit into the given knapsack together with the large spheres and into which we place our small spheres. For the remaining task of placing the small spheres, we argue that it is sufficient to have an algorithm that uses resource augmentation, i.e., that increases the size of each knapsack by a factor of $1+\varepsilon$ (in each dimension). Thus, on a high level, we reduce the problem of packing arbitrary spheres into *one* knapsack to the problem of packing small spheres into a *constant number* of knapsacks *with resource augmentation*.

This remaining problem can be solved via an algorithm in [9]; however, we present a more general routine that works even for arbitrary convex fat objects. Also, it is arguably simpler than the corresponding algorithm in [9]. On a high level, we prove that there is a well-structured solution based on a hierarchical decomposition of the knapsacks into grid cells. The grid cells are partitioned such that each placed object P is contained in a constant number of grid cells whose size is comparable to P. Importantly, these grid cells are used *exclusively* by P and not by any other placed object (not even partially). This allows us to devise a dynamic program (DP) that computes the optimal structured packing of this type. Our DP has a subproblem for each combination of a level (corresponding to a size range of the input objects) and a number of available grid cells corresponding to this level. Given such a subproblem, it suffices to enumerate a polynomial number of possibilities for selecting and placing objects of this level, which reduces the given subproblem to a subproblem corresponding to the next level. This DP might have applications in other related packing problems.

In our algorithm for fat convex polygons (with the properties described above), we extend our algorithm for spheres as follows. For the guessed large polygons, we compute their coordinates *exactly* in polynomial time. Here, we use the (known) fact that there exists a placement for them that corresponds to an extreme point solution of a suitable linear program, which has rational coordinates. Then, intuitively we use the condition for the polygons' angles to ensure that the large objects leave a certain area of the knapsack empty. We use this empty area in a similar way as in the setting of circles. Again, we place the small objects into a constant number of knapsacks under resource augmentation, using our new subroutine described above.

If all input objects are sufficiently small compared to the size of the knapsack (formally, we assume that each of them fits in a smaller knapsack with side length $\Theta(\varepsilon)$) there are no large objects and, hence, we can omit the step of enumerating them. In particular, we do not need the conditions of the polygons' edges anymore. Since the input objects are so small, we can show that by losing a factor of $1+\varepsilon$ in the approximation ratio, we may pretend that we have resource augmentation available. Hence, we can directly call our subroutine for small objects under resource augmentation.

We leave it as an open question to determine whether irrational coordinates are sometimes necessary for optimal or $(1+\varepsilon)$-approximate solutions for geometric knapsack for spheres. If yes, it would be interesting to determine the best possible approximation ratio one can achieve with rational coordinates only. Note that this question is related to the well-studied problem of determining the size of the smallest knapsack needed to pack a given number of unit circles. For that problem, it is known that for some number of unit circles the smallest knapsack has irrational edge lengths [18]. On the other hand, recall that if rational coordinates with a polynomial number of bits always suffice, our algorithm for spheres can compute the coordinates of *all* returned spheres of our $(1+\varepsilon)$-approximation algorithm *exactly*.

1.3 Other related work

For geometric knapsack for axis-parallel rectangles (i.e., when $d = 2$), the best known polynomial time algorithm has an approximation ratio of $17/9 + \varepsilon$ [19]. There is a pseudo-polynomial time algorithm with a ratio of $4/3 + \varepsilon$ [20] and a pseudo-polynomial time approximation scheme if we require guillotine-separable packing [28] . If it is allowed to rotate the rectangles by 90 degrees, there is also a polynomial time $(1.5 + \varepsilon)$-approximation algorithm known [19]. Moreover, the problem admits a QPTAS if the input data are quasi-polynomially bounded integers [3]. For the setting of packing circles, there is a PTAS under resource augmentation in one dimension, assuming that the profit of each circle equals its area, due to Lintzmayer, Miyazawa, and Xavier [31]. This was improved to the above mentioned PTAS under resource augmentation in one dimension for spheres with arbitrary profits in any constant dimension d, due to Chagas, Dell'Arriva, and Miyazawa [9]. In addition, there have been many attempts to develop heuristics and other optimization methods on circle packing, see e.g., [38, 25, 32].

A related problem is the geometric bin packing problem in which we want to place a given set of geometric objects into the smallest number of unit size bins. For the settings of squares or (hyper-)cubes [4] or skewed rectangles [29], the problem admits an asymptotic PTAS. In the case of general rectangles, the best known result is an asymptotic 1.405-approximation [5] but an asymptotic PTAS cannot exist unless $\mathsf{P} = \mathsf{NP}$ [12]. Maximum independent set in geometric intersection graphs [10, 34, 21] is another well-studied related problem.

In a recent paper, Abrahamsen, Miltzow, and Seiferth [1] developed a framework to show that for many combinations of allowed pieces, containers, and motions, the resulting packing problem is $\exists\mathbb{R}$-complete. For example, they showed that it is $\exists\mathbb{R}$-complete problem to decide if a set of convex polygons with at most seven corners each can be packed into a square if arbitrary rotations are allowed. However, it is not known if the setting of packing circles into a square knapsack is $\exists\mathbb{R}$-complete.

There is also a large body of work on questions about the optimal packings of unit circles into unit squares or equilateral triangles. We refer to [15, 16, 22, 25] for an overview.

1.4 Organization of this paper

In Section 2, we discuss the PTAS when the input items are sufficiently small fat convex objects. In Section 3, we give our algorithm for spheres. In Section 4, we consider the case of convex polygons. Finally, in Section 5 we end with conclusions. Due to space constraints, many proofs have been omitted. The corresponding lemmas and theorems are marked with $(\star)$. Please see the full version of the paper for the complete proofs.

2 PTAS under Resource Augmentation

In this section we present a PTAS when the input items are fat convex objects, and we are allowed to increase the size of the given knapsack by a factor of $1 + \varepsilon$ in each dimension. Chagas et al. [9] presented a PTAS for circles with resource augmentation in one dimension. Their result is based on a combination of multiple integer programs with variables for different configurations for packing parts of the given knapsack. Our result is arguably simpler and purely based on dynamic programming.

Let P_i be a two-dimensional convex object and let $r_i^{\text{out}}(P_i)$ and $r_i^{\text{in}}(P_i)$ be the radius of the smallest circle containing P_i and the radius of the largest circle contained in P_i, respectively. We will drop P_i when it is clear from the context. We say that P_i is *f-fat* if $r_i^{\text{out}}/r_i^{\text{in}} \leq f$ for some value $f \geq 1$. In the remainder of this section, we prove the following theorem.

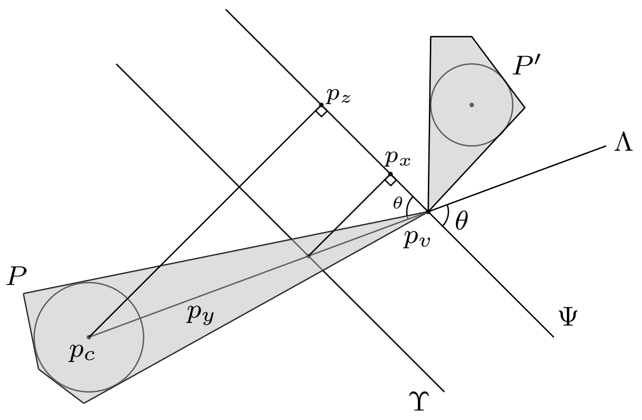

Figure 3 Line Ψ separates the two f-fat and convex objects P and P'. We construct line Υ such that it has intersects no common grid cells with line Ψ. We proceed to shrink the two objects P, P' by a factor of $1+\varepsilon$ such that they cannot intersect the space between lines Υ and Ψ.

▶ **Theorem 1.** *Let $f \geq 1$, $\varepsilon > 0$, and $d \in \mathbb{N}$ be constants. Given a set of d-dimensional f-fat convex input objects, there exists a polynomial time algorithm that can pack a subset of them with a total profit of $w(\mathsf{OPT})$ into a knapsack $K' := [0, 1+\varepsilon]^d$, where $w(\mathsf{OPT})$ is the optimal profit that can be packed into a knapsack $K := [0,1]^d$.*

For simplicity, we first describe our algorithm in the setting where $d = 2$. Given a packing of a set of f-fat objects $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ in our knapsack $K = [0,1] \times [0,1]$, we want to show that there is also a structured packing of these objects into an augmented knapsack $K' = [0, 1+\varepsilon] \times [0, 1+\varepsilon]$, defined via a discrete grid. Let $\delta_{cell} > 0$ be a constant to be defined later such that $1/\delta_{cell} \in \mathbb{N}$. We place a two-dimensional grid inside K' such that each grid cell has an edge length of δ_{cell}. Let $\mathcal{G}$ denote the set of all resulting grid cells. We assume first that each object $P_i \in \mathcal{P}$ is δ_{large}-*large*, meaning that $r^{in}(P_i) \geq \delta_{large}$ for some given constant $\delta_{large} > 0$. We say that our given packing for $\mathcal{P}$ is *discretized* if there is a partition of $\mathcal{G}$ into sets $\{\mathcal{G}_{P_1}, \mathcal{G}_{P_2}, \ldots, \mathcal{G}_{P_n}\}$ such that for each $P_i \in \mathcal{P}$, we have that P_i is contained in the union of the cells in $\mathcal{G}_{P_i}$. Therefore, each object $P_i \in \mathcal{P}$ has "its own" set of grid cells $\mathcal{G}_{P_i}$ that contain P_i and that do not intersect with any other object $P_j \in \mathcal{P} \setminus \{P_i\}$.

We show that for an appropriate choice of δ_{cell}, there is a discretized packing for $\mathcal{P}$ in K', i.e., if we can increase the size of our knapsack K by a factor of $1+\varepsilon$ in each dimension.

▶ **Lemma 2.** *For each $f \geq 1$, $\varepsilon > 0$, and $\delta_{large} > 0$, there is a value $\delta_{cell} > 0$ such that for any set of δ_{large}-large f-fat convex objects $\mathcal{P}$ that can be placed nonoverlappingly inside a knapsack $K = [0,1] \times [0,1]$, there is a discretized packing for $\mathcal{P}$ inside knapsack $K' = [0, 1+\varepsilon] \times [0, 1+\varepsilon]$ based on a grid in which each edge of each grid cell has a length of δ_{cell}.*

Proof. Let $P, P' \in \mathcal{P}$ be two f-fat convex δ_{large}-large objects packed inside the knapsack. Then by *separating hyperplane theorem for convex objects* [7], we know that there is a line Ψ containing a point p_v on the boundary of P, and Ψ separates P from P' (see Figure 3). Now, intuitively, increasing the size of the knapsack by a factor of $1+\varepsilon$ is equivalent to shrinking the objects in $\mathcal{P}$ by a factor of $1+\varepsilon$. So, we want to find the right constraints such that after shrinking P and P' do not share any grid cell. Let the center of the incircle (of radius r^{in}) contained in P be p_c and the line joining p_c and p_v be Λ. Let the foot of the image of the point p_c on line Ψ be the point p_z. Now the length of line segment $\overline{p_c p_z} := |\overline{p_c p_z}| \geq r^{in}$

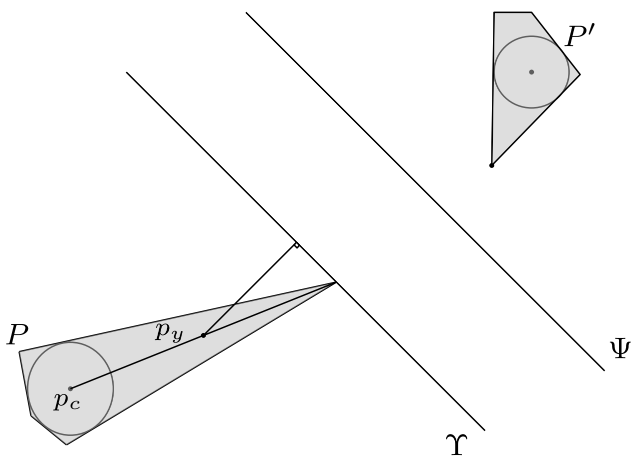

Figure 4 P, P' are shrunk so that they cannot intersect the space between the lines Υ and Ψ.

and $|\overline{p_c p_v}| \leq 2fr^{in}$, due to fatness. Hence, the angle θ between Ψ and Λ is at least $\sin^{-1}(\frac{1}{2f})$. Consider points p_x on Ψ and p_y on Λ such that $|\overline{p_x p_y}| = \sqrt{2}\delta_{cell}$ and the line Υ joining p_x, p_y is parallel to the line joining $p_c p_z$. Then any point on Ψ does not share a gridcell with any point on Υ. Also, $|\overline{p_y p_v}| \leq 2\sqrt{2}f\delta_{cell}$. Now we want to shrink P by $(1+\varepsilon)$ factor keeping p_c at the same position such that the shrunk version of P lies completely within one side of Υ (see Figure 4). After shrinking, $\overline{p_c p_v}$ gets smaller by $\varepsilon|\overline{p_c p_v}| \geq \varepsilon r^{in} \geq \varepsilon\delta_{large}$. Now we choose δ_{cell} such that $\delta_{cell} \leq \frac{\varepsilon}{2\sqrt{2}f} \cdot \delta_{large}$. Thus we satisfy $\varepsilon|\overline{p_c p_v}| \geq \varepsilon\delta_{large} \geq 2\sqrt{2}f\delta_{cell} \geq |\overline{p_y p_v}|$ and this ensures that the shrunk down version of P and P' do not share any grid cell. We assign each polygon to the grid cells that it intersects with. Hence this process leads to a discretization such that the no grid cell is intersected by two polygons. ◀

Next, we argue that there is also a structured packing for fat objects that are not necessarily all (relatively) large. Let $\delta_{large}, \delta_{cell}, \delta_{small} > 0$ be constants to be defined later (they will depend on ε which will denote the amount by which we increase the size of our knapsack). We place now a *hierarchical* two-dimensional grid with multiple levels. We define that the whole knapsack K is one grid cell of level 0 of side length $\delta_{c,0} := 1$. For each level $\ell \geq 1$, we define grid cells whose edges all have a length of $\delta_{c,\ell} := \delta_{cell}\delta_{c,(\ell-1)}$. Recursively, for each $\ell \geq 1$ we partition each grid cell of level $\ell - 1$ into $1/\delta_{cell}^2$ grid cells of level ℓ with side length $\delta_{c,\ell}$ each. Similarly as before, we want that there is a partition of the grid cells such that for each object $P \in \mathcal{P}$ there is a set of grid cells $\mathcal{G}_P$ that contain P and that are disjoint from the grid cells $\mathcal{G}_{P'}$ for each object $P' \in \mathcal{P} \setminus \{P\}$. Also, we want that all grid cells in $\mathcal{G}_P$ are of the same level, that their size is comparable to the size of P, and that the number of grid cells in $\mathcal{G}_P$ is bounded. To ensure this, we group the objects $P \in \mathcal{P}$ according to their respective values $r^{in}(P)$ which we use as a proxy for their sizes. Formally, we define $\delta_{small,0} := 1$ and for each level $\ell \geq 1$ we define $\delta_{large,\ell} := \delta_{large}\delta_{c,(\ell-1)}$ and $\delta_{small,\ell} := \delta_{small}\delta_{c,(\ell-1)}$. For each level ℓ we define

- L_ℓ to be all objects $P \in \mathcal{P}$ with $r^{in}(P) \in (\delta_{large,\ell}, \delta_{small,(\ell-1)}]$; intuitively, they are "large" for level ℓ,
- M_ℓ to be all objects $P \in \mathcal{P}$ with $r^{in}(P) \in (\delta_{small,\ell}, \delta_{large,\ell}]$.

Note that our grid and the sets L_ℓ and M_ℓ depend on the (initial) choice of $\delta_{large}, \delta_{small}$ and δ_{cell}. In the next lemma, we show via a shifting argument that there are choices for these values such that the total area of the objects in $\bigcup_\ell M_\ell$ is very small. This will allow us later to pack them separately via a simple greedy algorithm. In particular, these choices are from a set of $O_\varepsilon(1)$ candidate values D, and thus we will be able to guess them later easily.

▶ **Lemma 3** ($\star$)**.** *Let $f \geq 1$. There is a global set D with $|D| \leq O_\varepsilon(1)$ such that for any set of f-fat convex objects $\mathcal{P}$ that can be packed in a knapsack $K = [0,1] \times [0,1]$, there are values $\delta_{large}, \delta_{small}, \delta_{cell} > 0$ that are all contained in D such that for the resulting hierarchical grid and the corresponding sets $\{L_\ell, M_\ell\}_\ell$ we have that the total area of all objects in $\bigcup_\ell M_\ell$ is bounded by ε.*

We generalize now our notion of discretized packings. Intuitively, like before, we require that there is a partition of the grid cells such that for each object $P \in \mathcal{P}$ there is a set of grid cells $\mathcal{G}_P$ that contain P and that are disjoint from the grid cells $\mathcal{G}_{P'}$ for each object $P' \in \mathcal{P} \setminus \{P\}$. Formally, we define that our packing of $\mathcal{P}$ is *discretized* if

- for each level ℓ and for each object $P \in \mathcal{P} \cap L_\ell$ there is a set of $O(1/\delta^2_{cell})$ grid cells $\mathcal{G}_P$ of level ℓ such that P is contained in $\mathcal{G}_P$, and there is a single grid cell of level $\ell - 1$ that contains all grid cells in $\mathcal{G}_P$, and
- for any two objects $P, P' \in \mathcal{P}$ (not necessarily of the same level) and for any two grid cells $C \in \mathcal{G}_P$ and $C' \in \mathcal{G}'_P$ their relative interiors are disjoint.

We show that by increasing the size of our knapsack by a factor of $1+\varepsilon$, there is a discretized packing for all objects in $\bigcup_\ell L_\ell$. As mentioned above, we will pack the objects in $\bigcup_\ell M_\ell$ separately later.

▶ **Lemma 4.** *Let each $f \geq 1$ and $\varepsilon > 0$. There is a global set D with $|D| \leq O_\varepsilon(1)$ such that for any set of f-fat objects $\mathcal{P}$ that can be placed non-overlappingly inside a knapsack $K = [0,1] \times [0,1]$, there is a choice for the grid with parameters $\delta_{large}, \delta_{small}, \delta_{cell} > 0$ such that all these values are contained in D and there is a discretized packing for $\mathcal{P} \cap (\bigcup_\ell L_\ell)$ inside an (augmented) knapsack $[0, 1 + O(\varepsilon)] \times [0, 1 + O(\varepsilon)]$.*

Proof. We start with the given packing of $\mathcal{P}$ in K and do a sequence of refinements which leads to our discretized packing for $\mathcal{P} \cap \bigcup_\ell L_\ell$. First, we use the increased size of the knapsack to ensure that for each level ℓ and any two objects $P, P' \in L_\ell$, the distance between P and P' is at least $2\delta_{c,\ell}$. Intuitively, increasing the size of the knapsack by a factor of $1+\varepsilon$ is equivalent to shrinking the objects in $\mathcal{P}$ by a factor of $1+\varepsilon$. Therefore, we can achieve this required minimum distance of $2\delta_{c,\ell}$ by choosing δ_{cell} appropriately according to the multiple constraints given in the proof of Lemma 2.

Next, we would like that for each level ℓ, each object in $\mathcal{P} \cap L_\ell$ is contained in a grid cell of level $\ell - 1$. This might not be the case, however, via a shifting argument (giving the grid a random shift) we can argue that this is the case for almost all objects in $\mathcal{P}$. The probability that an object in level ℓ intersects a grid line from level $\ell - 1$ is at most $8\delta_{small,\ell-1}/\delta_{cell,\ell-1} = 8\delta_{small}/\delta_{cell}$. Let this probability be smaller than $\varepsilon^2/2$, leading to a constraint $\delta_{small} \leq \varepsilon^2 \delta_{cell}/16$ on the choice of $\delta_{cell}, \delta_{small}$. Then the total area of intersecting objects must also be smaller than ε^2 as the area of all packed objects can be at most the area of the augmented knapsack. Thus we can easily pack these intersected objects into extra space that we gain via increasing the size of the knapsack (i.e., for a second time).

After this preparation, we process the objects $P \in \mathcal{P}$ level by level and define their corresponding sets $\mathcal{G}_P$, starting with the highest level. Consider a level ℓ. For each object $P \in \mathcal{P}$ of level ℓ we define $\mathcal{G}_P$ to be the set of all grid cells of level ℓ that intersect with P. Due to our minimum distance between any two objects in $\mathcal{P}$ of level ℓ, for any two different objects $P, P' \in \mathcal{P}$ of level ℓ we have that $\mathcal{G}_P \cap \mathcal{G}_{P'} = \emptyset$. Now it could be that a cell $\mathtt{C} \in \mathcal{G}_P$ for some $P \in \mathcal{P}$ intersects not only with P, but also with another object $P' \in \mathcal{P}$ of some deeper level $\ell' > \ell$. We call such a cell $\mathtt{C}$ *problematic*; recall that we wanted the cells in $\mathcal{G}_P$ to be used exclusively by P. Therefore, we move all objects $P' \in \mathcal{P}$ of some level $\ell' > \ell$ that intersect a problematic grid cell in $\mathcal{G}_P$. We pack them into extra space that we gain due to resource augmentation. We do this operation for all levels ℓ.

In the process above, we move objects that intersect problematic grid cells. We need to argue that the total area of these moved objects is small compared to the size of the knapsack and that, therefore, we can pack them into additional space that we gain due to our resource augmentation. In particular, we need to argue this globally, over all levels. The key insight is that if we define a set $\mathcal{G}_P$ for some object $P \in \mathcal{P}$ as above, then each problematic cell $\mathtt{C} \in \mathcal{G}_P$ must intersect the boundary of P and, since P is fat, the number of problematic cells $\mathtt{C} \in \mathcal{G}_P$ is very small compared to the number of cells $\mathtt{C}' \in \mathcal{G}_P$ that are contained in P and, thus, for sure *not* problematic.

By the classical *Barbier's theorem*, we know the perimeter of a convex set P of level ℓ is at most $\pi \cdot \text{diameter}(P) \leq 2\pi r^{out} \leq 2\pi f r^{in}$. A curve of length $\delta_{cell,\ell}$ is bound to be contained inside a circle of radius $\delta_{cell,\ell}$. This implies that this curve can intersect at most 9 grid cells as any circle of radius $\delta_{cell,\ell}$ can be bounded in a 3×3 grid square. Hence, the number of grid cells (of level ℓ) N_1 that the perimeter can intersect is at most $18\pi f r^{in}/\delta_{cell,\ell}$. On the other hand, P completely contains at least grid cells of area $\pi(r^{in} - 2\delta_{cell,\ell})^2$, i.e., the number of such gridcells N_2 is at least $\frac{\pi(r^{in}-2\delta_{cell,\ell})^2}{(\delta_{cell,\ell})^2}$. We need $N_1 \leq \varepsilon N_2$. Equivalently, we want to show, $\delta_{cell,\ell} \leq \frac{\varepsilon}{18f} \cdot \frac{(r^{in}-2\delta_{cell,\ell})^2}{r^{in}}$. For this we impose the condition that $\delta_{large} \geq \frac{72f}{\varepsilon}\delta_{cell}$. Then, $\frac{\varepsilon}{18f} \cdot \frac{(r^{in}-2\delta_{cell,\ell})^2}{r^{in}} \geq \frac{\varepsilon}{18f} \cdot \frac{(r^{in}/2)^2}{r^{in}} \geq \frac{\varepsilon}{18f} \cdot \frac{\delta_{large,\ell}}{4} \geq \delta_{cell,\ell}$.

Using this, we derive a *global* argumentation, stating that the total area of *all* problematic grid cells over all objects of all levels is at most an ε-fraction of the area of the knapsack. Also, if an object P' of some level ℓ' intersects a problematic grid cell $\mathtt{C}$ of some level $\ell < \ell'$, then P' is very small compared to $\mathtt{C}$. Thus, the total area of these objects intersecting a problematic grid cell $\mathtt{C}$ is essentially the same as the area of $\mathtt{C}$. Thus, we can pack all these objects into our additional space due to resource augmentation.

Finally, we can afford to increase the space of our knapsack such that this additional space is even by a constant factor larger than the total area of the objects we need to pack into it. Therefore, it is easy to find a discretized packing for them in this extra space. ◀

Algorithm. Now we describe our algorithm. First, we *correctly* guess (i.e., by brute-force enumeration of all possible cases) the values $\delta_{large}, \delta_{small}, \delta_{cell} > 0$ from set D due to Lemma 4. Note that we still do not know $\bigcup_\ell L_\ell$ or $\bigcup_\ell M_\ell$, i.e., which objects are there in the optimal packing. So, for each level ℓ we define $\tilde{L}_\ell$ to be all input objects P with $r^{in}(P) \in (\delta_{large,\ell}, \delta_{small,(\ell-1)}]$ and $\tilde{M}_\ell$ to be all input objects P with $r^{in}(P) \in (\delta_{small,\ell}, \delta_{large,\ell}]$.

Then, we compute an optimal discretized packing via a dynamic program. Intuitively, our DP computes an optimal subset of $\bigcup_\ell \tilde{L}_\ell$ for which there is a discretized packing. We introduce a DP-cell DP$[\ell, m]$ for each combination of a level ℓ and a value $m \in \{1, ..., n\}$. This cell corresponds to the subproblem of packing a maximum profit subset of the objects in $\tilde{L}_\ell, \tilde{L}_{\ell+1}, \tilde{L}_{\ell+2}, \ldots.$ via a discretized packing into at most m grid cells of level $\ell - 1$, i.e., with side length $\delta_{c,(\ell-1)}$ each. Recall that each object in $\tilde{L}_\ell$ is relatively large compared to the grid cells of level $\ell - 1$. Therefore, we can pack only constantly many items from $\tilde{L}_\ell$ into each of these m grid cells of level $\ell - 1$. Therefore, there are only constantly many options how the set $\mathcal{G}_P$ of an object $P \in \tilde{L}_\ell$ in the optimal solution to our subproblem can look like. We say that a *configuration* is a partition of a grid cell of level $\ell - 1$ into sets of grid cells of level ℓ. Each grid cell of level $\ell - 1$ contains $O(1/\delta^2_{cell})$ many grid cells of level ℓ. Hence, there are only constantly many configurations. We assume two configurations to be *equivalent* if they are identical up to translation by an integral multiple of $\delta_{c,(\ell-1)}$, i.e., by an integral multiple of the edge length of a grid cell of level $\ell - 1$. Denote by C the total number of resulting equivalence classes. We guess in time $m^{O(C)} \leq n^{O(C)}$ how many grid cells have each of the at most C configurations (up to equivalences). Then, we assign the items in $\tilde{L}_\ell$ into the grid cells according to this guess. We can do this by weighted bipartite matching. For each object $P \in \tilde{L}_\ell$, each possible configuration $\mathcal{G}'$, and each set in the partition of $\mathcal{G}'$, we can check easily whether P fits into $\mathcal{G}'$. In the bipartite graph, one side will contain the objects in $\tilde{L}_\ell$ and other side will contain the sets in the partition of $\mathcal{G}'$. If P fits into set Q in the partition of $\mathcal{G}'$, then there is an edge with edge cost as $w(P)$. Our guess yields a certain number m' of empty grid cells of level $\ell+1$ into which we need to pack items in $\tilde{L}_{\ell+1}, \tilde{L}_{\ell+2}, \dots$. We assign these items according to the solution in the DP-cell DP$[\ell+1, \min\{m', n\}]$. Note that there are at most n items and, hence, we never need more than n grid cells of level $\ell+1$. Also, since our input data is polynomially bounded, the number of classes $\tilde{L}_i$ is bounded by $n^{O(1)}$. Thus, our DP runs in time $n^{O(C)}$.

Additionally, we pack medium objects from the set $\bigcup_\ell \tilde{M}_\ell$ separately in a strip of the form $[0,1] \times [1, 1+O(\varepsilon)]$. We select the most profitable subset of $\bigcup_\ell \tilde{M}_\ell$ (up to a factor of $1+\varepsilon$) whose total area is bounded by ε (see Lemma 3). We replace each of these objects by the smallest square that contains it, which increases its area only by a constant factor. We can pack these squares efficiently into a (slightly larger) strip $[0,1] \times [1, 1+O(\varepsilon)]$ using the `NFDH` algorithm [13]. This yields the following lemma.

▶ **Lemma 5.** ($\star$) *In polynomial time we can compute a set $\mathcal{P}' \subseteq \bigcup_\ell \tilde{M}_\ell$ and a non-overlapping placement of $\mathcal{P}'$ inside $[0,1] \times [1, 1+O(\varepsilon)]$ such that $w(\mathcal{P}')$ is at least the profit of any subset of $\bigcup_\ell \tilde{M}_\ell$ whose total area is at most ε.*

One can easily extend our algorithm above to any constant dimension d. This completes the proof of Theorem 1. A consequence is that we obtain a polynomial time $(1+\varepsilon)$-approximation *without* resource augmentation if all input objects are small, i.e., if $r^{out}(P) \leq \varepsilon$ for each given object $P \in \mathcal{P}$. Using this property, we can argue that there is a $(1+\varepsilon)$-approximate solution in which only the area $[0, 1-\Theta(\varepsilon)] \times [0, 1-\Theta(\varepsilon)]$ of the knapsack is used. Thus, we can use the free space for the resource augmentation that is required by our algorithm due to Theorem 1.

▶ **Theorem 6.** ($\star$) *Let $d \in \mathbb{N}$ be a constant. There is a polynomial time $(1+O(\varepsilon))$-approximation for the geometric knapsack problem if the set of input objects $\mathcal{P}$ consists of convex fat d-dimensional objects such that $r^{out}(P) \leq \varepsilon$ for each $P \in \mathcal{P}$.*

3 Spheres

In this section we present our $(1+\varepsilon)$-approximation algorithm for the case of d-dimensional spheres. Let $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$ be a set of n number of d-dimensional hyperspheres. We denote the radius and profit of each hypersphere $C_i \in \mathcal{C}$ by r_i and w_i. For an object C_i we denote its volume (or area in 2-dimension) to be $a(C_i)$. For a collection of objects $\mathcal{A}$, we define its volume and profit to be $a(\mathcal{A}) := \sum_{C_i \in A} a(C_i)$ and $w(\mathcal{A}) := \sum_{C_i \in A} w(C_i)$, respectively. We are given a unit knapsack $K := [0,1]^d$.

We first consider the case of circles, i.e., $d = 2$. Let OPT be an optimal solution and $\mathcal{C}_{\mathsf{OPT}}$ be the circles in OPT. Let $\varepsilon \in (0, 1/2]$ be a constant and assume that $1/\varepsilon \in \mathbb{N}$. First, we want to classify the input circles into *small* and *large* circles such that each large circle is much larger than any small circle. Due to the following lemma, we can do this such that we can ignore all circles that are neither large nor small by losing only a factor of $1+\varepsilon$. We will use this standard shifting argument throughout the paper.

▶ **Lemma 7** ($\star$). *There is a set of global constants $\varepsilon^{(0)}, \dots, \varepsilon^{(1/\varepsilon)} \geqslant 0$ such that $\varepsilon^{(j)} = (\varepsilon^{(j-1)})^{24}$ for each $j \in \{1, \dots, 1/\varepsilon\}$ and a value $k \in \{1, \dots, 1/\varepsilon - 1\}$ with the following property: if we define $\varepsilon_{large} := \varepsilon^{(k)}$ and $\varepsilon_{small} := \varepsilon^{(k+1)}$, then sum of profits of all circles C_i in OPT with radii $\varepsilon_{small} \leq r_i \leq \varepsilon_{large}$ is at most $\varepsilon \cdot w(\mathsf{OPT})$.*

We guess the value $k \in \{0, ..., 1/\varepsilon - 1\}$ due to Lemma 7. We define that a circle $C_i \in \mathcal{C}$ is *large* if $r_i > \varepsilon_{large}$ and *small* if $r_i \leqslant \varepsilon_{small}$. Also, note that $\varepsilon_{small} = \varepsilon_{large}^{24}$.

3.1 Guessing large circles

We observe that in OPT there can be only a constant number of large circles since each large circle covers a constant fraction of the available area in the knapsack.

▶ **Proposition 8** (⋆). *Any feasible solution can contain at most $(1/\varepsilon)^{O_k(1)}$ large circles.*

We guess a feasible solution of the large circles in OPT that satisfy the packing constraints in time $n^{(1/\varepsilon)^{O_k(1)}}$, denote them by $\mathcal{C}_L^*$. In related problems, like the two-dimensional knapsack problem with squares or rectangles, one can easily guess the correct placement of the guessed large circles (assuming rational input data). For packing circles, it is not clear that if there is a packing in which the centers of the circles are placed at rational coordinates. However, in the following section, when we pack polygons, we can guarantee that there is an optimal solution in which the corner of each polygon has a rational coordinate.

Therefore, instead we first guess for each circle $C_i \in \mathcal{C}_L^*$ an *estimate* for its placement in OPT. Denote by $\hat{x}_i^{(1)}, \hat{x}_i^{(2)} \in [0, 1]$ the coordinates of the center of C_i in OPT. We guess values $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)} \in \{0, \frac{\varepsilon}{n}, \frac{2\varepsilon}{n}, ..., 1\}$ such that $\hat{x}_i^{(1)} \in [\tilde{x}_i^{(1)}, \tilde{x}_i^{(1)} + \frac{\varepsilon}{n})$ and $\hat{x}_i^{(2)} \in [\tilde{x}_i^{(2)}, \tilde{x}_i^{(2)} + \frac{\varepsilon}{n})$. Note that there are only $O(n^2/\varepsilon^2)$ possibilities for each $C_i \in \mathcal{C}_L^*$, and hence only $n^{(1/\varepsilon)^{O_k(1)}}$ possibilities overall for all circles $C_i \in \mathcal{C}_L^*$.

Given these guessed values $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ for each circle $C_i \in \mathcal{C}_L^*$, we verify that our guess was correct or not, i.e., confirm that there exists, indeed a corresponding placement for each circle $C_i \in \mathcal{C}_L^*$ such that the circles in $\mathcal{C}_L^*$ do not overlap. Therefore, we define a system of quadratic inequalities that describes the problem of finding such a placement. We require that this placement is consistent with our guesses $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ for each $C_i \in \mathcal{C}_L^*$.

$$\begin{aligned} \max\{\tilde{x}_i^{(1)}, r_i\} \leqslant x_i^{(1)} &\leqslant \min\left\{\tilde{x}_i^{(1)} + \frac{\varepsilon}{n}, 1 - r_i\right\} && \forall C_i \in \mathcal{C}_L^* \\ \max\{\tilde{x}_i^{(2)}, r_i\} \leqslant x_i^{(2)} &\leqslant \min\left\{\tilde{x}_i^{(2)} + \frac{\varepsilon}{n}, 1 - r_i\right\} && \forall C_i \in \mathcal{C}_L^* \\ (x_i^{(1)} - x_j^{(1)})^2 + (x_i^{(2)} - x_j^{(2)})^2 &\geqslant (r_i + r_j)^2 && \forall C_i, C_j \in \mathcal{C}_L^* \\ x_i^{(1)}, x_i^{(2)} &\geqslant 0 && \forall C_i \in \mathcal{C}_L^* \end{aligned} \tag{1}$$

Let $|\mathcal{C}_L^*| =: t$. Then, the above system has $2t$ variables and $k := O(t^2)$ constraints. It is not clear how to compute a solution to this system in polynomial time. It is not even clear whether it has a solution in which each variable has a rational value. However, in polynomial time, we can *decide* whether it has a solution (without computing the solution itself) using an algorithm from [23].

Note that the set of solutions satisfying system (1) is a semi-algebraic set in the field of real numbers. Thus, whether a given set of circles can be packed or not (the decision problem) reduces to a decision problem of whether this semi-algebraic set is nonempty or not. Here, each constraint $i \in [k]$ in (1) can be written as a function $f_i(x_1^{(1)}, x_1^{(2)}, \ldots, x_t^{(1)}, x_t^{(2)}) \geqslant 0$ where each $f_i \in \mathbb{Q}[x_1^{(1)}, x_1^{(2)}, \ldots, x_t^{(1)}, x_t^{(2)}]$ is a polynomial with rational coefficients of degree at most two. Thus deciding the circle packing problem is equivalent to deciding the truth of the following formula: $F := (\exists x_1^{(1)})(\exists x_1^{(2)}) \ldots (\exists x_t^{(1)})(\exists x_t^{(2)}) \wedge_{i=0}^{k} f_i(x_1^{(1)}, x_1^{(2)}, \ldots, x_t^{(1)}, x_t^{(2)}) \geqslant 0$. To solve this decision problem, we use the following result.

▶ **Theorem 9** ([23]). *Let $f_1, f_2, \ldots, f_k \in \mathbb{Q}[x_1^{(1)}, x_1^{(2)}, \ldots, x_t^{(1)}, x_t^{(2)}]$ be polynomials with absolute value of any coefficient to be represented by M bits and maximum degree Δ. There is an algorithm that decides whether the formula $F := (\exists x_1^{(1)})(\exists x_1^{(2)}) \ldots (\exists x_t^{(1)})(\exists x_t^{(2)}) \wedge_{i=0}^{k} f_i(x_1, y_1, \ldots, x_n, y_n) \geqslant 0$ is true, with a running time of $M^{O(1)}(k\Delta)^{O(t^2)}$.*

If it is true, the algorithm also returns polynomials $f, g_1, h_1, \ldots, g_t, h_t \in \mathbb{Q}[x]$ with coefficients of bit size at most $M^{O(1)}(k\Delta)^{O(t)}$ and maximum degree $k^{O(t)}$, such that for a root x of $f(x)$, the assignment $x_1^{(1)} = g_1(x), x_1^{(2)} = h_1(x), \ldots, x_t^{(1)} = g_n(x), x_t^{(2)} = h_n(x)$ satisfies the formula F.

Moreover, for any rational $\alpha > 0$, it returns values $\bar{x}_1^{(1)}, \bar{x}_1^{(2)} \ldots, \bar{x}_t^{(1)}, \bar{x}_t^{(2)} \in \mathbb{Q}$ such that $|\bar{x}_i^{(1)} - x_i^{(1)}| \leqslant \alpha$ and $|\bar{x}_i^{(2)} - x_i^{(2)}| \leqslant \alpha$, for $1 \leqslant i \leqslant t$, in time at most $(\log(1/\alpha)M)^{O(1)}(k\Delta)^{O(t^2)}$.

We crucially use here that our system has only constantly many variables and constraints, i.e., in (1), we have that Δ, k, t are constants and that M is polynomially bounded in n. From Theorem 9, we see that in polynomial time we can decide whether (1) has a solution.

If the system (1) does not have a solution, then we reject this guessed combination of $\mathcal{C}_L^*$ and values $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ for each circle $C_i \in \mathcal{C}_L^*$. We assume in the following that it has a solution. Observe that the guessed values $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ yield an estimate for $\hat{x}_i^{(1)}, \hat{x}_i^{(2)}$ up to a (polynomially small) error of ε/n.

3.2 Placing small circles

We want to select small circles from $\mathcal{C}$ and place them inside the knapsack, so that they do not overlap with each other or with the circles in $\mathcal{C}_L^*$. To this end, we define $\varepsilon_{cell} := \varepsilon_{large}^{12}$ (i.e., $\varepsilon_{small} = \varepsilon_{large}^{12}\varepsilon_{cell} = \varepsilon_{cell}^2$) to subdivide the knapsack into a grid with $1/\varepsilon_{cell}^2$ square grid cells of side length ε_{cell} . Our choice of parameters ensures that each small circle is small compared to each grid cell and each large circle is big compared to each grid cell. Formally, for each $\ell, \ell' \in \{0, 1, \ldots, \frac{1}{\varepsilon_{cell}} - 1\}$ we define a grid cell $G_{\ell,\ell'} := [\ell \cdot \varepsilon_{cell}, (\ell+1) \cdot \varepsilon_{cell}) \times [\ell' \cdot \varepsilon_{cell}, (\ell'+1) \cdot \varepsilon_{cell})$. We define the set of all grid cells by $\mathcal{G} := \{G_{\ell,\ell'} : \ell, \ell' \in \{0, 1, \ldots, 1/\varepsilon_{cell} - 1\}\}$.

We say that a placement of a circle $C_i \in \mathcal{C}_L^*$ is *legal* if its center is placed at a point $(x_i^{(1)}, x_i^{(2)})$ such that $\max\{\tilde{x}_i^{(s)}, r_i\} \leqslant x_i^{(s)} \leqslant \min\left\{\tilde{x}_i^{(s)} + \frac{\varepsilon}{n}, 1 - r_i\right\}$ for each $s \in \{1, 2\}$. We show that there is a structured packing with near-optimal profit in which each small circle is contained in a grid cell that does not intersect with any large circle in $\mathcal{C}_L^*$ in any legal packing of them. This will allow us to decouple the remaining problem for the small circles from the large circles, even though we do not know the exact placement for the latter. Moreover, in each grid cell the small circles use only a reduced area of size $(1-\varepsilon)\varepsilon_{cell} \times (1-\varepsilon)\varepsilon_{cell}$. Let $\mathcal{C}_S^*$ denote the small circles in OPT.

▶ **Lemma 10.** *In polynomial time, we can compute a set of grid cells $\mathcal{G}_w$ such that no grid cell in $\mathcal{G}_w$ intersects with any circle $C_i \in \mathcal{C}_L^*$ for any legal placement of C_i. Moreover, there is a set of small circles $\mathcal{C}_S \subseteq \mathcal{C}_S^*$ such that $w(\mathcal{C}_S) \geqslant (1-\varepsilon)w(\mathcal{C}_S^*)$ and the circles in $\mathcal{C}_S$ can be packed non-overlappingly inside $|\mathcal{G}_w|$ grid cells of size $(1-\varepsilon)\varepsilon_{cell} \times (1-\varepsilon)\varepsilon_{cell}$ each.*

We will prove Lemma 10 later in Section 3.3. Using it, we compute an approximation to the set $\mathcal{C}_S$ via Theorem 1.

We pack the computed circles into our grid cells $\mathcal{G}_w$, denote them by $\mathcal{C}_S'$. In particular, they do not intersect any of the large circles in $\mathcal{C}_L^*$ in any legal placement of them. Our solution (corresponding to the considered guesses) consists of $\mathcal{C}_L^* \cup \mathcal{C}_S'$. Recall that we can guarantee that these circles can be packed non-overlappingly inside the knapsack. Also, for the circles in $\mathcal{C}_S'$ we computed their placement *exactly* and for the circles in $\mathcal{C}_L^*$ we computed their placement up to our polynomially small error of ε/n. In Appendix A we show how to reduce this error to an exponentially small error.

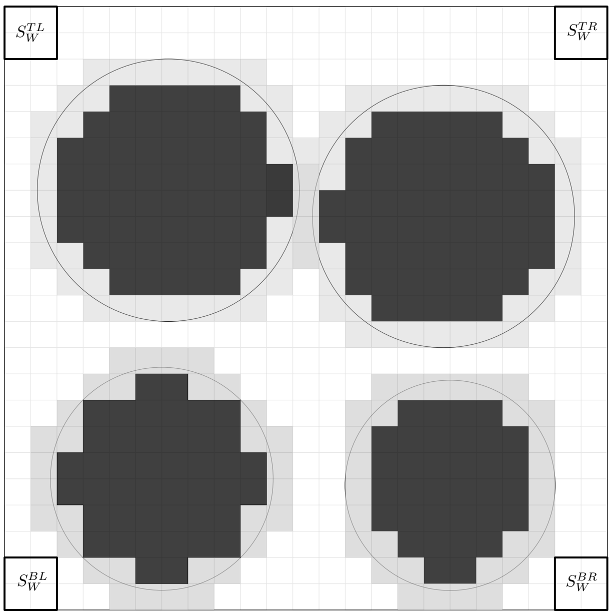

Figure 5 Partitioning grid cells into white, black and gray cells. Later corner regions $S_W^{BL}, S_W^{TL}, S_W^{BR}, S_W^{TR}$ are used to to pack items in gray cells.

3.3 Structural packing for small circles

In this section, we prove Lemma 10. First, we show that intuitively almost every small circle in $\mathcal{C}_S^*$ is contained inside some grid cell. Formally, we show that the total area of all other small circles in $\mathcal{C}_S^*$ is small. For any set $\mathcal{S}$ of circles or grid cells, we define $a(\mathcal{S})$ to be the total area of the elements in $\mathcal{S}$.

▶ **Lemma 11** (⋆). *Let $\mathcal{C}_{cut} \subseteq \mathcal{C}_S^*$ be the set of all small circles in $\mathcal{C}_S^*$ that intersect more than one grid cell. We have that $a(\mathcal{C}_{cut}) \leqslant 8\varepsilon_{small}/\varepsilon_{cell} \leqslant \varepsilon\varepsilon_{large}^2/64$.*

We will repack the circles in $\mathcal{C}_{cut}$ later such that each of them is contained inside one single grid cell. Thus, for each small circle $C_i \in \mathcal{C}_{\mathsf{OPT}} \setminus \mathcal{C}_{cut}$ there is a grid cell $G_{\ell,\ell'}$ for some $\ell, \ell' \in \{0, 1, ..., 1/\varepsilon_{cell} - 1\}$ such that C_i is contained in $G_{\ell,\ell'}$ in OPT. When we select and place small circles, we must be careful that they do not intersect any large circles from $\mathcal{C}_L^*$. One difficulty for this is that we do not know the precise coordinates of the large circles. Therefore, we place small circles only into grid cells that do not overlap with any large circle from $\mathcal{C}_L^*$ in any legal placement of them. Formally, we partition the cells in $\mathcal{G}$ into three types: white, gray, and black cells (see Figure 5).

▶ **Definition 12.** Let $G_{\ell,\ell'} \in \mathcal{G}$ for some $\ell, \ell' \in \{0, 1, ..., 1/\varepsilon_{cell} - 1\}$. The cell $G_{\ell,\ell'}$ is

- *white* if $G_{\ell,\ell'}$ does not intersect with any circle $C_i \in \mathcal{C}_L^*$ for any legal placement of C_i,
- *black* if $G_{\ell,\ell'}$ is contained in some circle $C_i \in \mathcal{C}_L^*$ for any legal placement of C_i,
- *gray* if $G_{\ell,\ell'}$ is neither white nor black.

The gray cells are problematic for us since a gray cell might be (partially) covered by a large circle in $\mathcal{C}_L^*$ but we do not know by how much (and which part of the cell). Therefore, we do not place any small circles into gray cells. However, OPT might place small circles into these cells (and obtain the profit of these circles). On the other hand, we can show that the number of gray cells is very small, only a small fraction of all grid cells can be gray. In order to do this, we use the fact that the values $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ for each circle $C_i \in \mathcal{C}_L^*$ estimate the placement of each large circle relatively accurately, and that the grid cells are relatively small. This allows us to prove that almost all cells are black or white. Also, we can compute all gray cells efficiently. Let $\mathcal{G}_g \subseteq \mathcal{G}$ denote the set of all gray grid cells in $\mathcal{G}$.

▶ **Lemma 13** ($\star$). *The total area of gray cells $a(\mathcal{G}_g)$ is at most $\varepsilon\varepsilon_{large}^2/5$. We can compute $\mathcal{G}_g$ in polynomial time.*

Unfortunately, it is not sufficient for us that there are only few gray cells. It might be that almost all cells are either gray or black and, hence, we need to place most of the selected small circles into gray cells (in order to obtain an $(1+\varepsilon)$-approximate solution).

However, we can show that this is not the case. We prove that the number of white cells (which we can safely use for small circles) is at least by a factor $1/\varepsilon$ larger than the number of gray cells. To show this, we exploit the geometry of the circles. In each corner of the knapsack, there are cells that cannot intersect with any large circle, simply because the grid cells are small compared to the large circles and because of the shape of the large circles (see the corner regions in Figure 5). Hence, these grid cells are white. Let $\mathcal{G}_w \subseteq \mathcal{G}$ denote the set of all white grid cells in $\mathcal{G}$.

▶ **Lemma 14** ($\star$). *The total area of white grid cells $a(\mathcal{G}_w)$ is at least $\varepsilon_{large}^2/4$. We can compute $\mathcal{G}_w$ in polynomial time.*

Using Lemmas 11, 13, and 14, we show that there is a $(1+\varepsilon)$-approximate solution in which each small circle is contained in some white cell; in particular, no small circle is placed inside a gray cell. To prove this, we delete all small circles in the $O(\varepsilon|\mathcal{G}_w|)$ white grid cells with the smallest total profit among all white cells and place all circles from gray cells and all circles from $\mathcal{C}_{cut}$ into those.

▶ **Lemma 15** ($\star$). *There is a set $\mathcal{C}_S' \subseteq \mathcal{C}_S^*$ of small circles with $p(\mathcal{C}_S') \geqslant (1-\varepsilon)p(\mathcal{C}_S^*)$ such that there is a packing for $\mathcal{C}_S'$ using the grid cells in $\mathcal{G}_w$ only.*

We complete the proof of Lemma 10 by applying the following lemma to $\mathcal{C}_S := \mathcal{C}_S''$ which shows that we can sacrifice a factor of $1 + O(\varepsilon)$ to be able to use resource augmentation when we pack the small circles.

▶ **Lemma 16** ($\star$). *There is a set of small circles $\mathcal{C}_S'' \subseteq \mathcal{C}_S'$ such that $w(\mathcal{C}_S'') \geq (1-\varepsilon)w(\mathcal{C}_S')$ and it is possible to place the circles in $\mathcal{C}_{\text{sml}}''$ non-overlappingly inside $|\mathcal{G}_w|$ square knapsacks of size $(1-\varepsilon)\varepsilon_{cell} \times (1-\varepsilon)\varepsilon_{cell}$ each.*

3.4 Higher dimensions

Our techniques from the previous section extend directly to the problem of packing hyperspheres in any (constant) dimension d which yields our main theorem for the setting of packing (hyper-)spheres.

▶ **Theorem 17** ($\star$). *Let $d \in \mathbb{N}$ be a fixed constant. For the geometric knapsack problem with d-dimensional hyperspheres, there is a polynomial time algorithm that computes a set of hyperspheres $\tilde{\mathcal{C}}$ with $p(\tilde{\mathcal{C}}) \geqslant (1-\varepsilon)\mathsf{OPT}$ that can be placed non-overlappingly inside the knapsack. For all but $O_\varepsilon(1)$ hyperspheres in $\tilde{\mathcal{C}}$ we compute the precise coordinate of the corresponding packing; for the other $O_\varepsilon(1)$ circles we compute an estimate of the packing with an additive error of at most $\frac{1}{2^{n/\varepsilon}}$ in each dimension.*

4 Polygons

In this section, we adjust our techniques from the previous sections to obtain a PTAS for the case that each input object is a fat and convex polygon with at most a constant number of edges whose lengths differ by at most a constant factor, and such that each angle between adjacent edges is larger than $\pi/2$. These objects generalize regular polygons with greater than 4 sides.

Formally, we assume that we are given a set $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ of n polygons that are (f, α, q, t)*-well-behaved*, i.e., for each polygon $P_i \in \mathcal{P}$ we assume that

- P_i is fat, i.e., $r_i^{\text{out}}/r_i^{\text{in}} \le f$ for some (global) constant $f \ge 1$, where r_i^{out} and r_i^{in} is the radius of the smallest circle containing P_i and the radius of the largest circle contained in P, respectively,
- the angle between any two consecutive edges of P_i is at least $\pi/2 + \alpha$, for some (global) constant $\alpha > 0$,
- P_i has at most q edges for some (global) constant q such that the lengths of any two of its edges differ at most by a factor of t.

For example, regular pentagons are $(2, \pi/10, 5, 1)$-well-behaved. For each polygon $P_i \in \mathcal{P}$ we denote by w_i its profit, and for a set of polygons $\mathcal{P}' \subseteq \mathcal{P}$ we denote by $w(\mathcal{P}') := \sum_{P_i \in \mathcal{P}'} w_i$ their total profit. For any object C we define its area to be $a(C)$, and for any collection of objects $\mathcal{A}$ we denote their total area by $a(\mathcal{A}) := \sum_{P_i \in A} a(P_i)$. We want to pack a subset of $\mathcal{P}$ non-overlappingly into the unit knapsack $K := [0,1]^2$. We do not allow rotations in our packing.

Let $\varepsilon > 0$. We require that $\varepsilon < g(f, \alpha, q, t)$ for a function g to be defined later. In contrast to the case with hyperspheres, we show that we can compute each coordinate of our packing exactly. We classify each polygon $P_i \in \mathcal{P}$ as large or small according to the respective value r_i^{in}. For this, we define values ε_{large} and ε_{small}. For technical reasons, we need that $\varepsilon_{small} \le h(\varepsilon_{large})$ for some decreasing function $h : \mathbb{R} \to \mathbb{R}$ to be defined later.

▶ **Lemma 18.** ($\star$) *There is a set of global constants $\varepsilon^{(0)}, \dots, \varepsilon^{(1/\varepsilon)} \geqslant 0$ such that $\varepsilon^{(j)} = h(\varepsilon^{(j-1)})$ for each $j \in \{0, \dots, 1/\varepsilon - 1\}$ and a value $k \in \{0, \dots, 1/\varepsilon - 1\}$ with the following properties. If we define $\varepsilon_{large} := \varepsilon^{(k)}$ and $\varepsilon_{small} := \varepsilon^{(k+1)}$, then by losing a factor of $1 + \varepsilon$ in our approximation guarantee, we can assume that each polygon $P_i \in \mathcal{P}$ satisfies that $r_i^{\text{in}} \le \varepsilon_{small}$ or $r_i^{\text{in}} > \varepsilon_{large}$.*

We guess the value $k \in \{0, \dots, 1/\varepsilon - 1\}$ due to Lemma 18 and define that a polygon $P \in \mathcal{P}$ is *large* if $r_i^{\text{in}} \ge \varepsilon^{(k-1)} = \varepsilon_{large}$ and *small* if $r_i^{\text{in}} < \varepsilon^{(k)} = \varepsilon_{small}$. We discard all input polygons that are neither large nor small. Similar to the case of hyperspheres, we guess the large polygons in OPT. Since they are fat, there can be only constantly many of them.

▶ **Proposition 19.** *Any feasible solution can contain at most $(1/\varepsilon)^{O(1)}$ large polygons.*

We now calculate the placement of the large polygons using a linear program. Define $\mathcal{P}_L^*$ to be the set of large polygons in OPT. Note that, since they are convex polygons instead of circles or hyperspheres, we can compute an *exact* placement of the polygons with *rational* coordinates. For this, we use the following approach, which was also noted by Abrahamsen et al. in [1]. Consider the placement of the polygons $\mathcal{P}_L^*$ in OPT. Each side e of each polygon $P_i \in \mathcal{P}_L^*$ is contained in a line $\{x : a_e x = b_e\}$ for some vector a_e and a scalar b_e. For each corner vertex v of each polygon $P_j \in \mathcal{P}_L^*$, we have that $a_e x \ge b_e$ or $a_e x \le b_e$ (or both); we guess which of these cases applies. Let v_i be a special vertex for each polygon P_i defined as a vertex with the least value of $x_i^{(1)}$. Then, the coordinates $(x_i^{(1)}, x_i^{(2)})$ of the special vertex v_i of each polygon $P_i \in \mathcal{P}_L^*$ satisfy a system of linear inequalities defined as follows. There are three types of inequalities:

- *Positivity constraints*: $x_i^{(1)}, x_i^{(2)} \geq 0, \forall P_i \in \mathcal{P}_L^*$
- *Packing constraints*: $\forall P_i, P_j \in \mathcal{P}_L^*$ any vertex v in polygon $P_j \neq P_i$ cannot lie inside P_i. A vertex v lies inside polygon P_i if it satisfies the inequalities described above.
- *Container constraints*: $\forall P_i \in \mathcal{P}_L^*$, $0 \leq x_i^{(1)} \leq a_i$ and $b_i \leq x_i^{(2)} \leq 1 - c_i$, where a_i, b_i, and c_i can be calculated exactly in constant time for a given polygon. They represent the maximum value of $x_{i,v}^{(1)} - x_i^{(1)}$, maximum value of $x_i^{(2)} - x_{i,v}^{(2)}$, and maximum value of $x_{i,v}^{(2)} - x_i^{(2)}$, where $x_{i,v}^{(1)}, x_{i,v}^{(2)}$ vary over all vertices v of polygon P_i.

From Proposition 19, we know that there can only be at most $(1/\varepsilon)^{O(1)}$ large polygons in OPT. We take all possible subsets $\mathcal{P}_L^*$ of this size and smaller from the set $\mathcal{P}$, which is polynomial in number. We compute a feasible solution of packing of these subsets P_L^* to it which is easy since it has only $O_{\varepsilon,k,f,\alpha,t,q}(1)$ variables and constraints for each subset for polynomially many subsets, by using the ellipsoid method.

Now for each guessed large subset P_L^*, we compute a near-optimal packing of the small polygons P_S^*. Our goal is to pack the small polygons in the bin with only a loss of ε-fraction of profit, corresponding to the guessed P_L^*. We then return the solution $P_L^* \cup P_S^*$ which has maximum weight over all guessed values of P_L^* initially and claim that this packing is near-optimal.

In order to pack small polygons, we need a corresponding version of Lemma 10. We define grid cells again similarly such that ε_{cell} is much smaller compared to ε_{large} and ε_{small} is much smaller compared to ε_{cell}. Intuitively, since our input polygons are well-behaved, we can prove that a certain amount of space is not used by the large polygons, similar to Lemma 14. To ensure this, we require that ε is sufficiently small, which in particular also yields a bound on ε_{cell}. Using this, we show that there are many grid cells that are disjoint from any large polygon (similarly as the white grid cells in Section 3).

▶ **Lemma 20.** ($\star$) *There is a function $g : \mathbb{R}_{\geq 0}^4 \to \mathbb{R}_{\geq 0}$ such that if all given polygons are (f, α, q, t)-well-behaved and $\varepsilon < g(f, \alpha, q, t)$ then*

- *in polynomial time we can compute a set of grid cells $\mathcal{G}_w$ such that no grid cell in $\mathcal{G}_w$ intersects with any polygon $P_i \in \mathcal{P}_L^*$,*
- *there is a set of small polygons $\mathcal{P}_S \subseteq \mathcal{P}_S^*$ such that $w(\mathcal{P}_S) \geq (1 - O(\varepsilon))w(\mathcal{P}_S^*)$ for the optimal packing of large polygons, and*
- *the polygons in $\mathcal{P}_S$ can be packed non-overlappingly inside $|\mathcal{G}_w|$ grid cells that have size $(1 - \varepsilon)\varepsilon_{cell} \times (1 - \varepsilon)\varepsilon_{cell}$ each.*

We can prove Lemma 20 with similar techniques as we used in the proof of Lemma 10. In order to select and place the small polygons, we use the algorithm due to Theorem 1.

Let P_S^* denote our computed solution for the small polygons. We return the solution $\tilde{P} = P_L^* \cup P_S^*$ which has maximum weight over all initially guessed combinations for the polygons in P_L^* and their approximate coordinates.

▶ **Theorem 21.** *For any constants $f, q \geq 1$ and $t, \alpha > 0$ there is a PTAS for the geometric knapsack problem for (f, α, q, t)-*well-behaved *polygons.*

5 Conclusion

We almost settle the approximability of the geometric knapsack problem in the setting of packing spheres into a hypercube knapsack. However, it remains an open problem whether rational coordinates always suffice in an optimal packing. If not, it would be an interesting question to determine the best approximation ratio one can obtain if we allow only rational coordinates for the centers of the circles (while the optimal packing has no such restrictions). It would be also interesting to obtain a PTAS for the case of d-dimensional fat convex objects. Another interesting but difficult open question is whether the case of convex but not necessarily fat input objects in the plane admits a PTAS. The best known result for this setting is only an $O(1)$-approximation in quasi-polynomial time (assuming polynomially bounded integral input data [33]). Already for the special case of axis-parallel rectangles, it is open whether a PTAS exists.

References

1 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for ER-completeness of two-dimensional packing problems. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1014–1021. IEEE, 2020.

2 Anna Adamaszek, Tomasz Kociumaka, Marcin Pilipczuk, and Michał Pilipczuk. Hardness of approximation for strip packing. *ACM Transactions on Computation Theory*, 9(3):14:1–14:7, 2017.

3 Anna Adamaszek and Andreas Wiese. A quasi-ptas for the two-dimensional geometric knapsack problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1491–1505. SIAM, 2014.

4 Nikhil Bansal, José R. Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31–49, 2006.

5 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 13–25. SIAM, 2014.

6 Julia A. Bennell and José F. Oliveira. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:S93–S105, 2009.

7 Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

8 Ignacio Castillo, Frank J. Kampas, and János D. Pintér. Solving circle packing problems by global optimization: numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786–802, 2008.

9 Vítor Gomes Chagas, Elisa Dell'Arriva, and Flávio Keidi Miyazawa. Approximation schemes under resource augmentation for knapsack and packing problems of hyperspheres and other shapes. In *International Workshop on Approximation and Online Algorithms (WAOA)*, pages 145–159. Springer, 2023.

10 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

11 Hai-Chau Chang and Lih-Chung Wang. A simple proof of Thue's theorem on circle packing, 2010. `arXiv:1009.4322`.

12 Miroslav Chlebík and Janka Chlebíková. Hardness of approximation for orthogonal rectangle packing and covering problems. *J. Discrete Algorithms*, 7(3):291–305, 2009.

13 E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

14 Erik D. Demaine, Sándor P. Fekete, and Robert J. Lang. Circle packing for origami design is hard. *CoRR*, 2010. `arXiv:1008.1224`.

15 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Packing disks into disks with optimal worst-case density. *Discrete & Computational Geometry*, 69(1):51–90, 2023.

16 Ferenc Fodor. The densest packing of 13 congruent circles in a circle. *Beiträge zur Algebra und Geometrie*, 44(2):431–440, 2003.

17 Hamish J. Fraser and John A. George. Integrated container loading software for pulp and paper industry. *European Journal of Operational Research*, 77(3):466–474, 1994.

18 E. Friedman. Circles in squares. `https://erich-friedman.github.io/packing/cirinsqu/`. Accessed: 2024-03-03.

19 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, Sandy Heydrich, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. *ACM Trans. Algorithms*, 17(4):33:1–33:67, 2021.

20 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramírez-Romero, and Andreas Wiese. Improved Approximation Algorithms for 2-Dimensional Knapsack: Packing into Multiple L-Shapes, Spirals, and More. In *Symposium on Computational Geometry (SoCG)*, pages 39:1–39:17, 2021.

21 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.

22 Michael Goldberg. Packing of 14, 16, 17 and 20 circles in a circle. *Mathematics Magazine*, 44(3):134–139, 1971.

23 Dima Grigoriev and Nicolai N. Vorobjov Jr. Solving systems of polynomial inequalities in subexponential time. *J. Symb. Comput.*, 5(1/2):37–64, 1988.

24 Thomas C. Hales and Samuel P. Ferguson. A formulation of the kepler conjecture. *Discrete & Computational Geometry*, 36:21–69, 2006.

25 Mhand Hifi and Rym M'hallah. A literature review on circle and sphere packing problems: Models and methodologies. *Advances in Operations Research*, 2009.

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

27 Johannes Kepler. *The six-cornered snowflake.* Paul Dry Books, 2010.

28 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *Symposium on Computational Geometry (SoCG)*, pages 48:1–48:17, 2021.

29 Arindam Khan and Eklavya Sharma. Tight approximation algorithms for geometric bin packing with skewed items. *Algorithmica*, 85(9):2735–2778, 2023.

30 Robert J. Lang. A computational algorithm for origami design. In *Symposium on Computational Geometry (SoCG)*, pages 98–105, 1996.

31 Carla Negri Lintzmayer, Flávio Keidi Miyazawa, and Eduardo Candido Xavier. Two-dimensional knapsack for circles. In *Latin American Theoretical Informatics Symposium (LATIN)*, pages 741–754. Springer, 2018.

32 Boris D. Lubachevsky and Ronald L. Graham. Curved hexagonal packings of equal disks in a circle. *Discrete & Computational Geometry*, 18(2):179–194, 1997.

33 Arturo I. Merino and Andreas Wiese. On the two-dimensional knapsack problem for convex polygons. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 84:1–84:16, 2020.

34 Joseph S. B. Mitchell. Approximating maximum independent set for rectangles in the plane. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 339–350. IEEE, 2021.

35 Flávio K. Miyazawa and Yoshiko Wakabayashi. Techniques and results on approximation algorithms for packing circles. *São Paulo Journal of Mathematical Sciences*, 16(1):585–615, May 2022.

36 Ronald Peikert, Diethelm Würtz, Michael Monagan, and Claas de Groot. Packing circles in a square: a review and new results. In *System Modelling and Optimization*, pages 45–54. Springer, 1992.

37 E. Specht. The best known packings of equal circles in a square. `http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html`. Accessed: 2024-03-03.

38 Péter Gábor Szabó, Mihaly Csaba Markót, Tibor Csendes, Eckard Specht, Leocadio G Casado, and Inmaculada García. *New approaches to circle packing in a square: with program codes*, volume 6. Springer Science & Business Media, 2007.

39 Maryna S. Viazovska. The sphere packing problem in dimension 8. *Annals of Mathematics*, pages 991–1015, 2017.

40 Huaiqing Wang, Wenqi Huang, Quan Zhang, and Dongming Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(2):440–453, 2002.

A Improving the precision of the large spheres

Recall that for each large circle $C_i \in \mathcal{C}_L^*$ we guessed its center in the optimal packing up to a polynomial error of $\frac{\varepsilon}{n}$. We improve this to only an exponential error of at most $\frac{1}{2^{n/\varepsilon}}$. To do this, we apply Theorem 9 with $\alpha := \Theta(\frac{1}{2^{n/\varepsilon}})$. This yields more precise estimates $\bar{x}_i^{(1)}, \bar{x}_i^{(2)}$ for each $C_i \in \mathcal{C}_L^*$. There is an important subtlety though: for our guessed coordinates $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ we can assume that they differ from the coordinates of OPT by at most our polynomial error of $\frac{\varepsilon}{n}$. For the new estimates $\bar{x}_i^{(1)}, \bar{x}_i^{(2)}$ we can *not* guarantee this: our subroutine from Theorem 9 possibly returns a solution that is (close to) feasible for the large circles, but not (close to) a solution that is feasible for the large and for the small circles. Because of this, we guessed the estimates $\tilde{x}_i^{(1)}, \tilde{x}_i^{(2)}$ for each $C_i \in \mathcal{C}_L^*$, so that we can assume that these estimates really correspond to OPT and not just to some arbitrary solution to (1).

If it were true that there is always a $(1+\varepsilon)$-approximate solution in which the center of each circle has rational coordinates that can be encoded with a polynomially bounded number of bits, then we could choose α appropriately to compute it. More precisely, we could compute a range for each coordinate that contains only one rational number with a bounded number of bits, and we could compute this number afterward.