Algorithmic Aspects of Packing Problems

Dissertation zur Erlangung des Grades einer
Doktorin der Naturwissenschaften - (Dr. rer. nat.)

vorgelegt am

Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Nadja Seiferth

Berlin, 2021

Erstgutachter: Prof. Dr. Helmut Alt
Zweitgutachter: Prof. Dr. Otfried Cheong

Tag der Disputation: 29.11.2021

Abstract

In this thesis, we study different kinds of *packing problems*. A packing is an arrangement of geometric objects in Euclidean space of any fixed dimension such that the interiors of the packed objects are pairwise disjoint. The aim is to optimize some goal function such as the area of the smallest convex container enclosing the objects or the number of objects that can be packed into a given container. Here, we study problems only in two or three dimensions.

The thesis is divided into two parts. In the first part, we study packing equal objects. In most cases, we are given a container and want to compute the maximum number of copies of an object that can be packed into it. We obtain the following results:

The container is a fat parallelogram or triangle. Given a container parallelogram by a base edge and an additional point, there is a polynomial-time approximation scheme (PTAS) to compute the maximum number of copies of an object that can be packed into the container under translation or rigid motions if the following conditions hold:

- The inner angles of the container are bounded from below by a constant, i.e., the container is *fat*.
- Either the object to be packed is part of the problem description and not part of the algorithm input or the area of the object to be packed is bounded from below by a constant times its squared diameter.

The same holds for a triangular container given by its side lengths.

The container is an arbitrary triangle. If the container is an arbitrary non-fat triangle given by its side lengths, there can be a constant factor approximation for the maximum number of unit disks that can be packed into it obtained in polynomial time.

The objects to be packed are unit disks or unit spheres. Given a convex container, such that its area is bounded from below by a constant times its squared diameter, there is a PTAS for computing the maximum number of unit disks that can be packed into it. The same holds for any polygonal container, if its description is not part of the input but part of the problem description and the input is just a scaling factor for it. We also obtain a PTAS for a similar problem in three dimensions: Given a sphere by its radius, what is the maximum number of unit spheres that can be packed into it? For this result, we revisit parts of the proof of Hales [29] to obtain the following result: There exists a constant c such that for every packing of infinitely many unit spheres into

three dimensional space, the density inside a sphere of radius r is bounded from above by $\pi/\sqrt{18} + c/r$. It is crucial, that c is, in contrast to the proof of Hales et al. [30, 29], independent of the packing.

Most variants of the problems just mentioned have in common that their input is very concise, i.e., it consists only of a constant number of numbers. Therefore, it seems hard to prove any kind of hardness or give algorithms and indeed, neither one of the problems just mentioned is known to be NP-hard nor to be in NP. To our knowledge, these are the first results of the approximability for this kind of problems with mentioned concise input and therefore, the first about the complexity of this kind of problems.

The next result differs from the previously studied problems in the sense that the input is larger. It is the last result in the first part of the thesis.

Packing unit disks in 3D under translation. In three dimensions, there is to our knowledge no approximation algorithm known for packing objects different from axis-parallel boxes under translation whereas in two dimensions, there is a constant factor approximation for packing convex polygons under translations minimizing the area of the smallest axis-parallel rectangular or convex container [6]. We make a step towards packing more general objects in three dimensions by giving a constant factor approximation for packing unit disks under translation into an axis-parallel box with minimum volume. Furthermore, we show that there can not be a convex container of bounded volume such that all possible unit disks can be packed into it. This is in contrast to the equivalent two dimensional problem where all possible segments with length one can be packed for example into half a circle of radius two.

The second part of the thesis studies the complexity of two basic two-dimensional packing problems.

Minimum area container packing. It is known that there can not be a PTAS for StripPacking unless $\mathsf{P} = \mathsf{NP}$ but there exists a constant factor approximation algorithm (see e.g. [5]). For packing convex polygons under translation into minimum area convex or axis-parallel rectangular containers, there is a constant factor approximation [6]. Unlike for StripPacking, it is not known whether a PTAS can exist or not. We narrow this gap by showing that there can not exist a FPTAS unless $\mathsf{P} = \mathsf{NP}$. The same holds when we allow rotations by $90°$ additionally to translations.

Line segment packing. Kim and Miltzow showed that maximizing the number of line segments from a given set of line segments that can be packed into a simple polygon is NP-hard[40]. We strengthen this result by showing that packing the maximum number of line segments into a square is NP-hard, albeit we allow parallel line segments, which is not necessary for Kim and Miltzows result.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich alle Hilfsmittel und Hilfen angegeben habe und versichere, auf dieser Grundlage die Arbeit selbständig verfasst zu haben. Die Arbeit habe ich nicht in einem frühren Promotionsverfahren eingereicht.

Berlin, den 7. Juli 2021

Acknowledgements

First and foremost, I would like to thank my advisor Helmut Alt. He came up with many interesting problems to study, always gave me the feeling that I am able to finish this thesis even though it took longer than he thought and always had an open (virtual) door for my questions and need for discussion. I am grateful that I could learn from his broad knowledge and experience.

Furthermore, I would like to thank Otfried Cheong for agreeing to be my second referee, and his friendly work environment, hospitality and all the nice excursions during my stay in Daejon.

I do not want to miss the opportunity to thank Günter Rote and Wolfgang Mulzer for funding me after Helmut had gotten his emeritus status, and that I always just could crash into their offices asking for their expertise.

I would like to thank all the wonderful people I got to work with at the AG Theoretische Informatik, some of whom became friends over the time, for their motivating words in the times of need thereof, the educational coffee rounds, fun trips to conferences and the encouraging chats in the corridor. I especially like to thank my "roommate" Jonas Cleve for being my rubber duck and computer specialist and Frank Hoffmann among other things for the occasionally shared nugget of wisdom and his facial expressions during the noon seminar. I hope, he knew how much I enjoyed having him as a colleague.

I am grateful for all my marvelous friends and family, who often listened to me complaining about what is not working or going to plan even though they probably got only a glimpse of what I was talking about.

Finally, I would like to thank my boyfriend Paul Seiferth, who became my husband and father of my son along the way. He encouraged me to do my PhD, always had my back, pushed me, but not too hard, and lately gave me free evenings and weekends to write instead of changing diapers in spite of his own weeks full of work. This thesis would surely never have been written without him.

Contents

Chapter 1

Introduction

Packing problems are an issue probably since the dawn of mankind. They have been investigated intensively at least for centuries. One of the arguably most famous ones first popularized by 17th-century-mathematician Johannes Kepler is the following: What is the densest arrangement of equally sized spheres in three dimensions? Allegedly, Kepler came up with it when corresponding with the English mathematician Thomas Harriot who was assigned to study how to stack cannonballs most efficiently by a befriended ship-owner. Kepler conjectured that the most efficient way would be to place the spheres like one would stack oranges (or cannonballs) intuitively (see Fig. 1.1).

Going forward in history, during the industrial revolution other packing problems became of interest. For example when manufacturing clothes, one would like to waste as little fabric as possible when cutting out the pieces the clothes are composed from. The problem can be modeled as a two dimensional strip of infinite length and the goal is to minimize the used length (*strip packing*) and is known to be NP- hard (see the reduction from PARTITION in [5]). Similar problems arise when cutting metal sheets or other materials or - a more every day like problem - when baking cookies and trying to minimize the times one has to roll out the dough.

The increasing globalization made packing problems in the literal sense an issue: To

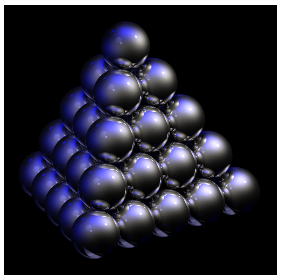


Figure 1.1: Spheres packed according to the Kepler conjecture.

save money, it is important to save space when packing goods into parcels and parcels into ship or plane containers. Saving space would also save resources , e.g. cardboard (lots of online traders are not really good at it these days...) and fuel, and eventually the environment. This is true for a lot of applications of packing problems.

In todays digital era, the world is embraced by a network of cables, many of whom are connecting continents through the oceans. To save money and resources when producing and installing the cables it is a good idea to minimize the diameter of the cables. Since one cable is composed of several cores, this can be modeled as packing equally sized circles into a larger circle, see Fig. 1.2. In the beginning of 2020 another unpleasant

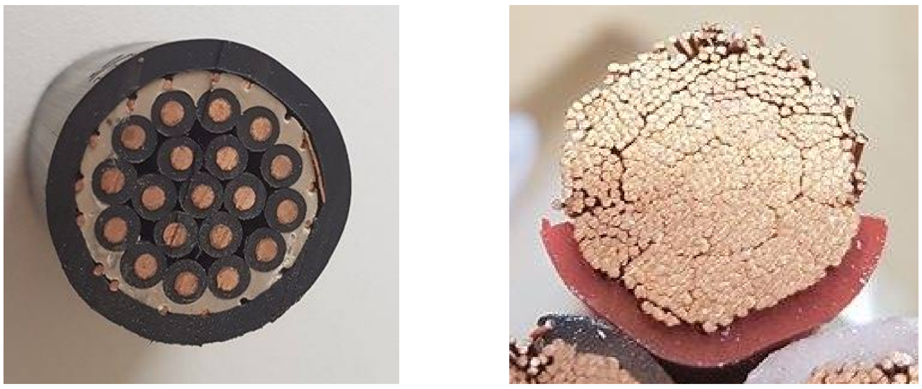


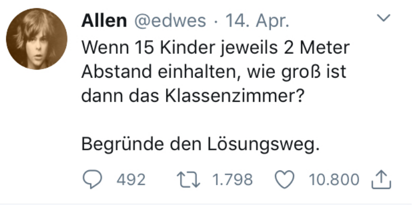
Figure 1.2: Profile of wires.

application of packing equally sized circles emerged. The pairwise distance of people is of high interest to contain the Corona-pandemic. So, asking how many people can be in one room with enough distance between them can be modeled as packing equally sized circles into the shape of the room. This issue and its mathematical formulation has even been discussed on Twitter, see Fig. 1.3.

The last two problems mentioned consider packing equally sized circles, so these problems are related to the two dimensional version of the problem studied by Kepler. It took more than 200 years from Keplers conjecture to a first proof of the two dimensional equivalent of the problem by Thue in 1890. A complete and rigorous proof was given later by Fejes Tóth [23] (see Chapter 2 for details) and the actual three dimensional Kepler conjecture was proven only recently by Hales et al.[29, 30] under the massive usage of computers to e.g. check huge case-distinctions and verify the correctness with theorem provers (we revisit a no-computer part of the proof in Chapter 6).

1.1 This Thesis

This thesis is divided into two parts. The first part is focused on packing congruent objects. We start with packing unit circles, i.e., we study the "corona-problem" of how many people are allowed in a room (see Fig. 1.3). In more detail, we are given a container shape and ask how many unit circles can be packed into it. It is not known, whether this

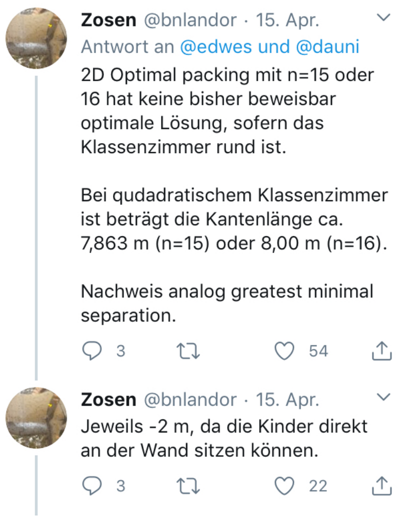


Allen @edwes · 14. Apr.
Wenn 15 Kinder jeweils 2 Meter Abstand einhalten, wie groß ist dann das Klassenzimmer?

Begründe den Lösungsweg.

492 1.798 10.800

(i) How many kids are allowed in one room? From www.twitter.com on 14/04/2020.



Zosen @bnlandor · 15. Apr.
Antwort an @edwes und @dauni
2D Optimal packing mit n=15 oder 16 hat keine bisher beweisbar optimale Lösung, sofern das Klassenzimmer rund ist.

Bei qudadratischem Klassenzimmer ist beträgt die Kantenlänge ca. 7,863 m (n=15) oder 8,00 m (n=16).

Nachweis analog greatest minimal separation.

3 54

Zosen @bnlandor · 15. Apr.
Jeweils -2 m, da die Kinder direkt an der Wand sitzen können.

3 22

(ii) Answer from another user. From www.twitter.com on 15/04/2020.

Figure 1.3

kind of problem is NP-hard. A similar problem is the pallet-loading-problem that asks for the maximum number of rectangles with side lengths a, b that can be packed into a container rectangle with side lengths A, B [45]. The problems have in common that the input is very concise, i.e., in many cases it is only a constant number of numbers (for instance when the container is a circle given by its radius). This concise input makes it hard to show any kind of hardness or give efficient algorithms since the computation time for reduction algorithms or algorithms solving the problems is very limited when aiming for polynomial running time. When the input is larger, more is known: Packing unit circles into a rectilinear polygon with holes is NP-hard[27]. Note, that here the container polygon has many vertices making the input larger. The decision problem if a given set of circles with different radii can be packed into a triangle, rectangle or container circle is also NP-hard [18]. Again, the input is larger since the radius for every input circle is given.

In Chapters 3 and 5, we give PTASs for packing unit circles into different shapes of containers like so called fat parallelograms and triangles (see Definition 3.4 for a precise definition of fatness), circles, fixed simple polygons and so called thick convex containers (see Definition 3.12 for the definition of thickness). We generalize some of the used ideas to pack objects other than circles like fixed or thick polygons into fat parallelograms or triangles under translations only or rigid motions, or spheres into a sphere in 3D. To achieve the generalization in 3D, we revisit a part of the proof of the Kepler conjecture in Chapter 6 (this result has been made public via arXiv [46]). Furthermore, we give a constant factor approximation for packing unit circles into a general triangle in Chapter 4. These results are to our knowledge the first about the approximability, and hence the complexity, of this kind of problems. Extended abstracts of the results in Chapters 3 to 5 have been presented on Workshops [11, 10].

For packing in 3D, Alt and Scharf [9] give constant factor approximation algorithms for the NP-hard problems of packing axis-parallel boxes under translation or convex polyhedra under rigid motions (the hardness follows form the two-dimensional equivalent, see [5]). To our knowledge there is no result to pack objects different from axis-parallel boxes under translation only. In contrast, in 2D, there is a constant factor approximation by Alt et al. to pack convex polygons under translation [6]. In Chapter 7 we give a constant factor approximation for packing unit disks in 3D under translation and, therefore, a first result for packing objects different from boxes under translation. The used techniques lead to the additional result, that not all possible unit disks in 3D can be packed into a container of constant size. This is in contrast to the equivalent two dimensional problem of packing unit line segments since all unit line segments can for instance be packed into a half circle. A preliminary version of Chapter 7 has been presented at a conference [8].

The second part deals with the complexity of some packing problems. Nevertheless, we already give two results concerning the complexity of packing in the preliminaries (Chapter 2) since we use it later: a wide range of packing problems can be formulated as first-order sentences of the reals and hence they are in PSPACE[15] and can be solved in exponential time. We use a similar argument to show that packing a set of polygons under translation into a polygonal container is in NP. In a nutshell, there is a set of linear inequalities that have to be fulfilled when the objects can be packed into the container. These inequalities divide the space of possible solutions into cells and we can nondeterministically guess a vertex in this arrangement by picking the right number of inequalities. Afterwards, we check if the picked vertex corresponds to a valid solution. This result seems to be folklore but we could not find a rigorous written down proof.

In Chapter 8 we study packing polygons into minimum area axis-parallel rectangles or convex containers under translations or allowing rotations by 90°. There is a constant factor approximation by Alt, de Berg and Knauer [5] for packing under translation but it is not known whether a PTAS can exist. We show a weaker result, i.e. that no FPTAS can exist for this problem unless $\mathsf{P} = \mathsf{NP}$. In Chapter 9, we show that the decision problem if a given set of line segments can be packed into a square is NP-hard by a reduction from Partition.

1.2 Further Related Work

Packing has been studied extensively, see e.g. [5] for a survey on complexity and approximability of packing problems,[47] for a survey on different kinds of packing problems and algorithms to tackle them (in German), [33] for a survey of packing circles in the field of operations research or [32, 35] for exemplary results. Also solving packing problems up to optimality for a fixed number of objects is a wide field of research, see e.g. [22] and [42] for overviews of the best known results for numbers partially up to thousands.

Besides the result that packing unit disks or squares into a rectilinear polygon with holes is NP-hard by Fowler, Paterson and Tanimoto [27], it is also known that packing

2x2 squares into an integer-grid polygon is NP-complete [39] (again, the input is large since each grid-cell that is part of the container is given). The framework from Hochbaum and Maas [34] can be used to obtain a PTAS for packing 2x2 squares into an integer-grid polygon. We will use the idea of dividing the container into cells and computing solutions for each cell individually in Chapter 3, however, our approach is much simpler.

For packing disks, the optimal worst case density, i.e., the maximum total area of a set of disks that can always be packed into the container, is known for triangular, square and circular containers by the split-packing-approach by Fekete et al. [26, 25]. Their algorithms subdivide the container and input circles further in a divide and conquer manner.

A very recent result by Abrahamsen, Miltzow and Seiferth shows that many variants of packing polygons or similar objects is $\exists\mathbb{R}$- complete [1]. There is a wide range of other packing problems such as the well studied bin and strip packing, and geometric knapsack problems, see e.g. [36, 2, 41, 19, 3], and variants of those that allow for example to augment resources or ask for online algorithms, see e.g. [41, 24]. There are also more specialized packing problems that have been studied recently, like placing coins on a shelf [7] or minimizing the perimeter of the convex hull [38]. Even though we only mentioned a small number of packing problems and known results, one can already imagine that packing is an interesting and broad field of research. To give a full overview would go beyond the scope of this introduction.

Chapter 2
Preliminaries

In this chapter we introduce basic notations and concepts that we are going to use throughout the thesis and give some preliminary results.

When talking about unit circles, unit spheres, unit disks or unit balls, we always mean that they have radius one. We use the terms circle or sphere for container shapes and disk or ball for the objects packed to emphasize that their interior is of relevance since it has to be disjoint from other packed objects. Vectors will be denoted in boldface to make it easier to distinguish them from numbers.

We start with the two dimensional equivalent of the Kepler conjecture proven by Fejes Tóth [23]. It says that packing unit disks with their center on an equilateral triangular grid with side length two (also called *hexagonal grid*, see Fig. 2.1) yields the highest density, i.e., the portion of space covered by the packed disks is maximal. This packing has a density of $\pi/\sqrt{12} \approx 0.9069$. It follows immediately by a limit argument from the following theorem in [23].

Theorem 2.1. *Let $\mathcal{S}$ be a set of congruent disks and S the total area of the disks in $\mathcal{S}$. Let $\mathcal{R}$ be a convex region with area R such that at least two disks from $\mathcal{S}$ can be packed into $\mathcal{R}$. If all disks in $\mathcal{S}$ can be packed into $\mathcal{R}$, then $S < \frac{\pi}{\sqrt{12}}R$.*

2.1 Model of Computation

In contrast to most literature on geometric algorithms, we do not use the real-RAM-model to analyze our algorithms. Even though the analysis would be easier in some parts, the real-RAM-model has downsides. For instance, one has to be extremely careful when defining the operation set of the real-RAM. Since one operation has unit cost, even a reasonable operation set might enable it to solve NP-complete or even PSPACE-complete problems in polynomial time [48, 31]. Hence, analyzing the complexity of an algorithm in the real-RAM- model is not very meaningful and it is not always obvious how to translate given algorithms to a suitable model like Turing Machines or equivalents. In his paper on shortest path in 3D [44], Papadimitriou tries to avoid this challenge by analyzing

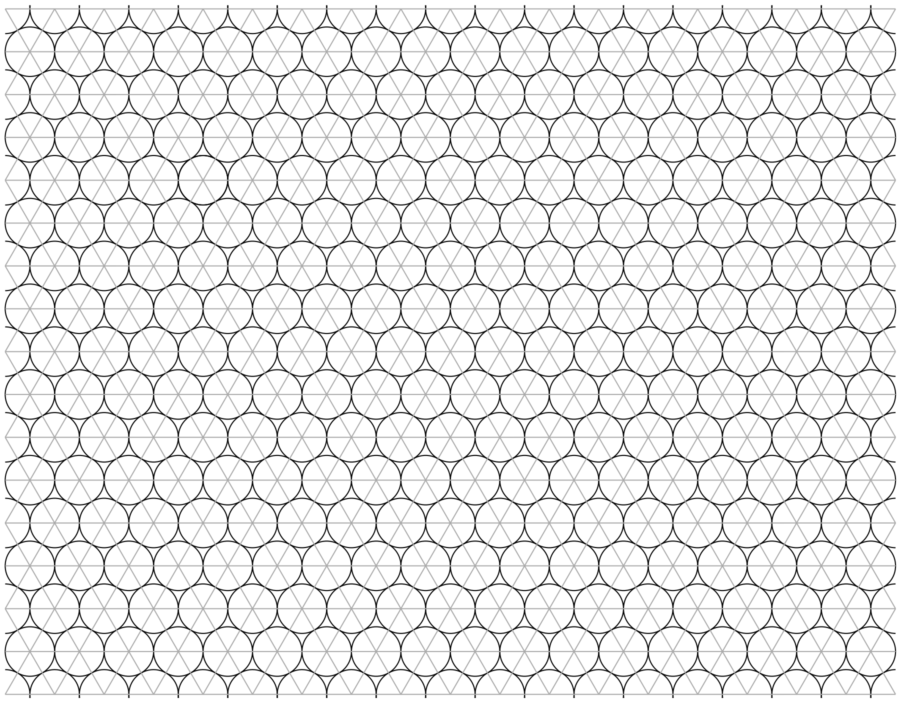


Figure 2.1: Optimal packing of unit disks in the plane.

his algorithm in the bit-model. Unfortunately he leaves gaps and misses details that have later been given in the follow-up paper by Choi, Sellen and Yap[16]. This example shows, how careful the analysis has to be made.

To circumvent the difficulties with the real-RAM-model, we use bit-complexity to analyze our algorithms, i.e., the number of bit operations used. Additionally, this model resembles reality more closely and is equivalent to Turing Machines under polynomial time reductions. Since we have a finite number of bits as input, we can assume that all input numbers are integers. Note that this also allows us to have rational numbers as input represented by their integer nominator and denominator.

2.2 First-Order Formulas, NP, and Packing

2.2.1 First-Order Formulas of the Reals and Packing

One of the main ingredients for some of the algorithms given later is a subroutine to decide if n identical objects can be packed into a given shape. The idea is to formulate the decision problem as first-order formulas with polynomial inequalities as predicates that we interpret over the reals. These formulas are called first-order formulas of the reals.

Assume the shape and the objects to be packed can be described as quantifier free first-order formulas of the reals. We can use these formulas as subformulas to build a first-order formula with no free variables (first-order sentence) representing our packing problem as described later. Then, we can use an algorithm for the general decision problem for the first-order theory of the reals, e.g. the algorithm of Basu et al. [12], to decide whether it is true. Since this approach works in any dimension d, we describe it here for the general d-dimensional case.

Let $\mathrm{c}(x_1, \ldots, x_d)$ be a quantifier free first-order formula that describes a container object, i.e., the formula evaluates to *true* if and only if we instantiate $x_1, \ldots, x_d$ with a d-dimensional point $(p_1, \ldots, p_d)$ that lies in the interior of the container. The atoms of c are polynomial inequalities and equalities with variables $x_1, \ldots, x_d$. Analogously, let $\mathrm{f}(x_1, \ldots, x_d)$ be another quantifier free first-order formula that describes the objects to be packed at some fixed position. In the following, we abbreviate $\exists x_1, \exists x_2, \ldots \exists x_l$ by $(\exists x_1 \ldots x_l)$ and $\forall x_1, \forall x_2, \ldots \forall x_l$ by $(\forall x_1 \ldots x_l)$. For $\mathbf{t} = (t_1, \ldots t_d)$, we use $\exists \mathbf{t}$ and $\forall \mathbf{t}$ as short notation for $(\exists t_1, \ldots t_d)$ and $(\forall t_1, \ldots t_d)$, respectively. The decision problem if n copies of the object described by f can be packed under translation into the object described by c can be formulated with the following formula:

$$\begin{aligned}(\exists \mathbf{t}_1, \ldots, \mathbf{t}_n)\forall \mathbf{x} \bigwedge_{i=1}^{n} (\mathrm{f}(\mathbf{x} - \mathbf{t}_i) \Rightarrow \mathrm{c}(\mathbf{x}))\wedge \\ \bigwedge_{1 \le i < j \le n} (\mathrm{f}(\mathbf{x} - \mathbf{t}_i) \Rightarrow \neg \mathrm{f}(\mathbf{x} - \mathbf{t}_j))\end{aligned} \tag{2.1}$$

Each $\mathbf{t}_i$ consists of d real valued variables and describes the translation vector for the i-th object to be packed. Also, $\mathbf{x}$ consists of d real valued variables and is used to check in the first part of the formula, that a point that is contained in an object is also contained in the container, i.e., the translated objects lie inside the container. The second part ensures that a point $\mathbf{x}$ cannot lie inside two objects at the same time, i.e., the objects to be packed do not overlap.

Let p_1 be the number of polynomial inequalities and equalities in c and a_1 their maximum degree. Let p_2 be the number of polynomial inequalities and equalities in f and a_2 their maximum degree. Let l_1 be the length of a description of c and l_2 be the length of a description of f in some constant size alphabet. Observe that formula (2.1) consists of $n \cdot d + d$ variables, $p_1 + n \cdot p_2$ different polynomials and has a maximum degree of $a = \max(a_1, a_2)$ and length $\mathcal{O}(n(l_1 + l_2) + n^2 l_2)$. From Theorem 1.3.2 in [12] we get the following insights. The truth of such a formula can be decided with $(p_1 + n \cdot p_2)^{(nd+1)(d+1)} a^{\mathcal{O}(nd^2)} \mathcal{O}(n(l_1 + l_2) + n^2 l_2)$ arithmetic and boolean operations. If we can ensure that all coefficients in the equalities and inequalities are integers of bit-size at most s then the bit-size of all output and intermediate integers is bounded by $sa^{\mathcal{O}(nd^2)}$. Let $\mathrm{M}(k)$ be the running time to multiply two k-bit integers. Since multiplication is the most expensive arithmetic operation (in terms of used boolean operations), we get the following theorem.

Theorem 2.2. *If the coefficients in the inequalities and equalities that form the atoms of* c *and* f *are integers with maximum bit-size* s*, we can decide in time*

$$(p_1 + n \cdot p_2)^{(nd+1)(d+1)} a^{\mathcal{O}\left(nd^2\right)} \mathcal{O}\Big(n(l_1 + l_2) + n^2 l_2\Big) \mathrm{M}\Big(s a^{\mathcal{O}\left(nd^2\right)}\Big) \tag{2.2}$$

if n *copies of the object described by* f *can be packed into the container described by* c *under translation.*

If the maximum number of copies of the object described by f that can be packed into the container is at most N for some $N \in \mathbb{N}$, we can find the maximum number in the following way: We perform binary search on the number of objects packed where in each step, we call the decision algorithm just described with n set to the number that is tested in the step. Therefore, the following corollary follows directly form Theorem 2.2.

Corollary 2.3. *If the coefficients in the inequalities and equalities that form the atoms of* c *and* f *are integers with maximum bit-size* s *and the maximum number of (translated) copies of the object described by* f *that can be packed into the object described by* c *is bounded from above by* N*, we can calculate in time*

$$(p_1 + N \cdot p_2)^{(Nd+1)(d+1)} a^{\mathcal{O}\left(Nd^2\right)} \mathcal{O}\Big(N(l_1 + l_2) + N^2 l_2\Big) \mathrm{M}\Big(s a^{\mathcal{O}\left(Nd^2\right)}\Big) \mathcal{O}(\log N)$$

the maximum number of copies of the object described by f *that can be packed into the container described by* c *under translation. If* p_1, p_2, a, d *are constants and* $l_1, l_2 \in \mathcal{O}(s)$*, and assuming* $\mathrm{M}(k) \in \mathcal{O}(k^m)$ *for some constant* $m < 2$*, the running time simplifies to*

$$N^{\mathcal{O}(N)} \mathcal{O}(s \cdot \mathrm{M}(s)). \tag{2.3}$$

Similar results can be obtained when rotations are allowed. We will show exemplary how to obtain them for packing in two dimensions under rigid motions next.

Packing under Rigid Motions As before, $\mathrm{c}(x, y)$ is a quantifier free first-order formula that evaluates to *true* if and only if x, y are instantiated with p, q where the point (p, q) lies inside the container and, analogously, $\mathrm{f}(x, y)$ is a formula describing the object to be packed. The decision problem if n copies of the object described by f can be packed into the container described by c under rigid motions can be formulated with the following formula (recall that we are packing in two dimensions now):

$$\begin{aligned}
&(\exists t_{1,x}, \dots, t_{n,x}, t_{1,y}, \dots, t_{n,y}, c_1, \dots, c_n, s_1, \dots, s_n)(\forall x, y) \\
&\qquad \bigwedge_{i=1}^{n} \Big(c_i^2 + s_i^2 = 1\Big) \\
&\qquad \wedge \bigwedge_{i=1}^{n} (\mathrm{f}(x \cdot c_i - y \cdot s_i - t_{i,x}, y \cdot c_i + x \cdot s_i - t_{i,y}) \Rightarrow \mathrm{c}(x, y)) \\
&\qquad \wedge \bigwedge_{1 \le i < j \le n} (\mathrm{f}(x \cdot c_i - y \cdot s_i - t_{i,x}, y \cdot c_i + x \cdot s_i - t_{i,y}) \Rightarrow \\
&\qquad\qquad \neg \mathrm{f}(x \cdot c_j - y \cdot s_j - t_{j,x}, y \cdot c_j + x \cdot s_j - t_{j,y})).
\end{aligned} \tag{2.4}$$

Again, $(t_{i,x}, t_{i,y})$ describes the translation of the i-th object. c_i and s_i are the cosine and sine, respectively, of the angle that the i-th object is rotated by (the equation $c_i^2 + s_i^2 = 1$ ensures that they are valid values). The rest of the formula is analogous to the one for packing without rotation.

As before, let p_1 be the number of polynomial inequalities and equalities in c and a_1 their maximum degree. Let p_2 be the number of polynomial inequalities and equalities in f and a_2 their maximum degree. Let l_1 be the length of a description of c and l_2 be the length of a description of f in some constant size alphabet. Observe that formula (2.4) consists of $4n+2$ variables, $n + p_1 + n \cdot p_2$ different polynomials and has a maximum degree of $a = \max(a_1, a_2, 2)$ and length $\mathcal{O}(n(l_1 + l_2) + n^2 l_2)$. From Theorem 1.3.2 in [12] we get the following insights. As before, the truth of such a formula can be decided with $(n + p_1 + n \cdot p_2)^{(4n+3)\cdot 3} a^{\mathcal{O}(n^2)} \mathcal{O}(n(l_1 + l_2) + n^2 l_2)$ arithmetic and boolean operations. Again, if we can ensure that all coefficients in the equalities and inequalities are integers of bit-size at most s then the bit-size of all output and intermediate integers is bounded by $sa^{\mathcal{O}(n^2)}$. Let $\mathrm{M}(k)$ be the running time to multiply two k-bit integers. Since multiplication is the most expensive arithmetic operation (in terms of used boolean operations), we get the following theorem.

Theorem 2.4. *If the coefficients in the inequalities and equalities that form the atoms of* c *and* f *are integers with maximum bit-size* s*, we can decide in time*

$$(n + p_1 + n \cdot p_2)^{\mathcal{O}(n)} a^{\mathcal{O}(n^2)} \mathcal{O}\Big(n(l_1 + l_2) + n^2 l_2\Big) \mathrm{M}\Big(sa^{\mathcal{O}(n^2)}\Big) \tag{2.5}$$

if n *copies of the object described by* f *can be packed into the container described by* c *under rigid motions.*

It should be noted that depending on the objects and container it might be necessary to solve large first- order formulas of the reals. Therefore, the algorithms are not really applicable in practice.

Similar to before, we can use binary search to compute the maximum number of copies of the object that can be packed under rigid motions into the container if we have an upper bound N for this number. Similar to Corollary 2.3, the running time would be exponential in N, but polynomial in the encodings of f and c.

A formula similar to (2.4) can also be obtained for higher dimensions. It requires inventing more variables beeing entries of a rotation matrix and additionally a polynomial number of inequalities ensuring that the variables indeed are the entries of a rotation matrix. Even though the formulas will get more complicated, it is still only a constant factor if the dimension is fixed. So, the running times are analogous.

In the remainder of this thesis, we will refer to solving packing problems with the approach discussed above as solving them with the *exact algorithm*. Whether we use the approach for packing under translation only or under rigid motions will become clear from the context.

2.2.2 Packing Polygons and NP

In the following, we will show that deciding whether a given set $\mathcal{P}$ of n polygons can be packed into a polygonal container C under translation is in NP. This seems to be folklore, but we could not find a rigorous written down proof of that. We believe that there are many other ways to prove this.

We will use the following intuitive definition in the proof in order to make it easier to read.

Definition 2.5. *Let ψ be a quantifier free first-order formula of the reals using variables $x_1, \dots, x_m$. We say a point $\mathbf{r} \in \mathbb{R}^m$* fulfills *$\psi$, if and only if ψ evaluates to true when replacing $x_1, \dots, x_m$ by the coordinates of $\mathbf{r}$ in this order.*

First, assume there is a quantifier free formula $\chi(A, B, C, D, \mathbf{t}_1, \mathbf{t}_2)$ that is true if and only if the line segments $\overline{AB}$ translated by $\mathbf{t}_1$ and $\overline{CD}$ translated by $\mathbf{t}_2$ do not cross and the atoms of $\chi(A, B, C, D, \mathbf{t}_1, \mathbf{t}_2)$ are a constant number of linear non-strict $(\leq, \geq)$ inequalities. By *crossing* we mean that they have exactly one point in common in their relative interiors. Furthermore, we assume that $P_i \in \mathcal{P}$ is given by the sequence of its vertices $\mathbf{v}_{i,1}, \dots, \mathbf{v}_{i,k_i}, \mathbf{v}_{i,k_i+1}$ where $\mathbf{v}_{i,1} = \mathbf{v}_{i,k_i+1}$ and analogously also C is given by the sequence of its vertices $\mathbf{v}_{C,1}, \dots, \mathbf{v}_{C,k_C}, \mathbf{v}_{C,k_C+1}$. Then, the following first-order formula of the reals is true, if the polygons in $\mathcal{P}$ can be packed into C under translation:

$$(\exists \mathbf{t}_1, \dots, \mathbf{t}_n) \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \bigwedge_{l=1}^{k_i} \bigwedge_{m=1}^{k_j} \chi(\mathbf{v}_{i,l}, \mathbf{v}_{i,l+1}, \mathbf{v}_{j,m}, \mathbf{v}_{j,m+1}, \mathbf{t}_i, \mathbf{t}_j)$$
$$\wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k_i} \bigwedge_{l=1}^{k_C} \chi(\mathbf{v}_{i,j}, \mathbf{v}_{i,j+1}, \mathbf{v}_{C,l}, \mathbf{v}_{C,l+1}, \mathbf{t}_i, \mathbf{0}). \quad (2.6)$$

Observe that there are values for $\mathbf{t}_1, \dots, \mathbf{t}_n$ that fulfill Eq. (2.6) even though they do not give valid packings, namely when objects are packed into other objects or placed outside the container. However, the inequalities in Eq. (2.6) form an arrangement of halfspaces $\mathcal{A}$ with the following properties:

- Let $\mathbf{t}_1, \dots, \mathbf{t}_n$ be a set of translations which gives a valid packing. Let $\mathcal{C}$ be the cell in $\mathcal{A}$ containing the point corresponding to $\mathbf{t}_1, \dots, \mathbf{t}_n$. Then, all points in $\mathcal{C}$ correspond to valid packings. This also holds for points on the boundary of $\mathcal{C}$ since the polygons are allowed to touch (share points on their boundary) or touch the boundary of the container polygon.

- The cells in $\mathcal{A}$ containing points that correspond to translations giving valid packings are bounded. This follows immediately from the fact that the polygons need to be packed inside a bounded container.

- The number of halfspaces in $\mathcal{A}$ is bounded by a polynomial in the total number of vertices of objects in $\mathcal{P}$ and vertices of C since $\chi(A, B, C, D, \mathbf{t}_1, \mathbf{t}_2)$ is build from a constant number of linear inequalities.

From the first two properties, it follows that there is a vertex in $\mathcal{A}$ representing a valid packing if the objects can be packed into the container. Due to the third property, we can guess nondeterministically such a vertex in $\mathcal{A}$ and check afterwards if it gives a valid packing in order to decide if the polygons in $\mathcal{P}$ can be packed into C nondeterministically. In more detail, we guess $2n$ inequalities that define a vertex in $\mathcal{A}$, i.e. by replacing the $\leq, \geq$ by $=$, and solving the resulting system of linear equalities. Solving a system of linear equalities in polynomial time in the bit model is not trivial. However, with a careful analysis it can be shown to be possible with a variant of Gaussian elimination [21, 49].

Afterwards, we check if the packing corresponding to the point that is the guessed vertex of $\mathcal{A}$ is valid, i.e. if all polygons lie inside the container and no to polygons intersect in their interior (see e.g. algorithms in 2.4 and 2.5 in [13] and the references therein on how to do that). In order to ensure polynomial running time with this procedure, we will show that indeed there exists $\chi(A, B, C, D, \mathbf{t}_1, \mathbf{t}_2)$ build from a constant number of linear inequalities as assumed and furthermore the coefficients in the linear inequalities are polynomials of bounded degree in the coordinates of A, B, C, D.

To obtain $\chi(A, B, C, D, \mathbf{t}_1, \mathbf{t}_2)$, we first generate equalities describing the lines supporting the line segments of the form $mx + n = y$. If the slope of the two line segments is equal, we set $\chi(A, B, C, D, \mathbf{t}_1, \mathbf{t}_2)$ to *true* and we are done.

Otherwise, we get from the equalities representing the supporting lines formulas for the coordinates of the intersection point of the two lines. Observe that m is independent of $\mathbf{t}_1, \mathbf{t}_2$ and therefore, these formulas are linear in the coordinates of $\mathbf{t}_1$ and $\mathbf{t}_2$. The line segments do not cross, if and only if

the x-value of the intersection point does not lie in the interior of the interval of the x-values of A and B translated by $\mathbf{t}_1$,

or the x-value of the intersection point does not lie in the interior of the interval of the x-values of C and D translated by $\mathbf{t}_2$.

or the y-value of the intersection point does not lie in the interior of the interval between the $y-$values of A and B translated by $\mathbf{t}_1$,

or the y-value of the intersection point does not lie in the interior of the interval between the $y-$values of C and D translated by $\mathbf{t}_2$.

This can easily be expressed as a disjunction of linear non-strict inequalities with coefficients being polynomials of the coordinates of A, B, C, D of bounded degree. This concludes the proof of the following theorem.

Theorem 2.6. *Packing polygons under translation into a polygonal container is in* NP*.*

I

Packing Equal Objects

Chapter 3

PTAS for Packing into Fat Parallelograms and Triangles

In this chapter, we start with packing unit disks into fat parallelograms and fat triangles (see Definition 3.4 for the definition of fatness) and show afterwards how to generalize the used techniques to pack other objects. We will give a PTAS for all mentioned problems, i.e., the algorithms achieve a $(1-\varepsilon)$ approximation ratio and their running time is polynomial in the input size for any fixed ε. Recall that the complexity measure used to analyze the running time of the algorithms is the bit-complexity, i.e., the number of bit-operations (which is within a constant factor of operations in any other constant size alphabet). The idea for the PTAS is to divide the container into small equal cells and compute optimal solutions for these cells. The solution returned is the sum of the solutions for the small cells. Even though this idea is seemingly simple, the algorithms give a first result on the approximability of these problems and the challenge lies in working out the details and proving their approximation factors and running times.

3.1 Preliminaries

In this section, we state some minor findings that will be useful to prove the main results of this chapter later on.

When analyzing our algorithms in the remainder of this section, we will use the following Lemma to get an upper and a lower bound on the maximum number of unit disks that can be packed into a parallelogram. Recall that unit disks have radius one.

Lemma 3.1. *Let n be the maximum number of unit disks that can be packed into a* *parallelogram with side lengths a, b and smaller angle γ. If $n \geq 2$ then*

$$n > \frac{ab\sin^2\gamma}{16}$$

and

$$\frac{1}{\sqrt{12}} ab\sin\gamma > n. \tag{3.1}$$

Proof. The second inequality follows immediately from Theorem 2.1. For the first inequality, consider a packing on a rhombus-grid with side length $\frac{2}{\sin\gamma}$ into the container parallelogram as shown in Fig. 3.1 ($\frac{2}{\sin\gamma}$ is the smallest side length of a rhombus grid such that unit disks can be packed onto it). This packing has size

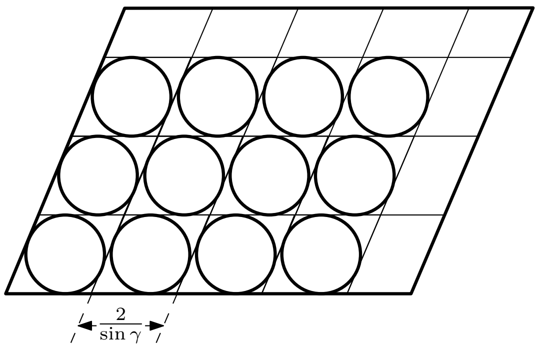


Figure 3.1: Lower Bound on the number of unit disks that can be packed into a parallelogram.

$$\left\lfloor \frac{a\sin\gamma}{2} \right\rfloor \left\lfloor \frac{b\sin\gamma}{2} \right\rfloor.$$

It is known that $\lfloor x \rfloor > \frac{x}{2}$ for $x > 1$. Since at least two unit disks can be packed into the container, we know that $\frac{a\sin\gamma}{2} > 1$ and $\frac{b\sin\gamma}{2} > 1$. This gives

$$\left\lfloor \frac{a\sin\gamma}{2} \right\rfloor \left\lfloor \frac{b\sin\gamma}{2} \right\rfloor > \frac{ab\sin^2\gamma}{16},$$

and completes the proof. □

Therefore, to apply the upper bound given in Lemma 3.1 when packing into a parallelogram with side lengths a, b and smaller angle γ we have to make sure that at least two unit disks can be packed into the parallelogram. We will use the following observation in the analysis of our algorithm to justify the application of this lemma. See Fig. 3.2 for illustration.

Observation 3.2. *Let C be a parallelogram with side length w, h and smaller angle γ where $w \geq \frac{2}{\sin\gamma}$ and $h \geq \frac{4}{\sin\gamma}$. Then, at least two unit disks can be packed into C.*

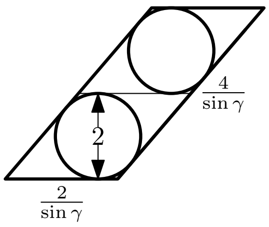


Figure 3.2: Two unit disks can be packed into a parallelogram with side lengths $\frac{2}{\sin\gamma}$ and $\frac{4}{\sin\gamma}$.

The side lengths given in the observation above are obviously not optimal but they are good enough for our purpose.

In the approximation algorithms given in the remainder of this problem, it is sometimes necessary to calculate rounded down values of fractions, where the numerator is of the form $f_1 + \sqrt(f_2)$ and f_1, f_2 and the denominator are polynomials with integer coefficients of bounded degree. We can calculate these values exactly by the following observation.

Observation 3.3. *Let* $a_1, \ldots, a_k$ *be a set of integers. Let* $\mathrm{f}_1(x_1,\ldots,x_k), \mathrm{f}_2(x_1,\ldots,x_k)$, $\mathrm{g}(x_1,\ldots,x_k)$ *be polynomials with integer coefficients of constant maximum degree. Assume* $\mathrm{g}(x_1,\ldots,x_k) > 0$. *Then, we can calculate* $\left\lfloor \frac{\mathrm{f}_1(a_1,\ldots,a_k)+\sqrt{\mathrm{f}_2(a_1,\ldots,a_k)}}{\mathrm{g}(a_1,\ldots,a_k)} \right\rfloor$ *in time polynomial in the total length of the bit-representation of* $a_1, \ldots, a_k$.

Proof. Observe that $\mathrm{f}_1(a_1,\ldots,a_k), \mathrm{f}_2(a_1,\ldots,a_k), \mathrm{g}(a_1,\ldots,a_k)$ are integers and can be computed with standard computer arithmetic in polynomial time. Assume, we know the integer part s of $\sqrt{\mathrm{f}_2(a_1,\ldots,a_k)}$, i.e., $s^2 \le \mathrm{f}_2(a_1,\ldots,a_k) < (s+1)^2$. Then, we can compute $\left\lfloor \frac{\mathrm{f}_1(a_1,\ldots,a_k)+\sqrt{\mathrm{f}_2(a_1,\ldots,a_k)}}{\mathrm{g}(a_1,\ldots,a_k)} \right\rfloor$ in polynomial time with standard computer arithmetic algorithms [14] since $\left\lfloor \frac{\mathrm{f}_1(a_1,\ldots,a_k)+\sqrt{\mathrm{f}_2(a_1,\ldots,a_k)}}{\mathrm{g}(a_1,\ldots,a_k)} \right\rfloor = \left\lfloor \frac{\mathrm{f}_1(a_1,\ldots,a_k)+s}{\mathrm{g}(a_1,\ldots,a_k)} \right\rfloor$. Since s can also be calculated with standard computer arithmetic [14], we can compute $\left\lfloor \frac{\mathrm{f}_1(a_1,\ldots,a_k)+\sqrt{\mathrm{f}_2(a_1,\ldots,a_k)}}{\mathrm{g}(a_1,\ldots,a_k)} \right\rfloor$ in polynomial time. □

3.2 PTAS for Packing Unit-Disks into Fat Parallelograms and Triangles

In this subsection, we will give PTASs to pack unit disks into fat parallelograms or triangles, i.e., additional to the input container we are given an ε with $0 < \varepsilon < 1$ and want to compute solutions of size at least $(1-\varepsilon)$ times the size of an optimal solution in time polynomial in the input size for fixed ε. The next definition specifies what we mean by fat.

Definition 3.4 (α_0-fatness)**.** *For a constant α_0 with $0 < \alpha_0 \leq \frac{\pi}{2}$, we call parallelograms and triangles α_0-*fat *iff all inner angles are at least α_0.*

In the following, we assume that α_0 is some fixed constant and call α_0-fat parallelograms and triangles just *fat*.

The basic idea of the algorithms given in this section is similar to the approach by Hochbaum and Maass [34]: We divide the container into smaller cells such that we can compute the optimal solution inside one cell in polynomial time for fixed ε. The solution returned is the sum of the partial solutions in the cells. To calculate the optimal solution in a grid-cell, we will use the approach discussed in Section 2.2.1.

To guarantee that the running time of the approach just described is polynomial, we would like to have among others three properties:

P1 The cells are not too big such that there is a constant upper bound that depends only on ε. See Corollary 2.3 for how this upper bound effects the running time (There, this upper bound is called N).

P2 There is only a polynomial number of differently shaped cells on the boundary of the container, since we need to run the exact algorithm discussed in Section 2.2.1 for each cell-shape.

P3 The formulas describing the cell-shapes are not too complicated, meaning that the number of different polynomials and the maximum degree are constants and that the length of the formula is a linear in the input-length. See Corollary 2.3 for the resulting running time.

Note that one could adapt the framework given by Hochbaum and Maass to approximate the maximum number of unit disks that can be packed into a fat parallelogram. Such an adaptation would also need to fulfill P1 to P3 despite others and hence, our algorithm is much simpler.

The following lemma will be the key-ingredient for the analysis of our algorithms for fat parallelograms and triangles.

Lemma 3.5. *Let $\mathrm{opt}_\gamma(a, b)$ be the maximum number of unit disks that can be packed into a parallelogram with side lengths a, b and smaller angle γ. Let $0 < \varepsilon < 1$.*

If $a \geq \frac{37}{\varepsilon \sin^2 \gamma}$ and $b \geq \frac{2}{\sin \gamma}$ then

$$\frac{\mathrm{opt}_\gamma(a, b)}{\mathrm{opt}_\gamma\left(a + \frac{2}{\sin \gamma}, b\right)} > 1 - \varepsilon.$$

If $a, b \geq \frac{37}{\varepsilon \sin^2 \gamma}$ then

$$\frac{\mathrm{opt}_\gamma(a, b)}{\mathrm{opt}_\gamma\left(a + \frac{2}{\sin \gamma}, b + \frac{2}{\sin \gamma}\right)} > 1 - \varepsilon.$$

Proof. We start with proving the first inequality. Observe that $\mathrm{opt}_\gamma(a + 2/\sin\gamma, b) \geq \mathrm{opt}_\gamma(a, b) > 1$ since $a > 4/\sin\gamma$ and $b \geq 2/\sin\gamma$ (see Observation 3.2). Consider an optimal packing into the enlarged parallelogram with side lengths $a+2/\sin\gamma, b$ together with a smaller parallelogram with side lengths a, b where all sides except one are aligned with the enlarged parallelogram (see Fig. 3.3i). Observe that all disks in the optimal packing that are not contained in the smaller parallelogram are contained in a parallelogram with side lengths $4/\sin\gamma, b$. Hence, the following holds

$$\begin{aligned}\frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma\left(a+\frac{2}{\sin\gamma},b\right)} &\geq \frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma(a,b)+\mathrm{opt}_\gamma\left(\frac{4}{\sin\gamma},b\right)}\\ &\geq 1-\frac{\mathrm{opt}_\gamma\left(\frac{4}{\sin\gamma},b\right)}{\mathrm{opt}_\gamma(a,b)}\\ &\geq 1-\frac{\frac{4}{\sin\gamma}\cdot b\cdot\sin\gamma\cdot\frac{1}{\sqrt{12}}}{ab\cdot\sin^2\gamma\cdot\frac{1}{16}},\end{aligned}$$

where the last inequality follows from Lemma 3.1. Further rearranging gives

$$\begin{aligned}\frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma\left(a+\frac{2}{\sin\gamma},b\right)} &\geq 1-\frac{4\cdot 16}{\sqrt{12}}\cdot\frac{1}{a\sin^2\gamma}\\ &\geq 1-\frac{64}{\sqrt{12}}\cdot\frac{\sin^2\gamma\cdot\varepsilon}{37\cdot\sin^2\gamma}>1-\varepsilon,\end{aligned}$$

by the definition of a.

The proof of the second inequality is similar. Consider an optimal packing of unit disks into a parallelogram with side lengths $a + 2/\sin\gamma, b + 2/\sin\gamma$ together with a smaller parallelogram with side lengths a, b aligned with two sides of the enlarged parallelogram as shown in 3.3ii. All disks packed into the enlarged parallelogram that are not contained in the smaller parallelogram, are contained in two parallelograms with side lengths $a +$

$2/\sin\gamma, 4/\sin\gamma$ and $4/\sin\gamma, b + 2/\sin\gamma$, respectively. Hence, the following holds

$$\begin{aligned}
&\frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma}, b+\frac{2}{\sin\gamma}\Big)} \\
&\geq \frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma(a,b)+\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma},\frac{4}{\sin\gamma}\Big)+\mathrm{opt}_\gamma\Big(\frac{4}{\sin\gamma}, b+\frac{2}{\sin\gamma}\Big)} \\
&\geq 1-\frac{\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma},\frac{4}{\sin\gamma}\Big)+\mathrm{opt}_\gamma\Big(\frac{4}{\sin\gamma}, b+\frac{2}{\sin\gamma}\Big)}{\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma}, b+\frac{2}{\sin\gamma}\Big)} \\
&\geq 1-\frac{\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma},\frac{4}{\sin\gamma}\Big)}{\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma}, b\Big)}-\frac{\mathrm{opt}_\gamma\Big(\frac{4}{\sin\gamma}, b+\frac{2}{\sin\gamma}\Big)}{\mathrm{opt}_\gamma\Big(a, b+\frac{2}{\sin\gamma}\Big)} \\
&\geq 1-\frac{\Big(a+\frac{2}{\sin\gamma}\Big)\frac{4}{\sin\gamma}\cdot\sin\gamma\cdot\frac{1}{\sqrt{12}}}{\Big(a+\frac{2}{\sin\gamma}\Big)b\cdot\sin^2\gamma\cdot\frac{1}{16}}-\frac{\frac{4}{\sin\gamma}\Big(b+\frac{2}{\sin\gamma}\Big)\cdot\sin\gamma\cdot\frac{1}{\sqrt{12}}}{a\Big(b+\frac{2}{\sin\gamma}\Big)\cdot\sin^2\gamma\cdot\frac{1}{16}},
\end{aligned}$$

where the last inequality follows from Lemma 3.1. Further rearranging gives

$$\begin{aligned}
&\frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma\Big(a+\frac{2}{\sin\gamma}, b+\frac{2}{\sin\gamma}\Big)} \\
&\geq 1-\frac{4\cdot 16}{\sqrt{12}}\left(\frac{1}{b\sin^2\gamma}+\frac{1}{a\sin^2\gamma}\right) \\
&\geq 1-\frac{64}{\sqrt{12}}\cdot 2\cdot\frac{\sin^2\gamma\cdot\varepsilon}{37\cdot\sin^2\gamma} > 1-\varepsilon,
\end{aligned}$$

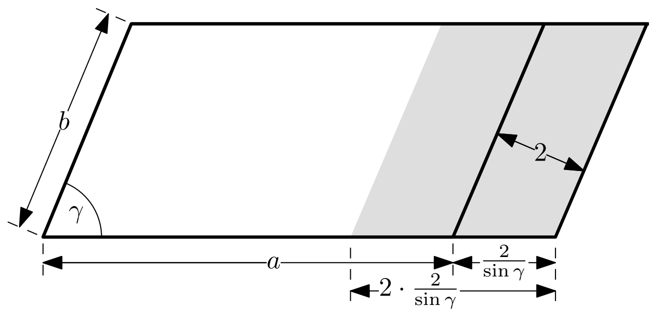
by the definition of a and b. □

3.2.1 Packing into Fat Parallelograms

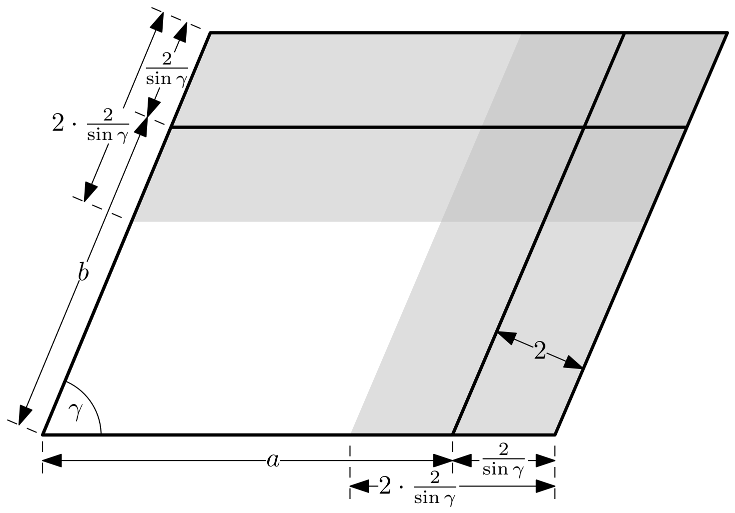
The idea of the PTAS in more detail is as follows: We divide the container parallelogram by a rhombus-grid with grid-lines parallel to the boundaries of the container-parallelogram into parallelogram-shaped cells. The container-parallelogram is given by the length of the base parallel to the x-axis a and a point (b_x, b_y) with $a, b_x, b_y \geq 0$. We assume that the smaller angle γ of the parallelogram is at least α_0, i.e., the parallelogram is fat (see Definition 3.4). Let $b = \sqrt{b_x^2 + b_y^2}$ be the side length of the container different from a. We will use b and γ in the following only for notation and will never calculate these values explicitly as we will see later.

We place the grid with side length $m \in \mathcal{O}(1/\varepsilon)$ such that it is aligned with the left and bottom boundary of the container parallelogram. See Fig. 3.4 for illustration.

Observe that we create only 4 different cell-shapes in this way. We calculate the optimal solution for every occurring cell-shape with the algorithm discussed in Section 2.2.1. The returned result is the sum of the optimal values multiplied by the number of occurrences of the corresponding cell-shape.



(i) All disks packed into the larger parallelogram with side lengths $a + 2/\sin\gamma, b$ that are not contained in the smaller parallelogram with side lengths a, b are contained in the grey parallelogram.



(ii) All disks packed into the larger parallelogram with side lengths $a + 2/\sin\gamma, b + 2/\sin\gamma$ that are not contained in the smaller parallelogram with side lengths a, b are contained in the two grey parallelograms.

Figure 3.3

In the following, let $\text{opt}_\gamma(v, w)$ denote the maximum number of unit disks that can be packed into a parallelogram with side lengths v, w and smaller angle γ. For ease of notation, we use the mod-operator for real values in the same way as for integers: Let $x, y, r \in \mathbb{R}$ with $x = \left\lfloor \frac{x}{y} \right\rfloor \cdot y + r$. Then $x \bmod y = r$.

Algorithm 3.1 summarizes the idea of the algorithm just described. We assume that all input-numbers are given as fractions of integers, i.e. an input-number x is given as (x_n, x_d) with $x = \frac{x_n}{x_d}$ and $x_n, x_d \in \mathbb{Z}_+$. For the calculation of the side length of the rhombus-grid m that we use to partition the container, we would like to use $\sin\alpha_0$. Since α_0 is a constant, there exists $s > 0$ with $s \leq \sin\alpha_0$ and $s = \frac{s_n}{s_d}$ with $s_n, s_d \in \mathbb{N}_+$.

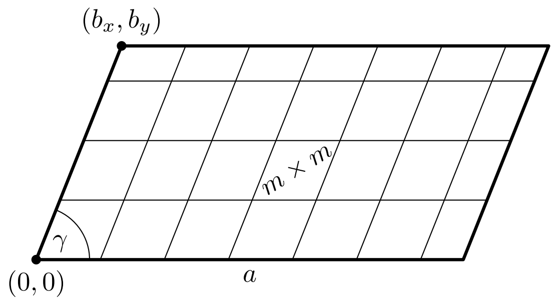


Figure 3.4: Parallelogram divided into cells defined by rhombus-grid with side length m.

Algorithm 3.1:

Input: Number $a \in \mathbb{Q}_+$, point $(b_x, b_y) \in \mathbb{Q}_+^2$, parameter $\varepsilon \in \mathbb{Q}_+$
Output: Nonnegative integer n

```
1 $m := \frac{37 s_d^2}{s_n^2 \cdot \varepsilon}$;
2 Compute the following values with the exact algorithm (see Section 2.2.1):
      (i) $n_1 = \text{opt}_\gamma(m, m)$;
      (ii) $n_2 = \text{opt}_\gamma(a \bmod m, m)$;
      (iii) $n_3 = \text{opt}_\gamma(m, b \bmod m)$;
      (iv) $n_4 = \text{opt}_\gamma(a \bmod m, b \bmod m)$;
3 Calculate $n = \lfloor \frac{a}{m} \rfloor \lfloor \frac{b}{m} \rfloor \cdot n_1 + \lfloor \frac{b}{m} \rfloor \cdot n_2 + \lfloor \frac{a}{m} \rfloor \cdot n_3 + n_4$;
4 return $n$
```

To show that Algorithm 3.1 is a PTAS we have to show that the algorithm computes indeed a $(1 - \varepsilon)$-approximation and that it can be implemented with running time polynomial in the input size for fixed ε. We start with analyzing the running time.

Lemma 3.6. *Algorithm 3.1 can be implemented with polynomial running time for fixed ε.*

Proof. As mentioned earlier, we will not calculate b or γ explicitly and will show later how to calculate the values depending on b and γ. We can store m as a fraction $\frac{m_n}{m_d}$, where $m_n = 37 \cdot \varepsilon_d \cdot s_d^2$ and $m_d = s_n^2 \cdot \varepsilon_n$ can obviously be calculated in constant time for fixed ε.

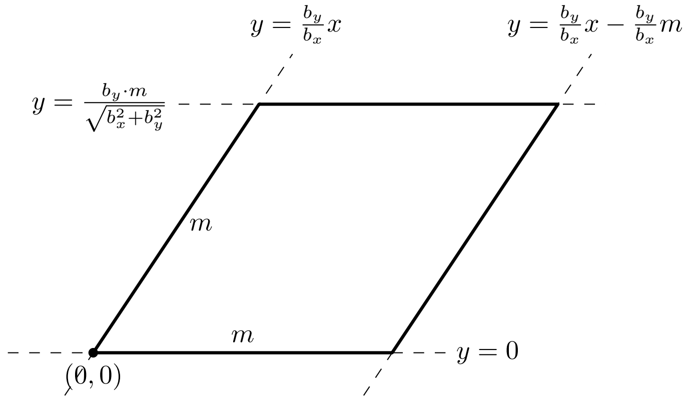
As described earlier in P1 to P3, we need to ensure that there exists an upper bound on the number of unit disks that can be packed into a cell, and the formulas describing the objects to be packed and the cells are not too complex.

The number of unit disks that can be packed in any of the cells is trivially upper bounded by $\frac{m^2}{\pi} = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$. The objects to be packed, i.e., unit disks, can easily be described with $\mathrm{f}(x, y) = x^2 + y^2 < 1$. In the following, we derive formulas for the different cell-shapes.

A rhombus with side length m and smaller angle γ can be seen in Fig. 3.5. We obtain the following formula for the container-shape:

$$\mathrm{c}_1(x, y) : y > 0 \wedge y < \frac{b_y}{b_x}x \wedge y > \frac{b_y}{b_x}x - \frac{b_y \cdot m_n}{b_x \cdot m_d} \wedge y < \frac{b_y \cdot m_n}{m_d\sqrt{b_x^2 + b_y^2}}. \tag{3.2}$$

Eq. (3.2) can obviously transformed such that all coefficients are integers which results



$y = \frac{b_y}{b_x}x$
$y = \frac{b_y}{b_x}x - \frac{b_y}{b_x}m$
$y = \frac{b_y \cdot m}{\sqrt{b_x^2+b_y^2}}$
m
m
$y = 0$
$(0, 0)$

Figure 3.5: Rhombus with side length m and equalities defining its boundaries.

partially in quadratic inequalities. Similarly, a parallelogram with side lengths m and $b \bmod m$, and smaller angle γ can be described with the following formula (see Fig. 3.6):

$$\mathrm{c}_2(x, y) : y < b_y \wedge y > \left\lfloor \frac{m_d\sqrt{b_x^2 + b_y^2}}{m_n} \right\rfloor \frac{b_y m_n}{\sqrt{b_x^2 + b_y^2}} \wedge y < \frac{b_y}{b_x}x \wedge y > \frac{b_y}{b_x}x - \frac{b_y m_n}{b_x m_d}. \tag{3.3}$$

Observe that $\left\lfloor \frac{m_d\sqrt{b_x^2+b_y^2}}{m_n} \right\rfloor = \left\lfloor \frac{m_d\sqrt{b_{x,n}^2 b_{y,d}^2 + b_{y,n}^2 b_{x,d}^2}}{m_n b_{x,d} b_{y,d}} \right\rfloor$, where $b_x = \frac{b_{x,n}}{b_{x,d}}, b_y = \frac{b_{y,n}}{b_{y,d}}$ with $b_{x,n}, b_{x,d}, b_{y,n}, b_{y,d} \in \mathbb{N}$. Therefore, we can calculate $\left\lfloor \frac{m_d\sqrt{b_x^2+b_y^2}}{m_n} \right\rfloor$ exactly by Observation 3.3. So, we can easily rewrite Eq. (3.3) such that all coefficients are integers as for the rhombus-shaped cell. We do the same for the other two cell-types,i.e., cells on the right boundary of the container-parallelogram and the cell in the upper right corner:

$$c_3(x, y) : y > 0 \wedge y < \frac{b_x}{b_y}x \wedge y < \frac{b_y}{b_x}x - \frac{b_y}{b_x}\left\lfloor \frac{a \cdot m_d}{m_n} \right\rfloor \frac{m_n}{m_d} \wedge y > \frac{b_y}{b_x}x - \frac{b_y}{b_x}a, \tag{3.4}$$

$$\begin{aligned} c_4(x, y) : & y < b_y \wedge y > \left\lfloor \frac{m_d\sqrt{b_x^2 + b_y^2}}{m_n} \right\rfloor \frac{b_y m_n}{\sqrt{b_x^2 + b_y^2}} \\ & \wedge y < \frac{b_y}{b_x}x - \frac{b_y}{b_x}\left\lfloor \frac{a \cdot m_d}{m_n} \right\rfloor \frac{m_n}{m_d} \wedge y > \frac{b_y}{b_x}x - \frac{b_y}{b_x}a. \end{aligned} \tag{3.5}$$

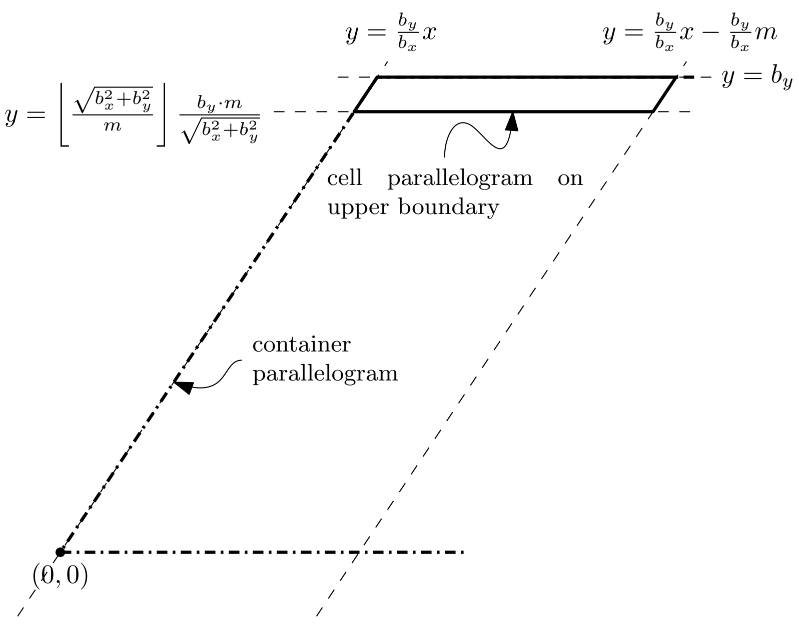

$y = \frac{b_y}{b_x}x$
$y = \frac{b_y}{b_x}x - \frac{b_y}{b_x}m$
$y = b_y$
$y = \left\lfloor \frac{\sqrt{b_x^2+b_y^2}}{m} \right\rfloor \frac{b_y \cdot m}{\sqrt{b_x^2+b_y^2}}$
cell parallelogram on upper boundary
container parallelogram
$(0,0)$

Figure 3.6: Parallelogram with side lengths $m, b \bmod m$ and equalities defining its boundaries.

Summarizing, each cell can be described by a formula consisting of 4 quadratic inequalities with only integer coefficients. We observed earlier that in each of the cells at most $m^2 = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ unit disks can be packed and a unit disk can be described with one quadratic inequality. Let l be the bit length of the largest integer in the input (remember that $b_x, b_y, a, \varepsilon \in \mathbb{Q}$, i.e., they consist of two integers). So, the bit length of the largest coefficient that we create and the total length of the formula when rearranging the inequalities such that all coefficients are integers is bounded by $\mathcal{O}(l)$. By Corollary 2.3, we can compute the optimal solution for one cell in time

$$\left(\frac{1}{\varepsilon}\right)^{\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)} \mathcal{O}(l \cdot \mathrm{M}(l))$$

which is polynomial in the input size for fixed ε. Hence, n_1, n_2, n_3, n_4 in Algorithm 3.1 can be calculated in time polynomial in the input size for fixed ε. Since $\left\lfloor \frac{a}{m} \right\rfloor$ can be easily calculated in polynomial time by algorithms from [14] and $\left\lfloor \frac{b}{m} \right\rfloor = \left\lfloor \frac{\sqrt{b_x^2+b_y^2}}{m} \right\rfloor$ can be calculated in polynomial time by Observation 3.3, the overall running time of the algorithm is polynomial in the input size for fixed ε. □

It remains to analyze the approximation factor of Algorithm 3.1.

Lemma 3.7. *Algorithm 3.1 computes a $(1-\varepsilon)$-approximation.*

Proof. In the following, we call grid-lines of the rhombus grid described in the beginning of this section *horizontal* if they are parallel to the x-axis and *vertical* otherwise. Consider an optimal packing of $\mathrm{opt}_\gamma(a,b) =:$ OPT unit disks into the container parallelogram together with the following assignment to the cells: If a unit disk is completely contained in a cell, assign it to that cell. If a unit disk intersects a horizontal grid-line and no vertical grid-line, assign it to the lower cell that it intersects. If a unit disk intersects only a vertical grid-line and no horizontal grid-line, assign it to the left cell that it intersects. If a unit disk intersects a horizontal and a vertical grid-line, assign it to the cell at the bottom left of the intersection of the vertical and the horizontal grid-line intersected. See Fig. 3.7 for illustration. Observe that a disk cannot intersect more than one horizontal or vertical grid line since the distance of the grid lines is $m \cdot \sin\gamma = \frac{37 s_d^2}{s_n^2 \cdot \varepsilon} \cdot \sin\gamma \geq \frac{37}{\sin^2 \alpha_0 \cdot \varepsilon} \cdot \sin\gamma \geq \frac{37}{\sin\gamma \cdot \varepsilon} > 2$.

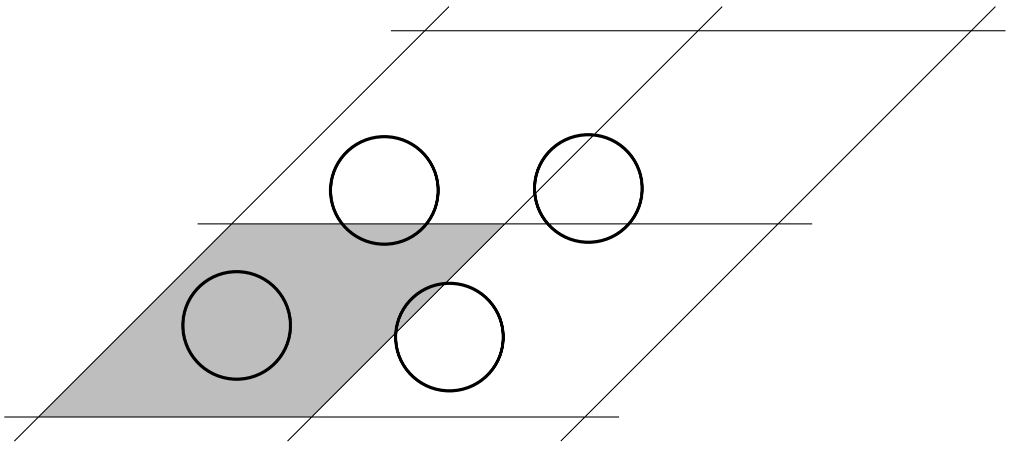


Figure 3.7: All depicted disks are assigned to the grey cell.

In order to show that Algorithm 3.1 is a PTAS, we need to show $\frac{n}{\mathrm{OPT}} \geq (1-\varepsilon)$, where n is the return value from the algorithm with

$$n = \left\lfloor \frac{a}{m} \right\rfloor \left\lfloor \frac{b}{m} \right\rfloor \cdot \mathrm{opt}_\gamma(m,m) + \left\lfloor \frac{a}{m} \right\rfloor \cdot \mathrm{opt}_\gamma(m, b \bmod m) + \\ \left\lfloor \frac{b}{m} \right\rfloor \cdot \mathrm{opt}_\gamma(a \bmod m, m) + \mathrm{opt}_\gamma(a \bmod m, b \bmod m).$$

Let $\mathrm{assign}_\gamma(v,w)$ denote the maximum number of unit disks that is assigned to a parallelogram-cell with side lengths v, w and smaller angle γ in the assignment described above. Then

$$\mathrm{OPT} \leq \left\lfloor \frac{a}{m} \right\rfloor \left\lfloor \frac{b}{m} \right\rfloor \cdot \mathrm{assign}_\gamma(m,m) + \left\lfloor \frac{a}{m} \right\rfloor \cdot \mathrm{assign}_\gamma(m, b \bmod m) + \\ \left\lfloor \frac{b}{m} \right\rfloor \cdot \mathrm{assign}_\gamma(a \bmod m, m) + \mathrm{assign}_\gamma(a \bmod m, b \bmod m).$$

So, if we can show

1. $\mathrm{opt}_\gamma(m,m) \geq (1-\varepsilon)\,\mathrm{assign}_\gamma(m,m)$ and

2. $\mathrm{opt}_\gamma(a \bmod m, m) \geq (1-\varepsilon)\,\mathrm{assign}_\gamma(a \bmod m, m)$ and
3. $\mathrm{opt}_\gamma(m, b \bmod m) \geq (1-\varepsilon)\,\mathrm{assign}_\gamma(m, b \bmod m)$ and
4. $\mathrm{opt}_\gamma(a \bmod m, b \bmod m) \geq (1-\varepsilon)\,\mathrm{assign}_\gamma(a \bmod m, b \bmod m)$,

the desired approximation factor follows immediately. We will prove these four inequalities in the following.

1. $\mathbf{opt}_\gamma\,(m,m) \geq (1-\varepsilon)\mathbf{assign}_\gamma\,(m,m)$ First, we want an upper bound on the number of unit disks that are assigned to a rhombus-shaped cell. Recall that only disks completely contained in the cell or disks close to the upper and/or right boundary of the cell are assigned to the cell. So, all disks assigned to a rhombus-cell with side length m are contained in a rhombus with side length $m + \frac{2}{\sin\gamma}$ (see Fig. 3.8).

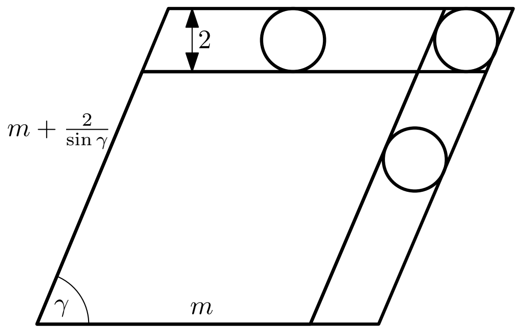


Figure 3.8: All disks assigned to the rhombus-cell with side length m are completely contained in the rhombus with side length $m + \frac{2}{\sin\gamma}$.

Therefore, we get

$$\frac{\mathrm{opt}_\gamma(m,m)}{\mathrm{assign}_\gamma(m,m)} \geq \frac{\mathrm{opt}_\gamma(m,m)}{\mathrm{opt}_\gamma\left(m + \frac{2}{\sin\gamma}, m + \frac{2}{\sin\gamma}\right)}.$$

Since $m = \frac{37 s_d^2}{s_n^2 \cdot \varepsilon} \geq \frac{37}{\sin^2\alpha_0 \cdot \varepsilon} \geq \frac{37}{\sin^2\gamma \cdot \varepsilon}$, we get by Lemma 3.5

$$\frac{\mathrm{opt}_\gamma(m,m)}{\mathrm{assign}_\gamma(m,m)} > 1 - \varepsilon.$$

2. $\mathbf{opt}_\gamma\,(a \bmod m, m) \geq (1-\varepsilon)\mathbf{assign}_\gamma\,(a \bmod m, m)$ We make a case-distinction depending on the value of $a \bmod m$. The first case follows the same idea as the previous paragraph.

Case 1: $a \bmod m \geq \frac{2}{\sin\gamma}$

All unit disks that get assinged to a parallelogram-cell with side lengths $a \bmod m, m$ are completely contained in a parallelogram with side lengths $a \bmod m, m + \frac{2}{\sin\gamma}$. Observe that we do not need to enlarge the other side of the parallelogram since this cell-shape only occurs on the boundary of the container. Thus,

$$\frac{\text{opt}_\gamma(a \bmod m, m)}{\text{assign}_\gamma(a \bmod m, m)} \geq \frac{\text{opt}_\gamma(a \bmod m, m)}{\text{opt}_\gamma\left(a \bmod m, m + \frac{2}{\sin\gamma}\right)}.$$

Since $a \bmod m \geq \frac{2}{\sin\gamma}$ and $m \geq \frac{37}{\sin^2\gamma\cdot\varepsilon}$, we get by Lemma 3.5

$$\frac{\text{opt}_\gamma(a \bmod m, m)}{\text{assign}_\gamma(a \bmod m, m)} > 1 - \varepsilon.$$

Case 2: $a \bmod m < \frac{2}{\sin\gamma}$

In this case either $a = a \bmod m < \frac{2}{\sin\gamma}$ which means that no disk can be packed into the container-parallelogram. Algorithm 3.1 returns 0 if this happens. Or $a \neq a \bmod m$ which means that no disks are assigned to this cell shape, i.e., $\text{assign}_\gamma(a \bmod m, m) = 0 = \text{opt}_\gamma(a \bmod m, m)$.

3. $\mathbf{opt}_\gamma\,(m, b \bmod m) \geq (1-\varepsilon)\mathbf{assign}_\gamma\,(m, b \bmod m)$ Since $\text{opt}_\gamma(m, b \bmod m) = \text{opt}_\gamma(b \bmod m, m)$, the calculations are analogous to the previous paragraph.

4. $\mathbf{opt}_\gamma\,(a \bmod m, b \bmod m) \geq (1-\varepsilon)\mathbf{assign}_\gamma\,(a \bmod m, b \bmod m)$ Recall that a parallelogram-cell with side lengths $a \bmod m, b \bmod m$ only occurs once at the upper right corner of the container. Therefore, only disks completely contained in that cell are assigned to the cell. This immediately gives

$$\begin{aligned}\text{opt}_\gamma(a \bmod m, b \bmod m) &\geq \text{assign}_\gamma(a \bmod m, b \bmod m)\\ &\geq (1-\varepsilon)\,\text{assign}_\gamma(a \bmod m, b \bmod m).\end{aligned}$$

This concludes the proof as described earlier. □

Lemma 3.6 and Lemma 3.7 together give one of the main results of this chapter.

Theorem 3.8. *Algorithm 3.1 is a PTAS for packing the maximum number of unit disks into a given fat container-parallelogram.*

3.2.2 Packing into Fat Triangles

We will use the same overall idea for packing unit disks into fat triangles, i.e., we divide the container with a grid into smaller cells, calculate the optimal solution for each cell type with the algorithm described in Section 2.2.1 and return the sum of the results multiplied by the number of occurrences of the cell-type. We assume that the triangle is given by its side lengths $a, b, c \in \mathbb{Q}$ with $a \geq b \geq c$ and all inner angles are at least α_0, i.e., the triangle is fat. There is one detail that does not work in the same way as for parallelograms (see Fig. 3.9): If we divide the triangle with a rhombus-grid of side length $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ into cells, we get $\Omega(\frac{c}{\varepsilon})$ many different cell shapes and we have to calculate the optimal solution for each of them. This is not polynomial in the input size and so the running time of the algorithm would not be polynomial in the input size.

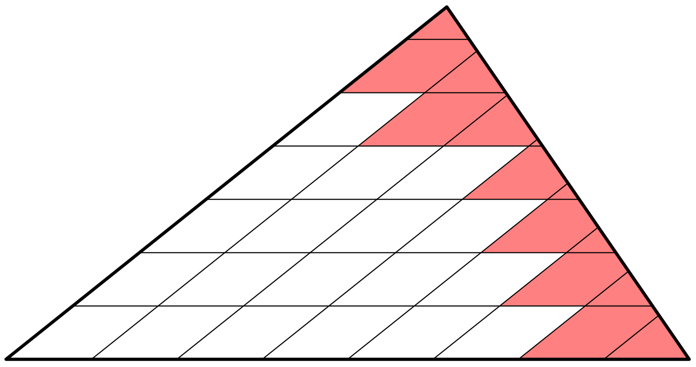


Figure 3.9: Dividing a triangle into cells with a rhombus-grid yields too many different cell-shapes.

This is why we will use another approach for the grid: The grid-cells will be parallelograms with side lengths $\frac{a}{g}, \frac{b}{g}$ for some $g \in \mathbb{N}$ with $g \in \Theta(\varepsilon \cdot b)$. We give the exact definition of g later. As one can see in Fig. 3.10, this partitioning yields only two different cell-types. As for Algorithm 3.1, we would like to use $\sin \alpha_0$ to calculate g. Again,

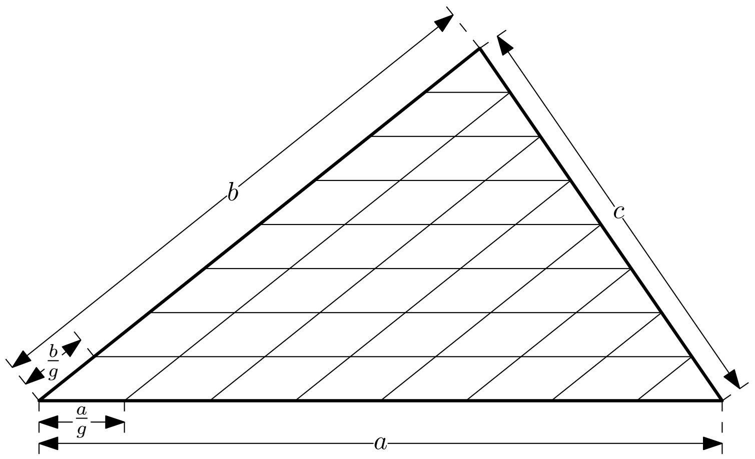


Figure 3.10: Dividing a triangle into cells with a parallelogram-grid with side lengths $\frac{a}{g}, \frac{b}{g}$ yields only two cell-types. Here $g = 8$.

since α_0 is a constant, there exist constants $s_n, s_d \in \mathbb{N}_+$ such that $\sin \alpha_0 \geq s_n/s_d > 0$. Let γ be the angle between the sides with length a and b. We only use it for notation in the algorithm and will never calculate it or use it otherwise. We state the detailed algorithm including the definition of g in the following. Again, we assume that rational numbers $x \in \mathbb{Q}$ are given as two integers x_n, x_d with $x = x_n/x_d$.

Algorithm 3.2:

```
Input: Number $a, b, c \in \mathbb{Q}_+$, parameter $0 < \varepsilon < 1 \in \mathbb{Q}_+$
Output: Nonnegative integer $n$
1 $g := \left\lfloor \frac{1}{37} \cdot \varepsilon b \cdot \frac{s_n^2}{s_d^2} \right\rfloor$;
2 if $g \leq 1$ then
3     Compute the maximum number of unit disks that can be packed into a
      triangle with side lengths $a, b, c$ with the exact algorithm (see
      Section 2.2.1) and return it;
4 Compute $n_1 = \mathrm{opt}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)$ with the exact algorithm;
5 Compute the maximum number $n_2$ of unit disks that can be packed into a
    triangle with side lengths $\frac{a}{g}, \frac{b}{g}, \frac{c}{g}$ with the exact algorithm;
6 Calculate $n = \frac{g^2 - g}{2} \cdot n_1 + g \cdot n_2$;
7 return $n$
```

As for parallelogram-shaped containers, we will first analyze the running time of the algorithm and afterwards its approximation factor.

Lemma 3.9. *Algorithm 3.2 runs in time polynomial in the input size.*

Proof. Since we use the exact algorithm described in Section 2.2.1 to calculate n_1 and n_2, we need to show that the number of unit disks that can be packed into a parallelogram with side lengths $\frac{a}{g}, \frac{b}{g}$ and smaller angle γ is bounded by a constant, and that the two different cell-shapes can be described with a polynomial-size formulas. The objects to be packed can be described easily with $f(x) = x^2 + y^2 \leq 1$. Observe that if the optimal value for the whole triangle is calculated in Line 3, we have $g \leq 1$, which implies $a \leq \frac{a}{g}$, $b \leq \frac{b}{g}$, and $c \leq \frac{c}{g}$. So, the analysis of the running time in this case is analogous to the analysis of the running time of the calculation of n_2.

The number of unit disks that can be packed in any of the cells is clearly upper bounded by

$$\begin{aligned}\frac{a \cdot b}{g^2 \cdot \pi} &= \frac{a \cdot b}{\left\lfloor \frac{1}{37} \cdot \varepsilon b \cdot \frac{s_n^2}{s_d^2} \right\rfloor^2 \cdot \pi} \\ &= \mathcal{O}\left(\frac{a \cdot b}{\varepsilon^2 \cdot b^2}\right) \\ &= \mathcal{O}\left(\frac{1}{\varepsilon^2}\right) && \left(b \geq \frac{a}{2} \text{ since } a \geq b \geq c\right).\end{aligned}$$

Similarly to the proof of Lemma 3.6, the cells can be described by first-order formulas with polynomial length description.

So, we have the same bounds as in the proof of Lemma 3.6. Again, let l be the maximum bit length of any input integer. Then, analogously to the proof of Lemma 3.6, the running time to compute the optimal solution for one cell is $(1/\varepsilon)^{\mathcal{O}(1/\varepsilon^2)}\mathcal{O}(l \cdot \mathrm{M}(l))$

which is polynomial in the input size for fixed ε. Hence, n_1 and n_2 in Algorithm 3.2 can be computed in polynomial time. Afterwards, n can easily be calculated and therefore, the complete algorithm has polynomial running time. □

Next, we analyze the approximation factor of Algorithm 3.2.

Lemma 3.10. *Algorithm 3.2 computes a* $(1-\varepsilon)$*-approximation.*

Proof. Let OPT be the maximum number of unit disks that can be packed into the input triangle with side lengths a, b, c. We like to show $\frac{n}{\text{OPT}} \geq (1-\varepsilon)$, where n is the return value from Algorithm 3.2. Consider an optimal packing of OPT disks together with the same assignment to the cells as in the proof of Lemma 3.7: If a unit disk is contained in a cell, assign it to that cell. If a unit disk intersects only the horizontal boundary of a cell, assign it to the lower cell that it intersects. If a unit disk intersects only a vertical boundary of a cell, assign it to the leftmost cell that it intersects. If a unit disk intersects a horizontal and a vertical grid-line, assign it to the cell at the lower left of the intersection of these two grid-lines. Observe that $g \leq \frac{\varepsilon b \sin^2 \gamma}{37}$ implies

$$\frac{a}{g} \geq \frac{b}{g} \geq \frac{37}{\sin^2 \gamma \varepsilon} \geq \frac{37}{\sin \gamma}, \tag{3.6}$$

i.e., a disk cannot intersect two horizontal or two vertical grid-lines. Furthermore, in the assignment described above, no unit disk not completely contained in a triangular cell gets assigned to a triangular cell. Therefore, it suffices to show

$$\frac{\text{opt}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)}{\text{assign}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)} \geq (1-\varepsilon),$$

i.e., that the number of disks packed into a parallelogram-cell by Algorithm 3.2 is at least $(1-\varepsilon)$ times the maximum number of disks that get assigned to a parallelogram-cell in an optimal packing. We use the same observations as in the proof of Lemma 3.7, namely that all disks assigned to a cell are contained in a parallelogram with side lengths $\frac{a}{g} + \frac{2}{\sin \gamma}, \frac{b}{g} + \frac{2}{\sin \gamma}$. This gives

$$\frac{\text{opt}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)}{\text{assign}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)} \geq \frac{\text{opt}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)}{\text{opt}_\gamma\left(\frac{a}{g} + \frac{2}{\sin \gamma}, \frac{b}{g} + \frac{2}{\sin \gamma}\right)}.$$

As observed in (3.6) above, $\frac{a}{g} \geq \frac{b}{g} \geq \frac{37}{\sin^2 \gamma \cdot \varepsilon}$ holds. So we get by Lemma 3.5

$$\frac{\text{opt}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)}{\text{assign}_\gamma\left(\frac{a}{g}, \frac{b}{g}\right)} > 1-\varepsilon.$$

As described earlier, this concludes the proof. □

Lemma 3.9 and Lemma 3.10 together give the following theorem.

Theorem 3.11. *Algorithm 3.2 is a PTAS for computing the maximum number of unit disks that can be packed into a fat triangle given by its side lengths* $a, b, c \in \mathbb{Q}$*.*

3.3 Generalization

We believe that the approach given in the previous section (Section 3.2) can easily be generalized for packing very many different objects than unit disks.

We assume in the following that for parallelogram-shaped and triangular containers, the edge with length a is parallel to the x-axis (See the inputs of Algorithms 3.1 and 3.2). There are two main ways to generalize the problem:

(a) The objects to be packed are part of the problem description, not part of an instance of the problem. Hence, all parameters of the objects to be packed are constants.

(b) The objects to be packed are part of an instance of the problem, i.e., they are part of the input for the algorithms to compute an approximate solution.

We will address both variants in the remainder of this section. Most of the time, the generalizations work for both variants and we will address explicitly, if they do not. In the following, we will give an outline on how to generalize the approach and afterwards follow this outline showing exemplary how to generalize the approach for packing copies of a polygon P under translation and, afterwards, under rigid motions.

Outline for Generalization Observe that there are four major points in the approach above that need to be modified:

1. We need an equivalent to Lemma 3.1, i.e., an upper and a lower bound for the maximum number of copies of P that can be packed into a parallelogram.

2. Following the proof of Lemma 3.5, we get from the modified version of Lemma 3.1 a modified version of Lemma 3.5 with adapted lower bounds on a and b, i.e., lower bounds for the side lengths of the cells such that we obtain a $(1-\varepsilon)$-approximation for each cell-type.

3. From the modified version of Lemma 3.5, we obtain updated values for m and g in Algorithm 3.1 and Algorithm 3.2, respectively. We need to ensure that these values or close enough approximations can be computed in polynomial time in order to obtain a polynomial running time for the algorithms.

4. We need to ensure that it is possible to calculate the optimal solution for one grid-cell in polynomial time. Therefore, we need to show that the object to be packed can be encoded with a polynomial length formula, and the number of objects that can be packed into one cell is bounded from above by a constant for fixed ε (see P1 and P3).

3.3.1 Packing Polygons under Translation

Next, we will generalize the approach following the outline above for packing copies of a simple polygon P under translation instead of unit disks. We assume that the polygon is given by a sequence of vertices $(v_1, \ldots, v_k) \in \mathbb{Q}^{2 \cdot k}$. We address the issues in the outline for packing polygons one after another. It should become clear that this is easily possible for many different shapes other than polygons.

1. An equivalent for Lemma 3.1 Let A be the area of P. We get a simple upper bound on the number n of copies of P that can be packed into a parallelogram with side lengths a, b and smaller angle γ by dividing their volumes:

$$n \leq \frac{ab \sin \gamma}{A}. \tag{3.7}$$

Let h be the maximum difference between the y-coordinates of any two vertices of P. Consider a smallest enclosing parallelogram for P with smaller angle γ and two edges parallel to the x-axis. Let w_γ be the side length of the edge parallel to the x-axis of this parallelogram. See Fig. 3.11 for illustration. We get a lower bound for n by packing

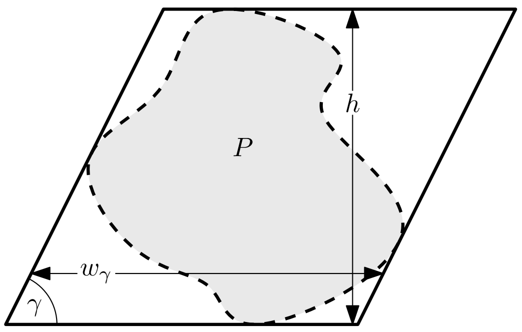


Figure 3.11: The smallest enclosing parallelogram with smaller angle γ and two sides parallel to the x-axis defines w_γ. h is the maximum difference between two points of P in their y-coordinates.

copies of P on a parallelogram-grid with side lengths $w_\gamma, h/\sin\gamma$:

$$\begin{aligned} n &\geq \left\lfloor \frac{a}{w_\gamma} \right\rfloor \left\lfloor \frac{b \sin \gamma}{h} \right\rfloor \\ &\geq \frac{ab \sin \gamma}{4 w_\gamma h}, \end{aligned} \tag{3.8}$$

for $a \geq w_\gamma$ and $b \geq h/\sin\gamma$ since $\lfloor x \rfloor \geq x/2$ for $x \geq 1$. Observe that these bounds hold independent of P being part of the problem description or an instance of the problem. Indeed, these bounds hold for all kinds of shapes, not only polygons when h and w_γ are defined accordingly.

2. A modified version of Lemma 3.5 Recall that Lemma 3.5 is used in the analysis of the approximation factors of Algorithms 3.1 and 3.2 (see the proofs of Lemmas 3.7 and 3.10) to compare the optimal number of objects that can be packed into a cell with the maximum number assigned to a cell. Recall also that the objects assigned to a cell lie completely in a cell enlarged to the top and right. Since P does not need to be symmetric as unit disks are, the values the cell needs to be enlarged to the top and right are not identical. See Fig. 3.12 for illustration. Therefore, the modified version of

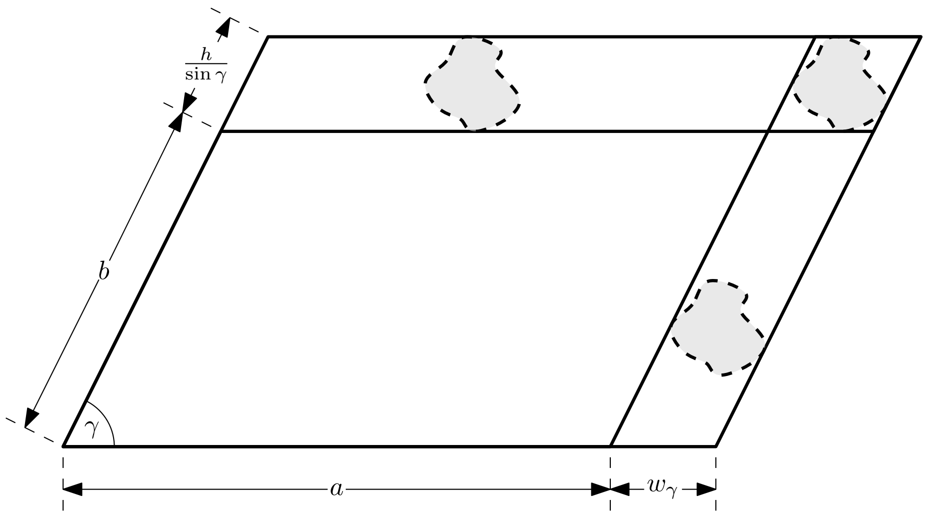


Figure 3.12: All copies of P that are assigned to a parallelogram shaped cell with side length a, b (see proofs of Lemmas 3.7 and 3.10) are contained in a parallelogram with side lengths $a + w_\gamma, b + \frac{h}{\sin\gamma}$.

Lemma 3.5 needs to contain three inequalities: Two when analyzing the factor between the number of copies of P assigned to a cell in an optimal packing and the number of copies packed into a cell by the algorithm for cells at the boundaries in parallelogram-shaped containers (Eqs. (3.9) and (3.10), see the proof of Lemma 3.7) and one for the inner cells (Eq. (3.11), see the proofs of Lemmas 3.7 and 3.10). From now on, $\text{opt}_\gamma(x, y)$ denotes the maximum number of copies of P that can be packed into a parallelogram with side lengths a, b and smaller angle γ. Hence, we want the following inequalities to hold:

$$\frac{\text{opt}_\gamma(a,b)}{\text{opt}_\gamma\left(a, b + \frac{h}{\sin\gamma}\right)} \geq 1 - \varepsilon, \tag{3.9}$$

$$\frac{\text{opt}_\gamma(a,b)}{\text{opt}_\gamma(a + w_\gamma, b)} \geq 1 - \varepsilon, \tag{3.10}$$

$$\frac{\text{opt}_\gamma(a,b)}{\text{opt}_\gamma\left(a + w_\gamma, b + \frac{h}{\sin\gamma}\right)} \geq 1 - \varepsilon. \tag{3.11}$$

Eq. (3.9) is true for $b \geq \frac{h}{\sin\gamma}$ and $a \geq \frac{8h^2 w_\gamma}{A\sin\gamma} \cdot \frac{1}{\varepsilon}$, Eq. (3.10) is true for $a \geq w_\gamma$ and $b \geq \frac{8hw_\gamma^2}{A} \cdot \frac{1}{\varepsilon}$, and Eq. (3.11) is true for $a \geq \frac{16w_\gamma^2 h}{A} \cdot \frac{1}{\varepsilon}$ and $b \geq \frac{16w_\gamma h^2}{A\sin\gamma} \cdot \frac{1}{\varepsilon}$ by similar arguments as in the proof of Lemma 3.5 using the bounds for the maximum number of copies of P that can be packed into a parallelogram obtained previously (Eqs. (3.7) and (3.8)).

3. Update values for m and g in Algorithms 3.1 and 3.2 Following the proof of Lemma 3.7, we see that the bounds for a and b in the updated version of Lemma 3.5 need to hold for m. Hence, we would like to set m to be $\frac{16w_\gamma h}{A} \cdot \max\left(w_\gamma, \frac{h}{\sin\gamma}\right) \cdot \frac{1}{\varepsilon}$. Since we may not be able to compute $\sin\gamma$ exactly, $\sin\gamma \geq \sin\alpha_0$, and α_0 is a constant, we replace $\sin\gamma$ by the lower bound $\frac{s_n}{s_d}$ for $\sin\alpha_0$ as before. h can be calculated in a straightforward manner from the vertices of P. Likewise, w_γ can be calculated from the vertices of P by solving a set of linear equalities. h and w_γ can be computed in time polynomial in the size of the bit-description of the vertices of P. Hence, it can be done in polynomial time independent of whether P is part of the input for the algorithms or not. Similarly, the area of a simple polygon can be computed using Gauss's well known area formula $\frac{1}{2}\left|\sum_{i=1}^{k}(y_i + y_{i+1})(x_i - x_{i+1})\right|$ where (x_i, y_i) are the coordinates of the v_i.

To sum up, if P is not part of the input, we first calculate w_γ. The area A and height h of P are constants. As mentioned before, there exist constants $s_n, s_d \in \mathbb{N}_+$ with $s_n/s_d \leq \sin\alpha_0 \leq \sin\gamma$. If P is part of the input we calculate h, w_γ, and A in polynomial time, s_n, s_d are constants as before. Afterwards, we set m to $\frac{16w_\gamma h}{A} \cdot \max\left(w_\gamma, \frac{hs_d}{s_n}\right) \cdot \frac{1}{\varepsilon}$ in both cases and proceed analogous to Algorithm 3.1.

One can obtain the updated value for g in Algorithm 3.2 in a similar way since we know that $\frac{b}{g}$ needs to meet the updated bounds in Lemma 3.5. Since $a \geq b$, it follows immediately that $\frac{a}{g}$ also fulfills the desired bounds.

4. Calculating the optimal solution in a cell in polynomial time P can be represented as a disjunction of conjunctions of linear inequalities of polynomial size (for example by triangulating P and each conjunction represents a triangle). It remains to show that the optimum in one cell can be computed in polynomial time. Since the cells are still parallelograms or triangles and the object to be packed, i.e. P, can be described by a polynomial length formula, the only thing we need to show is that there is a constant upper bound for the number of copies of P that can be packed into one cell for fixed ε (recall P1 to P3 given in the beginning of Section 3.2 or see Corollary 2.3 for how this bound affects the running time). We get a simple upper bound on the number of copies of P that can be packed into a cell under translation by dividing their areas. Recall that when packing into parallelograms, one cell is a parallelogram with side lengths at most m and smaller angle γ and, therefore, with area $\left(\frac{16w_\gamma h}{A} \cdot \max\left(w_\gamma, \frac{hs_d}{s_n}\right) \cdot \frac{1}{\varepsilon}\right)^2 \sin\gamma$. So, if P is part of the problem description and not part of the input, at most $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ copies of P can be packed into one cell.

To obtain the same result if P is part of the input, we need to make further assumptions on the shape of P.

Definition 3.12 (c-thickness)**.** *Let P be a polygon, A its area and d its diameter. We call P c-thick if and only if $A \geq c \cdot d^2$.*

This definition is stronger than Definition 3.4 in the following way: There exists an angle δ such that all parallelograms that are c'-thick for some constant c', are δ-fat. This is not true the other way around.

If P is part of the problem instance and c-thick we get the following when dividing the area of a cell by the area of P:

$$\begin{aligned}
&\left(\frac{16w_\gamma h}{A} \cdot \max\left(w_\gamma, \frac{hs_d}{s_n}\right) \cdot \frac{1}{\varepsilon}\right)^2 \sin\gamma \cdot \frac{1}{A} \\
\geq\ &\left(\frac{16w_\gamma h}{c \cdot d^2} \cdot \max\left(w_\gamma, \frac{hs_d}{s_n}\right) \cdot \frac{1}{\varepsilon}\right)^2 \sin\gamma \cdot \frac{1}{A} && \text{(by the def. of } c\text{-thickness)} \\
\leq\ &\left(\frac{16d \sin\gamma d}{cd^2} \cdot \frac{ds_d}{s_n}\frac{1}{\varepsilon}\right)^2 \sin\gamma \cdot \frac{1}{A} && \text{(since } h \leq d\text{)} \\
=\ &\mathcal{O}\left(\frac{1}{\varepsilon^2}\right) && \text{(since } d^2 \leq \frac{A}{c}\text{).}
\end{aligned}$$

Hence, at most $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ copies of P can be packed into one cell under translation. Similarly, we get a constant upper bound if ε is fixed for the maximum number of copies of P that can be packed in a parallelogram with side lengths $\frac{a}{g}, \frac{b}{g}$ and smaller angle γ under translation in Algorithm 3.2.

Summarizing, we get the two following theorems.

Theorem 3.13. *Let α_0 be a constant with $0 < \alpha_0 \leq \frac{\pi}{2}$.*

(a) Given an α_0-fat parallelogram by the length of an edge parallel to the x-axis and a point that describes the upper left corner, there exists a PTAS to approximate the maximum number of copies of a fixed Polygon P that can be packed into the parallelogram under translation.

(b) Let f be a constant with $0 < f < 1$. Given an α_0-fat parallelogram by the length of an edge parallel to the x-axis and an additional point that describes the upper left corner, and a f-thick polygon P by the sequence of its vertices, there exists a PTAS to approximate the maximum number of copies of P that can be packed into the parallelogram under translation.

Theorem 3.14. *Let α_0 be a constant with $0 < \alpha_0 \leq \frac{\pi}{2}$*

(a) Given an α_0-fat triangle by its side lengths, there exists a PTAS for approximating the maximum number of copies of a fixed polygon P that can be packed into the triangle under translation.

(b) *Let f be a constant with $0 < f < 1$. Given an α_0-fat triangle by its side lengths and a f-thick polygon P by the sequence of its vertices, there exists a PTAS to approximate the maximum number of copies of P that can be packed into the triangle under translation.*

3.3.2 Packing Polygons under Rigid Motions

In most parts, the generalization works analogous to the one without rotation shown before. Therefore, we will focus on the parts where it differs. Again, we will follow the outline for generalization.

1. An equivalent for Lemma 3.1 The upper bound of $n \leq \frac{ab\sin\gamma}{A}$ still holds. We obtain a lower bound by replacing w_γ and h by the following values. Consider a smallest enclosing parallelogram for P rotated by β with smaller angle γ and two sides parallel to the x-axis. Let $w_{\gamma,\beta}$ be the width of this parallelogram and h_β the maximum difference between the y-coordinates of any two vertices of P rotated by β. Let $\beta_{\min} = \operatorname{argmin}_{0\leq\beta\leq2\pi}(w_{\gamma,\beta})$. Replacing w_γ and h by $w_{\gamma,\beta_{\min}}$ and $h_{\beta_{\min}}$, respectively, in Eq. (3.8) gives:

$$\begin{aligned} n &\geq \left\lfloor \frac{a}{w_{\gamma,\beta_{\min}}} \right\rfloor \left\lfloor \frac{b\sin\gamma}{h_{\beta_{\min}}} \right\rfloor \\ &\geq \frac{ab\sin^2\gamma}{4w_{\gamma,\beta_{\min}}h_{\beta_{\min}}}, \end{aligned} \tag{3.12}$$

where the last inequality only holds for $a \geq w_{\gamma,\beta_{\min}}$ and $b \geq \frac{h_{\beta_{\min}}}{\sin\gamma}$.

2. A modified version of Lemma 3.5 Since we do not know how the copies of P are rotated, we need to enlarge the cells when analyzing the approximation factor (see the proof of Lemma 3.7) by the same amount to the top and right for the inner cells, namely by $\frac{d}{\sin\gamma}$ where d is the diameter of P. Therefore, we would like the following inequalities to hold:

$$\frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma\left(a, b+\frac{d}{\sin\gamma}\right)} \geq 1-\varepsilon, \tag{3.13}$$

$$\frac{\mathrm{opt}_\gamma(a,b)}{\mathrm{opt}_\gamma\left(a+\frac{d}{\sin\gamma}, b+\frac{d}{\sin\gamma}\right)} \geq 1-\varepsilon. \tag{3.14}$$

Eq. (3.13) holds for $a \geq w_{\gamma,\beta\min}$ and $b \geq \frac{8dw_{\gamma,\beta\min}h_{\beta_{\min}}}{A\sin^\gamma} \cdot \frac{1}{\varepsilon}$, and Eq. (3.14) is true for $a, b \geq \frac{16dw_{\gamma,\beta\min}h_{\beta_{\min}}}{A\sin^\gamma} \cdot \frac{1}{\varepsilon}$ by similar arguments as in the proof of Lemma 3.5 using Eqs. (3.7) and (3.12).

When proving the approximation factor of the algorithm analogously to the proof of Lemma 3.7, it might happen that we encounter a cell C that does not fulfill the bounds for a, b for Eq. (3.13) to hold even though copies of P are assigned to C. This only happens if C is the upper right cell and only copies of P completely contained in C get assigned to C or C equals the complete container parallelogram and we compute the exact solution with the exact algorithm (see the proof of Lemma 3.7 for the assignment and 2.2.1 for the exact algorithm).

3. Update values for m and g in Algorithms 3.1 and 3.2 We want the bounds for a, b from above to hold for m. Again, there exist constants $s_n, s_d \in \mathbb{N}$ such that $s_n/s_d \leq \sin\alpha_0 \leq \sin\gamma$ since the container parallelogram is α_0-fat. If P is part of the problem description, then there exist constants $d_n, d_d \in \mathbb{N}$ with $d_n/d_d \geq d$. So, we can set m to

$$\frac{16d_n^3 s_d^2}{d_d^3 A s_n^2} \cdot \frac{1}{\varepsilon} \geq \frac{16 d w_{\gamma,\beta_{\min}} h_{\beta\min}}{A \sin\gamma} \cdot \frac{1}{\varepsilon},$$

since $d \geq w_{\gamma,\beta_{\min}} \sin\gamma$ and $d \geq h_{\beta\min}$.

If P is part of the input and f-thick, the following holds:

$$\begin{aligned}\frac{16 d w_{\gamma,\beta_{\min}} h_{\beta\min}}{A \sin\gamma} &\leq \frac{16d}{\sin\gamma} && \text{(since } w_{\gamma,\beta_{\min}} h_{\beta_{\min}} \leq A\text{)}\\ &\leq \frac{16\sqrt{A}}{f \cdot \sin\gamma} && \text{(since } P \text{ is } f\text{-thick)}\\ &\leq \frac{16\max(w,h)}{f \cdot \sin\gamma},\end{aligned}$$

where w and h are the maximum differences between the x- and y coordinates, respectively, of any two points of the unrotated given P. Observe that w and h can easily be computed. Again, we use the existing bound s_n/s_d for $\sin\gamma$ as before and set m to $\frac{16\max(w,h)s_d}{f s_n} \cdot \frac{1}{\varepsilon}$. Afterwards, we proceed analogously to Algorithm 3.1.

Again, one can obtain updated values for g in Algorithm 3.2 for packing into a fat triangle in a similar way.

4. Calculating the optimal solution in a cell in polynomial time Since the description of a cell and P is equivalent to before, it suffices to show that there is a constant upper bound for the number of copies of P that can be packed into one cell under rigid motions (see P1 to P3 and Theorem 2.4). If P is part of the problem description and not part of the input, one cell has area at most $\left(\frac{16d_n^3 s_d^2}{d_d^3 A s_n^2} \cdot \frac{1}{\varepsilon}\right)^2 \sin\gamma$. Since all parameters of P are constants, $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ copies of P can be packed into one cell.

If P is part of the input and f-thick, dividing the area of a cell by the area of P gives at most

$$
\begin{aligned}
\left(\frac{16\max(w,h)s_d}{fs_n}\cdot\frac{1}{\varepsilon}\right)^2 \sin\gamma\cdot\frac{1}{A} &\le \left(\frac{16ds_d}{fs_n}\cdot\frac{1}{\varepsilon}\right)^2 \sin\gamma\cdot\frac{1}{A} && \text{(since } \max(w,h)\le d\text{)}\\
&\le \left(\frac{16s_d}{fs_n}\cdot\frac{1}{\varepsilon}\right)^2 \sin\gamma\cdot\frac{1}{f} && \text{(since } d^2\le\frac{A}{f}\text{ by def.)}\\
&= \mathcal{O}\left(\frac{1}{\varepsilon^2}\right).
\end{aligned}
$$

Hence, at most $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ copies of P can be packed into one cell under rigid motions. Similarly, one can get a constant upper bound for fixed ε on the maximum number of copies of P that can be packed into one cell in Algorithm 3.2 when packing into a fat triangle.

Summarizing, Theorems 3.13 and 3.14 also hold for packing under rigid motions.

Theorem 3.15. *Let α_0 be a constant with $0 < \alpha_0 \le \frac{\pi}{2}$.*

(a) Given an α_0-fat parallelogram by the length of an edge parallel to the x-axis and a point that describes the upper left corner, there exists a PTAS to approximate the maximum number of copies of a fixed Polygon P that can be packed into the parallelogram under rigid motions.

(b) Let f be a constant with $0 < f < 1$. Given an α_0-fat parallelogram by the length of an edge parallel to the x-axis and an additional point that describes the upper left corner, and a f-thick polygon P by a sequence of its vertices, there exists a PTAS to approximate the maximum number of copies of P that can be packed into the parallelogram under rigid motions.

Theorem 3.16. *Let α_0 be a constant with $0 < \alpha_0 \le \frac{\pi}{2}$*

(a) Given an α_0-fat triangle by its side lengths, there exists a PTAS for approximating the maximum number of copies of a fixed polygon P that can be packed into the triangle under rigid motions.

(b) Let f be a constant with $0 < f < 1$. Given an α_0-fat triangle by its side lengths and a f-thick polygon P by a sequence of its vertices, there exists a PTAS to approximate the maximum number of copies of P that can be packed into the triangle under rigid motions.

Chapter 4

Packing Unit Disks into General Triangles

In the previous chapter, we gave a PTAS to pack unit disks into triangles where all inner angles were lower bounded by a constant. In this chapter, we will turn to general triangles, i.e., the inner angles of the triangle can be arbitrarily small.

4.1 Preliminaries

In the following, we discuss some properties of packings of unit disks that we will use afterwards to give a constant factor approximation algorithm.

Consider a packing of unit disks in the plane, i.e., possibly infinitely many unit disks placed nonoverlappingly in the plane, together with a given container such that no disk can be added to the packing inside the container. We call such a packing *maximal*. The following lemma relates a maximal packing to an optimal packing.

Lemma 4.1. *For a maximal packing of s unit disks together with a given container $\triangle$ it holds that $s \geq \frac{1}{5}\,\mathrm{OPT}$, where OPT is the size of any optimal packing into $\triangle$.*

Proof. Consider a maximal packing P of t unit disks together with $\triangle$ and an optimal packing of OPT unit disks into $\triangle$. Every disk in P intersects at most 5 unit disks in the optimal packing since a unit disk can touch 6 but intersect only 5 pairwise disjoint disks (see Fig. 4.1). Every disk d in the optimal packing is intersected by at least one and at most 5 disks in P, since otherwise d could be added to P generating a larger valid packing contradicting that P is maximal. Thus, $5t \geq \mathrm{OPT}$. □

Observe that Lemma 4.1 implies that every maximal packing completely contained in the container is a $\frac{1}{5}$-approximation of the optimal packing. So, if we can find a packing

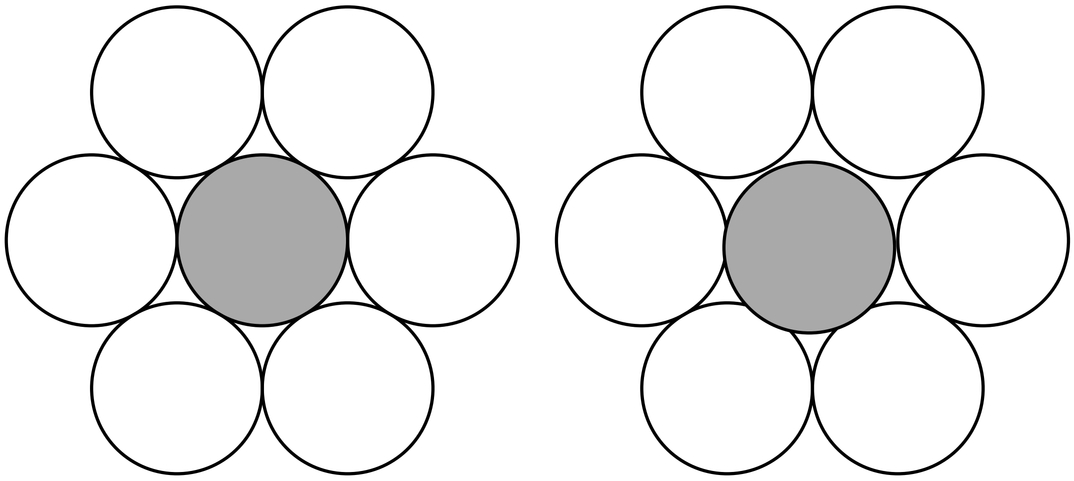


Figure 4.1: A disk can touch 6 pairwise disjoint disks of the same size simultaneously, i.e., the 2D-kissing number is 6. It is not possible that the center disk intersects all 6 surrounding disks since neighboring disks around the center disk touch.

that has size within a constant factor of the size of a maximal packing, we have a constant factor approximation. The idea of our algorithm is to arrange the disks in layers of height 2 inside the triangle as can be seen in Fig. 4.2i. We assume in the following that the container triangle is given by its side lengths a, b, c. Furthermore, we assume that the side of the triangle with length a is horizontal and the layers are parallel to that side. Then, the just described packing has size

$$\sum_{i=1}^{k} \left\lfloor \frac{s_i}{2} \right\rfloor \tag{4.1}$$

where s_i is the width of the triangle at height $2i$, i.e., the width of the ith layer and $s_0 = a$, and $k \in \mathbb{N}$ is the largest integer such that $s_k \geq 2$. Let

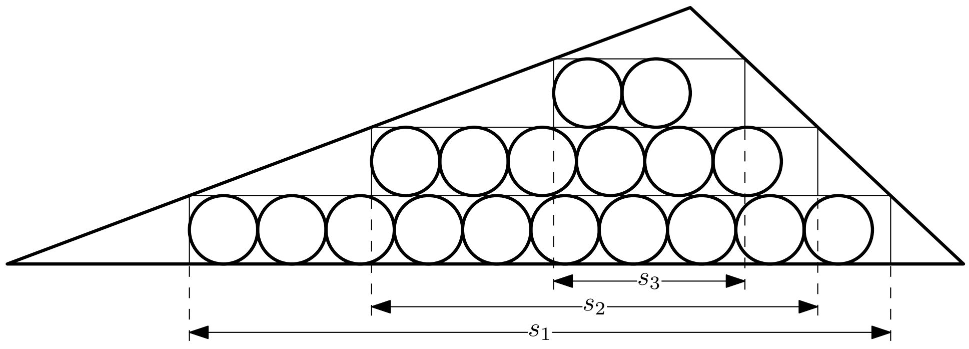
$$h = \sqrt{b^2 - \frac{b^2 - c^2 + a^2}{4a^2}} \tag{4.2}$$

be the height of the container triangle based on the side with length a. Then, we get for the widths of the layers

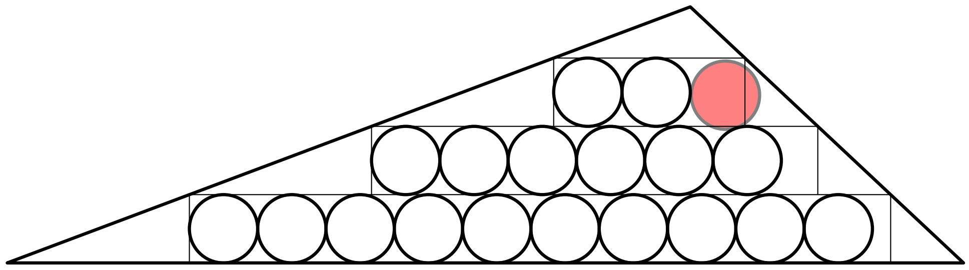
$$s_i = a - 2i \cdot \frac{a}{h}. \tag{4.3}$$

So, k is supposed to be the maximum integer such that

$$a - \frac{2ka}{h} \geq 2 \qquad\qquad ,\text{i.e.,}$$
$$k = \left\lfloor \frac{(a-2)h}{2a} \right\rfloor. \tag{4.4}$$



(i) Disks packed in layers of height 2 into a general triangle.



(ii) The packing in Fig. 4.2i is not maximal since the red disk can be added to the packing.

Figure 4.2

Observe that the described packing is not necessarily maximal as depicted in Fig. 4.2ii. We will show that this packing is within a constant factor of a maximal packing. The maximal packing discussed in the following lemma is the one we will compare our packing with.

Lemma 4.2. *There is a maximal packing for the triangle with side lengths a, b, c of size $\lceil \frac{s_1}{2} \rceil + \sum_{i=1}^{k} \lceil \frac{s_i}{2} \rceil$ with s_i and k defined as in Eqs. (4.3) and (4.4) respectively.*

Proof. Consider a placement of unit disks in horizontal layers of height 2 parallel to the side with length a as depicted in Fig. 4.3. The first layer is constructed from the first

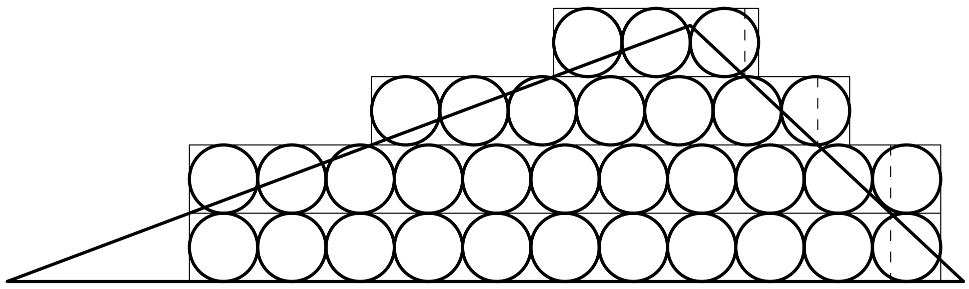


Figure 4.3: Disks placed in layers of height 2 onto a general triangle.

layer in Fig. 4.2i by enlarging its width to an integral multiple of 2, i.e. its width is

$\lceil \frac{s_1}{2} \rceil \cdot 2$. The ith layer for $2 \leq i \leq k+1$ is constructed from the $(i-1)$th layer in Fig. 4.2i by enlarging its width to an integral multiple of 2 and placing it on top of the $(i-1)$th layer such that its left boundary is aligned with the intersection of the top boundary of the layer below with the left triangle boundary. By construction, its width is $\lceil \frac{s_{i-1}}{2} \rceil \cdot 2$.

Observe that the triangle is almost entirely covered by rectangles forming the layers. It is not possible to pack a disk using the areas not covered at the bottom left and bottom right due to the definition of s_1. See Fig. 4.4 for illustration. It is also not possible to

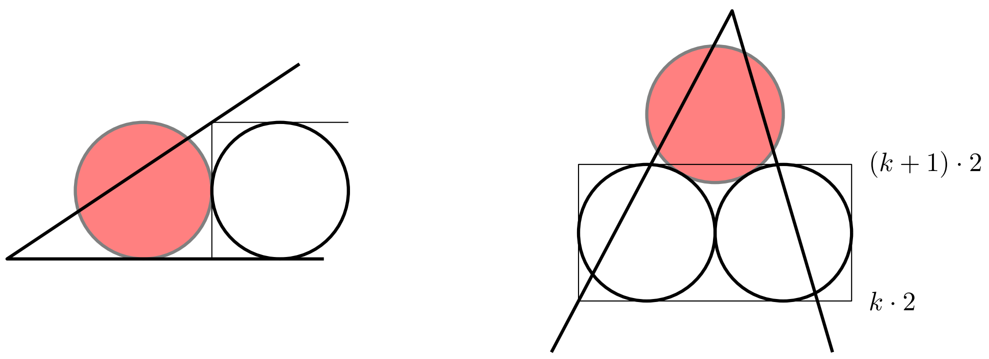


Figure 4.4: Extra disks cannot be packed in the corners of the triangle.

pack another disk at the top of the triangle. This is because even if there is an uncovered area, the width of the triangle at height $(k+1) \cdot 2$ is less than 2 due to how we defined k. Obviously, it is not possible to pack further disks inside a layer. The gaps between disks of two consecutive layers are also not large enough to pack more disks. Therefore, this placement is maximal and has size $\lceil \frac{s_1}{2} \rceil + \sum_{i=1}^{k} \lceil \frac{s_i}{2} \rceil$, concluding the proof. $\square$

We will use this maximal packing later in the analysis of the approximation factor.

4.2 Approximation Algorithm

Recall that the idea for the approximation algorithm is to compute the number of unit disks that can be packed inside the container triangle in layers of height 2, and the idea for the proof of the approximation factor is showing that this number is within the size of the maximal packing discussed in Lemma 4.2. The exact value for the packing in layers of height 2 as given in Eq. (4.1) is $\sum_{i=1}^{k} \lfloor \frac{s_i}{2} \rfloor$. Observe that we cannot compute this value simply by a loop over k since k might be exponentially large in the input size.

Instead, we want to use the following approximation:

$$\begin{aligned}\sum_{i=1}^{k}\left\lfloor\frac{s_i}{2}\right\rfloor > \sum_{i=1}^{k}\left(\frac{s_i}{2}-1\right) &= \sum_{i=1}^{k}\left(\frac{a}{2}-\frac{ia}{h}\right)-k\\ &= \frac{ka}{2}-\frac{ak(k+1)}{2h}-k\\ &> \left\lfloor\frac{ka}{2}-\frac{ak(k+1)}{2h}-k\right\rfloor.\end{aligned}$$

Recall that a is given as a fraction of two integers, k is an integer (see (4.4)) that can be computed in polynomial time by Observation 3.3, and h is given by the formula in (4.2). It is easy to see that this formula can be rewritten in the form

$$\begin{aligned}\left\lfloor\frac{v}{\sqrt{w}}\right\rfloor &= \left\lfloor\frac{v\sqrt{w}}{w}\right\rfloor\\ &= \left\lfloor\frac{\sqrt{v^2w}}{w}\right\rfloor,\end{aligned}$$

where v and w are polynomials over input-integers and k. Hence, this value can be calculated in polynomial time by Observation 3.3.

It might happen due to these approximations and the structure of the packing that the formula given above evaluates to zero even though at least one unit disk can be packed into the container. This would mean that the solution of the algorithm is infinitely worse than the optimal solution. To prevent this, we check also the following two lower bounds. First, we know that k is chosen in such a way, that $s_i \geq 2$ for $1 \leq i \leq k$ and therefore at least k unit disks can be packed into the triangle. Second, if k is zero at most one unit disk can be packed into the triangle (see case 1 in the proof of Lemma 4.4). We check if the inradius of the triangle is at least one and therefore one unit disk can be packed.

We summarize the ideas previously given in the following algorithm.

Algorithm 4.1:

```
   Input: Numbers $a, b, c \in \mathbb{Q}_+$
   Output: Nonnegative integer $n$
1  Calculate $k = \left\lfloor\frac{(a-2)h}{2a}\right\rfloor$;
2  Calculate $n = \max\left(k, \left\lfloor\frac{ka}{2}-\frac{ak(k+1)}{2h}-k\right\rfloor\right)$, where $h := \sqrt{b^2-\frac{b^2-c^2+a^2}{4a^2}}$;
3  if $n = 0$ then
4      Calculate the inradius $r$ of a triangle with side lengths $a, b, c$.;
5      if $r \geq 1$ then
6          return 1;
7      else
8          return 0;
9      end
10 end
11 return $n$
```

We will first analyze the running time of the algorithm and afterwards its approximation factor.

Lemma 4.3. *Algorithm 4.1 runs in time polynomial in the number of input bits.*

Proof. Recall that k can be computed in polynomial time by Observation 3.3. We do not calculate h but plug it into the formula for n. Recall, that we can rewrite $\left\lfloor \frac{ka}{2} - \frac{ak(k+1)}{2h} - k \right\rfloor$ such that we can use again Observation 3.3 to calculate it in polynomial time. Hence, n can be calculated in polynomial time. The inradius of a triangle with side lengths a, b, c is $\sqrt{\frac{(b+c-a)(a+c-b)(a+b-c)}{4(a+b+c)}}$. Since we only need to know if the inradius is at least one, it suffices to check if $\frac{(b+c-a)(a+c-b)(a+b-c)}{4(a+b+c)}$ is at least one and this is possible in polynomial time. Summarizing, Algorithm 4.1 runs in time polynomial in the input size. □

The following lemma relates the value n returned by the algorithm with the optimum and will yield the approximation factor and the asymptotic approximation factor.

Lemma 4.4. *Let* OPT *be the maximum number of unit disks that can be packed into a triangle with side lengths $a \geq b \geq c \in \mathbb{Q}^+$. Then, for the return value n of Algorithm 4.1 the following holds: $4n + 5 \geq \frac{1}{15}\,\text{OPT}$.*

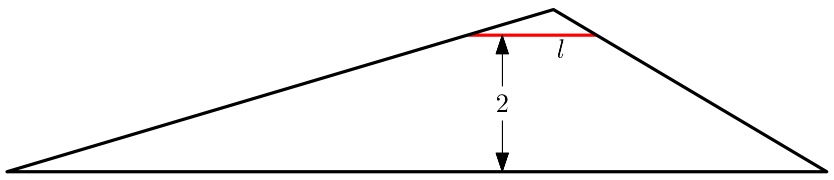
Proof. **Case 1:** $k = 0$

We will show that the algorithm returns the optimal solution in this case. First, we show that if $k = 0$ at most one unit disk can be packed into the triangle. Recall that we assume the longest side of the container triangle with length a to be parallel to the x-axis. Let l be the longest possible line segment parallel to the x-axis inside the container triangle at height 2 (the red line segment in Fig. 4.5i). Observe that the length of l is s_1 by definition (see Eq. (4.3)). From the definition of k in Eq. (4.4), we get

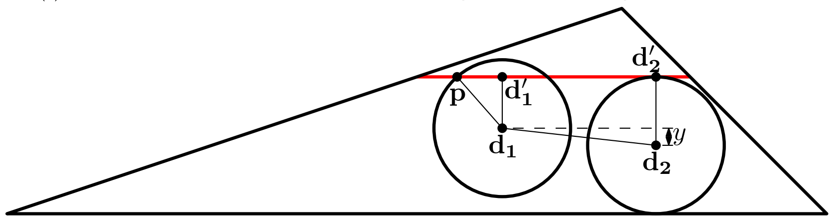
$$s_1 < 2, \tag{4.5}$$

i.e., the length of l is less than 2. We will prove by contradiction that at most one disk can be packed in such a triangle. We assume there exists a packing of two unit disks D_1 and D_2 with centers d_1 and d_2 respectively. Without loss of generality, we assume that the y-coordinate of d_2 is at most as large as the y-coordinate of d_1, i.e., D_2 lies lower than D_1, and the x-coordinate of d_1 is less than the x-coordinate of d_2, i.e., D_1 lies to the left of D_2. We observe that no disk can lie with its center above l since the width of the triangle above that line is less than 2. Obviously, we can move D_2 downwards until it touches the lower boundary of the container triangle without intersecting D_1 or any triangle-boundary. We show in the following that the width of the triangle at height 2, i.e., s_1, has to be at least 2, so, l has length at least 2, contradicting the assumption that $k = 0$ as described before.

Let $\mathbf{p}$ be the leftmost intersection of D_1 with l, and $\mathbf{d_1'}, \mathbf{d_2'}$ the orthogonal projections of $\mathbf{d_1}, \mathbf{d_2}$ onto l respectively. See Fig. 4.5ii for illustration. Let $\mathrm{d}(q, r)$ be the euclidean distance between any two points $\mathbf{q}$ and $\mathbf{r}$. Since D_1 and D_2 lie inside the triangle,



(i) $k = 0$ implies that the red line segment has length less than 2.



(ii) Setting of the proof. The figure is not up to scale since we show that this is not possible.

Figure 4.5: If $k = 0$ at most one unit disk can be packed into the container triangle.

$\mathrm{d}(\mathbf{p}, \mathbf{d_2'}) \leq s_1$. We will show in the following that $\mathrm{d}(\mathbf{p}, \mathbf{d_2'})$ has to be at least 2 in order to fit D_1 and D_2 inside the triangle.

Let y be the difference of $\mathbf{d_1}$ and $\mathbf{d_2}$ in their y-coordinates. The following holds by the Pythagorean Theorem:

$$\mathrm{d}(\mathbf{d_1'}, \mathbf{d_2'})^2 + y^2 \geq 2^2,$$

$$\text{so} \qquad \mathrm{d}(\mathbf{d_1'}, \mathbf{d_2'}) \geq \sqrt{4 - y^2}. \tag{4.6}$$

The following holds again by the Pythagorean Theorem:

$$\mathrm{d}(\mathbf{p}, \mathbf{d_1'})^2 + (1 - y)^2 = 1^2,$$

$$\text{so} \qquad \mathrm{d}(\mathbf{p}, \mathbf{d_1'}) = \sqrt{1 - (1 - y)^2} = \sqrt{2y - y^2}. \tag{4.7}$$

By Eqs. (4.6) and (4.7) we get

$$\begin{aligned}
s_1 &\geq \mathrm{d}(\mathbf{p}, \mathbf{d_2'}) = \mathrm{d}(\mathbf{p}, \mathbf{d_1'}) + \mathrm{d}(\mathbf{d_1'}, \mathbf{d_2'}) \\
&\geq \sqrt{2y - y^2} + \sqrt{4 - y^2} \\
&\geq \sqrt{2y - y^2 + 4 - y^2} \\
&= \sqrt{4 + 2(y - y^2)} \\
&\geq \sqrt{4} = 2 && \text{(since } y \in [0, 1]).
\end{aligned}$$

This is a contradiction to Eq. (4.5). We just showed that $k = 0$ implies that at most one unit disk can be packed. Since $k = 0$ implies $n = 0$ in Line 2 in Algorithm 4.1, the if-condition in Line 3 evaluates to true and the algorithm computes the inradius of the triangle. If the radius is at least one, one unit disk can be packed into the triangle and the algorithm returns 1. If the inradius is less than one, no unit disk can be packed into the triangle and the algorithm returns 0. So, in this case Algorithm 4.1 returns the optimal solution, i.e. $n = \mathrm{OPT}$.

Case 2: $k \geq 1$

Observe that

$$\begin{aligned}
n &\geq \left\lfloor \frac{ka}{2} - \frac{ak(k+1)}{2h} - k \right\rfloor && \text{(see Algorithm 4.1), so} \\
n &\geq \frac{ka}{2} - \frac{ak(k+1)}{2h} - k - 1 \\
&= \sum_{i=1}^{k} \left(\frac{a}{2} - \frac{ia}{h} - 1 \right) - 1 \\
&= \frac{a}{2} - \frac{a}{h} - 1 + \sum_{i=2}^{k} \left(\frac{a}{2} - \frac{ia}{h} - 1 \right) - 1 \\
&\geq \frac{a}{2} - \frac{a}{h} - 2 + \sum_{i=2}^{k} \left(\frac{a}{2} - \frac{(a-2)ha}{2ah} - 1 \right) && \left(\text{since } i \leq k \leq \frac{(a-2)h}{2a} \text{ by (4.4)}\right) \\
&= \frac{s_1}{2} - 2 && \text{(by Eq. (4.3)).} \qquad (4.8)
\end{aligned}$$

In the following, we consider the maximal packing of s disks as discussed in Lemma 4.2:

$$\begin{aligned}
s &= \left\lceil \frac{s_1}{2} \right\rceil + \sum_{i=1}^{k} \left\lceil \frac{s_i}{2} \right\rceil \\
&\leq \frac{s_1}{2} + 1 + k + \sum_{i=1}^{k} \frac{s_i}{2} \\
&= \frac{s_1}{2} + 1 + k + \frac{ka}{2} - \frac{ak(k+1)}{2h} && \text{(by the def. of } s_i \text{ (4.3))} \\
&= \frac{s_1}{2} + 1 + 2k + \frac{ka}{2} - \frac{ak(k+1)}{2h} - k \\
&\leq \frac{s_1}{2} + 2 + 2n + \frac{ka}{2} - \frac{ak(k+1)}{2h} - k - 1 && (k \leq n, \text{ see Algorithm 4.1}) \\
&\leq \frac{s_1}{2} + 2 + 2n + n && \text{(see Algorithm 4.1)} \\
&\leq n + 2 + 2 + 3n = 4n + 4 && \text{(see (4.8)).}
\end{aligned}$$

From Lemma 4.1 we know that for a maximal packing of s unit disks it holds that $s \geq \frac{1}{5}\,\mathrm{OPT}$. Together, we get

$$4n + 4 \geq s \geq \frac{1}{5}\,\mathrm{OPT}\,.$$

□

The following theorem follows from Lemma 4.3 and Lemma 4.4.

Theorem 4.5. *Algorithm 4.1 computes in polynomial time a* 40*-approximation for the maximal number of unit disks that can be packed into a given triangle with side lengths* a, b, c. *It has an asymptotic approximation factor of* 20.

Proof. Since the algorithm returns the optimal solution if no unit disk can be packed and otherwise for the return value n holds $\frac{1}{5}\,\mathrm{OPT} \leq 4n + 4 \leq 8n$ by Lemma 4.4, it computes a 40-approximation. Similarly, from $\frac{1}{5}\,\mathrm{OPT} - 4 \leq 4n$ follows that the algorithm has an asymptotic approximation factor of 20. It runs in polynomial time by Lemma 4.3. □

4.3 Remarks

The approximation factors obtained above are forbiddingly high to use the algorithms in practice. But they show that the problem of finding the maximum number of unit-disks that can be packed into a triangle can generally be approximated within a constant factor.

The approach of packing unit-disks in layers of height two does not yield a constant-factor approximation for parallelograms: Assume that the parallelogram is given by its base-edge length a and a point (b_x, b_y) as in the algorithm for fat parallelograms. Consider a parallelogram with smaller angle $\gamma > 0$, where the two non-horizontal edges have distance two and the height of the parallelogram is $b_y = 2$. The number of unit-disks that can be packed into this parallelogram in layers of height two is two, as can bee seen in Fig. 4.6, whereas the maximum number of unit-disks that can be packed tends to infinity when γ tends to zero. Observe that we cannot just compute a similar parallelogram where the other edges are horizontal since the distance of (b_x, b_y) to the origin involves a square root.

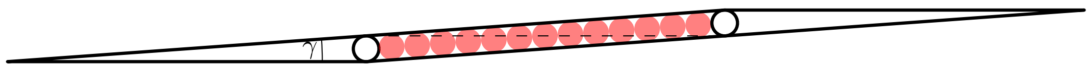


Figure 4.6: When γ tends to zero while the distance between the two non-horizontal edges stays two, the number of unit-disks that can be packed into this parallelogram grows to infinity while the number of unit-disks packed in horizontal layers of height two stays two.

Chapter 5
PTAS for Packing Unit Disks

In this chapter, we will give another PTAS for packing unit disks. We start with packing unit disks into circles and show afterwards how this approach can be used as a framework to pack into other container shapes such as polygons and general convex shapes or to the three dimensional equivalent problem of packing balls into spheres. Differently to Chapter 3, the container shape is for most variants studied here not part of the problem input but part of the problem description, i.e., only a factor $r \geq 1$ is given as input for fixed ε. The goal is to approximate the maximum number of unit disks that can be packed into the container shape scaled by r. As before, the input is very concise since only r is given by a fraction where nominator and denominator are given in binary. So, the complexity seems considerably high. However, also for these variants of the problem slightly different from the problems studied in the previous chapters, not much is known about their complexity, for example if they are NP-hard or not.

We will give polynomial time approximation schemes (PTAS) and therefore a first result on the complexity and approximability of this kind of problem.

The approximation algorithms are based on the simple idea of just multiplying the volume of the given container with the density of the optimal packing of the entire plane or space. For large containers, dividing this value by the volume of a unit disk or ball is a reasonably good approximation (where a few correction terms are necessary). For small containers, we could just run the exact algorithm given in Section 2.2.1. Observe that we will again not actually return a packing in the sense that we give centers for each disk packed. We will rather give the number of disks that can be packed since otherwise the output might have exponential size in the mentioned concise input. Later on, we will show how to extend these ideas to other container shapes. We believe that the ideas given here can also be used as a framework to derive PTAS for other container shapes than the ones studied here.

Although the idea of our algorithms just explained is quite simple, the actual challenge lies in working out the details and proving their correctness, i.e., that they are indeed PTASs for the problems stated. It should be noted that, especially since large first-order formulas of the reals need to be solved, the algorithms are not really applicable in practice. The major contribution of this result is rather the theoretical result that PTASs exist for this kind of packing problems with concise input.

5.1 Packing Unit Disks into a Circle

Let us first consider the problem of finding the maximum number $n_{\max}$ of unit disks to be packed into a container circle of a given radius r. In the following, we will describe the approximation algorithm in more detail. The input consists of a radius $r > 3$ given as a fraction $\frac{p}{q}$ and some $\varepsilon > 0$ given as fraction $\frac{u}{v}$ with $p, q, u, v \in \mathbb{N}$. Recall that the idea is to calculate the exact value for small r and returning an approximate value for the rest based on the area of the container multiplied by the density of the hexagonal packing.

Algorithm 5.1 states the algorithm in more detail. We will explain possible implementations of Line 2 and Lines 5 to 8 later when analyzing the running time.

Algorithm 5.1:

```
Input: Number $r > 3$ given as $\frac{p}{q}$, parameter $\varepsilon > 0$ given as $\frac{u}{v}$, with $p, q, u, v$
       positive integers
Output: Nonnegative integer $n_{\text{approx}}$
1  if $r < 6 \cdot \frac{1}{\varepsilon}$ then
2  |   Compute $n_{\text{approx}}$ with the exact algorithm;
3  end
4  else
5  |   $k = 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor) + 3$ ;
6  |   compute some $a$ with $\pi \geq a \geq \pi - 2^{-k}$;
7  |   compute some $b$ with $\sqrt{12} \leq b \leq \sqrt{12} + 2^{-k}$;
8  |   $n_{\text{approx}} = \left\lceil \frac{a(r-3)^2}{b} \right\rceil$;
9  end
10 return $n_{\text{approx}}$;
```

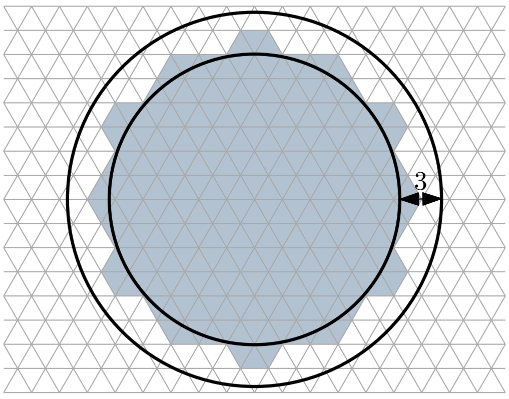
Observe that it is not possible to give the packing explicitly since n_{approx} is exponential in the input size (the actual value, not its representation) and therefore giving the position of every disk in the packing would take exponential time. Nevertheless, the packing is implicitly given by the points of a hexagonal grid that lie inside the container and have distance at least 1 from the boundary or by the semialgebraic set returned by the exact algorithm (see [12] for details).

What remains to be done is to fill in the details of each step, i.e., how to compute k, a, and b, to show that the algorithm is correct, and to analyze its runtime, i.e. show that it really is a PTAS.

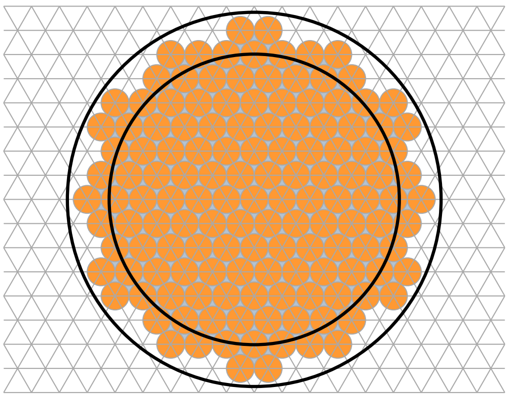
First, we show a lower bound for $n_{\max}$ and then use it to show n_{approx} unit disks can indeed be packed into the container. Afterwards, we will show that n_{approx} lies within a factor $(1-\varepsilon)$ of $n_{\max}$ by using Theorem 2.1. Lastly, we will analyze the running time. The following Lemma gives a lower bound for $n_{\max}$.

Lemma 5.1. $n_{\max} \geq \left\lceil \frac{\pi(r-3)^2}{\sqrt{12}} \right\rceil$.

Proof. Consider the dense hexagonal packing of the plane with unit disks, i.e., the centers of the disks form a triangular grid with side length 2 (see Fig. 2.1). The vertices of every triangle of the triangular grid at least partially contained in the circle with radius $r-3$ have distance at most two from the boundary of the circle with radius $r-3$. Therefore, they have distance at least 1 from the boundary of the container circle, i.e., unit disks can be packed with their centers on the vertices of these triangles (see Figs. 5.1i and 5.1ii).



(i) The vertices of triangles intersecting the circle of radius $r-3$ have distance at least one from the boundary of the container circle with radius r.


(ii) Disks placed on the vertices of triangles intersecting the circle with radius $r-3$ are completely contained in the container circle with radius r and, therefore, form a valid packing.

Any triangle contains in total at most half a disk of the packing. Hence, if we divide the area of the circle with radius $r-3$ by the area of a triangle, we get a lower bound for twice the number of disks that can be packed into the container. The area of an equilateral triangle with edge length 2 is $\sqrt{3}$. The Lemma follows immediately. □

Now, we use this lower bound for $n_{\max}$ to show that n_{approx} unit disks can surely be packed.

Lemma 5.2. $n_{approx} \leq n_{\max}$.

Proof. If n_{approx} is computed with the exact algorithm, we know that $n_{\text{approx}} = n_{\max}$. If n_{approx} is not computed with the exact algorithm, we know that

$$n_{\text{approx}} \leq \left\lceil \frac{\pi}{\sqrt{12}}(r-3)^2 \right\rceil,$$

since $a \leq \pi$ and $b \geq \sqrt{12}$. So following Lemma 5.1 $n_{\text{approx}} \leq n_{\max}$. □

Next, we prove the approximation factor of the algorithm.

Lemma 5.3. $n_{approx} \geq (1-\varepsilon) \cdot n_{\max}$.

Proof. To compute the approximation factor of the algorithm, we first determine bounds for a and b.

Since $k = 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor) + 3$, we get

$$a \geq \pi - 2^{-k} = \pi - \frac{1}{2^{2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)+3}} \geq \pi - \frac{1}{2^{2(\log p - \log q)} \cdot 8} = \pi - \frac{1}{8r^2},$$

similarly, we get

$$b \leq \sqrt{12} + 2^{-k} \leq \sqrt{12} + \frac{1}{8r^2}.$$

Now, we are going to use these bounds to calculate a lower bound on n_{approx} in case it is not calculated with the exact algorithm. Then,

$$n_{\text{approx}} = \left\lceil \frac{a(r-3)^2}{b} \right\rceil,$$

so, using the bounds for a and b:

$$\begin{aligned}
n_{\text{approx}} &\geq \frac{\pi - \frac{1}{8r^2}}{\sqrt{12} + \frac{1}{8r^2}}(r-3)^2 \\
&= \left(\frac{\pi - \frac{1}{8r^2}}{\sqrt{12} + \frac{1}{8r^2}} + \frac{\pi}{\sqrt{12}} - \frac{\pi}{\sqrt{12}} \right)(r-3)^2 \\
&= \frac{\pi}{\sqrt{12}}(r-3)^2 + \frac{\sqrt{12}\left(\pi - \frac{1}{8r^2}\right) - \pi\left(\sqrt{12} + \frac{1}{8r^2}\right)}{\sqrt{12}\left(\sqrt{12} + \frac{1}{8r^2}\right)}(r-3)^2 \\
&= \frac{\pi}{\sqrt{12}}(r-3)^2 - \frac{\sqrt{12} + \pi}{8r^2\left(12 + \frac{\sqrt{12}}{8r^2}\right)}(r-3)^2 \\
&> \frac{\pi}{\sqrt{12}}(r-3)^2 - \left(\frac{\sqrt{12} + \pi}{96 + \frac{\sqrt{12}}{r^2}} + \frac{9\left(\sqrt{12} + \pi\right)}{96r^2 + \sqrt{12}} \right) \\
&\geq \frac{\pi}{\sqrt{12}}(r-3)^2 - \left(\frac{\sqrt{12} + \pi}{96} + \frac{9\left(\sqrt{12} + \pi\right)}{96 + \sqrt{12}} \right) \qquad \text{(since } r \geq 1\text{)} \\
&\approx \frac{\pi}{\sqrt{12}}(r-3)^2 - 0.67 \\
&> \frac{\pi}{\sqrt{12}}(r-3)^2 - 1. \qquad (5.1)
\end{aligned}$$

The following upper bound for $n_{\max}$ follows directly from Theorem 2.1:

$$n_{\max} \leq \frac{\pi}{\sqrt{12}} r^2. \tag{5.2}$$

With the lower bound on n_{approx} given in (5.1) and the upper bound on $n_{\max}$ given in (5.2), we can compute the approximation factor of the algorithm stated above:

$$\begin{aligned}\frac{n_{\text{approx}}}{n_{\max}} &\geq \frac{\frac{\pi}{\sqrt{12}}(r-3)^2 - 1}{\frac{\pi}{\sqrt{12}} r^2} \\ &= \frac{\frac{\pi}{\sqrt{12}} r^2 - \frac{\pi}{\sqrt{12}} 6r + \frac{\pi}{\sqrt{12}} \cdot 9 - 1}{\frac{\pi}{\sqrt{12}} r^2} \\ &> 1 - \frac{\frac{\pi}{\sqrt{12}} 6r}{\frac{\pi}{\sqrt{12}} r^2} \\ &= 1 - \frac{6}{r}.\end{aligned}$$

Since we calculate the exact solution for $r < \frac{6}{\varepsilon}$, we can assume that $r \geq \frac{6}{\varepsilon}$ and therefore obtain the approximation factor of $1 - \varepsilon$, i.e.

$$\frac{n_{\text{approx}}}{n_{\max}} \geq 1 - \varepsilon.$$

□

The last component for showing that Algorithm 5.1 is a PTAS is its running time and filling in the details for the calculation of k, a, b and n_{approx}.

Lemma 5.4. *Algorithm 5.1 can be realized with running time polynomial in the input size for fixed ε.*

Proof. Let L be the bit-length of the input, i.e. the total length of the encodings of p, q, u, v. The decision if r is small, i.e. $r < 6 \cdot \frac{1}{\varepsilon}$ can be made by calculating $pu < 6qv$ which can be done in $\mathcal{O}(\mathrm{M}(L))$ time.

If $r < 6 \cdot \frac{1}{\varepsilon}$ the exact algorithm is executed (see Section 2.2.1). Since the disks to be packed can be represented by the inequality $x^2 + y^2 \leq 1$ and the container by $x^2 + y^2 \leq r^2$ and at most $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ unit disks can be packed into the container, the running time is by Corollary 2.3

$$\left(\frac{1}{\varepsilon}\right)^{\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)} \mathcal{O}(L \cdot \mathrm{M}(L)).$$

Observe, that this bound depends only polynomially on the length L of the input.

Otherwise, first the algorithm computes k. This can be done in $\mathcal{O}(L)$ time since $\lfloor \log p \rfloor + 1$ is the bit length of p and therefore we can compute $\lfloor \log p \rfloor + 1$ and $\lfloor \log q \rfloor$ by counting.

The value for a can be computed by calculating π with precision k, i.e. calculating the binary representation of π up to the bit representing $\frac{1}{2^k}$. In the same way, we can compute b by calculating $\sqrt{12}$ up to the bit representing $\frac{1}{2^k}$ and then adding $\frac{1}{2^k}$. These calculations can be done in $\mathcal{O}(\mathrm{M}(k)\log k)$ or $\mathcal{O}(\mathrm{M}(k))$ time, respectively (see e.g. [14]). Since $k = \mathcal{O}(\log r)$, the running time to compute a and b is in $\mathcal{O}(L^3)$.

The computation of n_{approx} is then a constant number of arithmetic operations on at most L-bit integers, so in total it is possible in $\mathcal{O}(L^3)$ time.

In total, the running time is polynomial in L for constant ε. □

Lemmas 5.1 to 5.4 together show the following theorem.

Theorem 5.5. *Algorithm 5.1 is a PTAS for finding the maximal number of disks that can be packed into a circle of given radius.*

5.2 Packing Unit Disks into Simple Polygons

Now, we use the ideas given in the previous section to pack disks into polygons. For now, we assume that the polygon is part of the problem description and not part of the input. Hence, the input is again just a scaling factor r.

5.2.1 Packing into Triangles

First, we consider the slightly easier problem of finding the maximal number $n_{\max}$ of unit disks that can be packed into a fixed triangle with side length a, b, c scaled by r given as fraction $\frac{p}{q}$ with $p, q \in \mathbb{N}_+$. The approximation algorithm is mainly analogous to the algorithm described in Section 5.1 but with different constants that also depend on the side lengths of the container triangle.

Algorithm 5.2:

```
  Input: Number $r$ given as $\frac{p}{q}$, parameter $1 > \varepsilon > 0$ given as $\frac{u}{v}$, with $p, q, u, v$
         positive integers
  Output: Nonnegative integer $n_{\text{approx}}$
1 $k = \log(ab) + 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)$ ;
2 Compute some $h$ with $\sin\gamma \geq h \geq \sin\gamma - 2^{-k}$;
3 if $r < \frac{6(a+b+c)+7}{abh} \cdot \frac{1}{\varepsilon}$ then
4     Compute $n_{\text{approx}}$ with the exact algorithm;
5 end
6 else
7     Compute some $g$ with $\sqrt{12} \leq g \leq \sqrt{12} + 2^{-k}$;
8     $n_{\text{approx}} = \left\lceil \frac{1}{g}\left(h\frac{ab}{2}r^2 - 3(a+b+c)r\right)\right\rceil$;
9 end
10 return $n_{\text{approx}}$;
```

The analysis of the algorithm works analogously to the analysis in Section 5.1. In the proof of Lemma 5.1, replace the container circle by the triangle scaled by r and the circle of radius $r-3$ by a triangle similar to the container triangle such that each point in the smaller triangle has distance at least 3 to the boundary of the container triangle. We estimate the area of the smaller triangle by the area of the container triangle minus $3(a+b+c)r$. It is well known that the area of the triangle is $\frac{ab\sin\gamma}{2}$. In this way, we get the following lemma.

Lemma 5.6. $n_{\max} \geq \left\lceil \frac{1}{\sqrt{12}} \left(\frac{ab\sin\gamma}{2} r^2 - 3(a+b+c)r \right) \right\rceil$.

It is easy to see that n_{approx} is at most this lower bound or the optimal value and so $n_{\text{approx}} \leq n_{\max}$, i.e. n_{aprox} disks can indeed be packed. Next, we analyze the approximation factor of the algorithm.

Lemma 5.7. $n_{approx} \geq (1-\varepsilon) \cdot n_{\max}$.

Proof. First, we will show that at least two unit disks can be packed into the container if Lines 7 and 8 get executed in order to apply Theorem 2.1 later in the analysis. The radius d of the incircle of a triangle T is given by the well known formula

$$d = \frac{2 \cdot \text{area}(T)}{\text{perimeter}(T)}.$$

Line 8 gets executed for triangles with inradius

$$\begin{aligned} d &= \frac{ab\sin\gamma}{a+b+c} \cdot r \\ &\geq \frac{ab\sin\gamma}{a+b+c} \cdot \frac{6(a+b+c)+7}{abh \cdot \varepsilon} && \text{(by the bound for } r \text{ in the if-condition)} \\ &\geq \frac{ab\sin\gamma}{a+b+c} \cdot \frac{6(a+b+c)+7}{ab\sin\gamma \cdot \varepsilon} && \text{(since } h \leq \sin\gamma) \\ &\geq 6 \cdot \frac{1}{\varepsilon} > 6 && \text{(since } \varepsilon < 1). \end{aligned}$$

This implies that at least three unit disks can be packed into the container triangle by simply placing them on top of each other.

Now, as before, we calculate a lower bound for n_{approx} from the bounds for g, h. Since $k = \log(ab) + 2(\lfloor \log p \rfloor + 1 + \lfloor \log q \rfloor)$, we have

$$h \geq \sin\gamma - 2^k = \sin\gamma - \frac{1}{2^{\log(ab)+2(\lfloor p \rfloor + 1 + \lfloor q \rfloor)}} \geq \sin\gamma - \frac{1}{ab \cdot 2^{2(\log p - \log q)}} = \sin\gamma - \frac{1}{abr^2}.$$

In the same way we get

$$g \leq \sqrt{12} + \frac{1}{abr^2}.$$

Plugging in these bounds for h and g into the formula for the return value in Line 8 of Algorithm 5.2 gives

$$\begin{aligned}
n_{\text{approx}} \geq & \left\lceil \frac{1}{g}\left(h\frac{ab}{2}r^2 - 3(a+b+c)\right)r \right\rceil \\
\geq & \frac{1}{\sqrt{12}+\frac{1}{abr^2}}\left(\left(\sin\gamma - \frac{1}{abr^2}\right)\frac{ab}{2}r^2 - 3(a+b+c)r\right) \\
> & \left(\frac{1}{\sqrt{12}} - \frac{1}{\sqrt{12}} + \frac{1}{\sqrt{12}+\frac{1}{abr^2}}\right)\left(\sin\gamma - \frac{1}{abr^2}\right)\frac{ab}{2}r^2 - \frac{\pi}{\sqrt{12}}\cdot 3(a+b+c)r \\
= & \frac{1}{\sqrt{12}}\left(\frac{ab\sin\gamma}{2}r^2 - 3(a+b+c)r\right) \\
& - \left(\frac{\sqrt{12}+\frac{1}{abr^2}-\sqrt{12}}{12+\frac{\sqrt{12}}{abr^2}}\sin\gamma + \frac{1}{\sqrt{12}abr^2+1}\right)\frac{ab}{2}r^2 \\
\geq & \frac{1}{\sqrt{12}}\left(\frac{ab\sin\gamma}{2}r^2 - 3(a+b+c)r\right) \\
& - \left(\frac{abr^2}{\left(12\cdot abr^2+\sqrt{12}\right)\cdot 2} + \frac{abr^2}{\left(\sqrt{12}abr^2+1\right)\cdot 2}\right),
\end{aligned}$$

since $\sin\gamma \leq 1$. Further rearranging gives

$$\begin{aligned}
n_{\text{approx}} > & \frac{1}{\sqrt{12}}\left(\frac{ab\sin\gamma}{2}r^2 - 3(a+b+c)r\right) - \left(\frac{1}{24} + \frac{1}{2\sqrt{12}}\right) \\
> & \frac{1}{\sqrt{12}}\left(\frac{ab\sin\gamma}{2}r^2 - 3(a+b+c)r\right) - 1. \qquad (5.3)
\end{aligned}$$

As we explained earlier, we return the optimal value if not at least two unit disks (we showed three indeed) can be packed into the container. Hence, we have by Theorem 2.1 $n_{\max} \leq \frac{\pi}{\sqrt{12}}\frac{ab\sin\gamma}{2}$. Together with Eq. (5.3) we get

$$\begin{aligned}
\frac{n_{\text{approx}}}{n_{\max}} \geq & \frac{\frac{1}{\sqrt{12}}\left(\frac{ab\sin\gamma}{2}r^2 - 3(a+b+c)r\right) - 1}{\frac{1}{\sqrt{12}}\frac{ab\sin\gamma}{2}r^2} \\
= & 1 - \frac{3(a+b+c)}{\frac{ab\sin\gamma}{2}r} - \frac{2\sqrt{12}}{ab\sin\gamma\cdot r^2} \\
\geq & 1 - \frac{6(a+b+c)+2\sqrt{12}}{ab\sin\gamma\cdot r},
\end{aligned}$$

since we may assume that $r \geq 1$. Otherwise we can calculate the optimal solution in constant time. Furthermore, the approximate value is only computed for $r \geq \frac{6(a+b+c)+7}{ab \sin \gamma} \cdot \frac{1}{\varepsilon}$, so we get

$$\begin{aligned}\frac{n_{\text{approx}}}{n_{\max}} &\geq 1 - \frac{6(a+b+c)+2\sqrt{12}}{ab\sin\gamma} \cdot \frac{ab\sin\gamma}{6(a+b+c)+7} \cdot \varepsilon \\ &= 1 - \frac{6(a+b+c)+2\sqrt{12}}{6(a+b+c)+7} \cdot \varepsilon \\ &> 1 - \varepsilon.\end{aligned}$$

□

It remains to analyze the running time of the algorithm.

Lemma 5.8. *Algorithm 5.2 can be realized with running time polynomial in the input for fixed ε.*

Proof. Analogous to the algorithms studied before, $\lfloor \log p \rfloor + 1$, $\lfloor \log q \rfloor$ be calculated in time polynomial in the input size. We do not need the exact value of $\log ab$ but an upper bound for it, so we can simply count the bits in the representations of a and b which can be done in constant time since a and b are not part of the input. Then, we can compute g in polynomial time as before. Also, since $\sin\gamma = \sqrt{1 - \left(\frac{a^2+b^2-c^2}{2ab}\right)^2}$, we can use the same techniques to calculate h as for the calculation of g. The running time of the exact algorithm is analogous to the algorithms described before. In total, the running time is polynomial in the input size for fixed ε. □

Altogether we have the following result.

Theorem 5.9. *Algorithm 5.2 is a PTAS for calculating the maximal number of unit disks that can be packed into a fixed triangle scaled by a given factor r.*

5.2.2 Packing into Simple Polygons

Now, we turn to packing unit disks into a fixed simple polygon scaled by r (given as fraction of positive integers p and q). The idea is to calculate the optimal solution for small r depending on $\frac{1}{\varepsilon}$ by using the approach discussed in Section 2.2.1 (see Corollary 2.3). For large r we triangulate the polygon and reduce the problem to finding approximations in each triangle and return the summed up approximations of all triangles. For describing the algorithm in more detail, we need the following definitions. Let a triangulation $\mathcal{T}$ of a polygon with v vertices be given as a set of triangles $\mathcal{T} = \{T_1, \ldots, T_{v-3}\}$ with T_i having side lengths a_i, b_i, c_i and angle γ_i between the sides of length a_i and b_i. We assume in the following, that the container polygon has v vertices.

For the analysis of the algorithm, we will first give a lower bound on the number $N_{\max}$ of unit disks that can be packed. It follows directly from Lemma 5.6.

Algorithm 5.3:

```
Input: Number $r$ given as $\frac{p}{q}$, parameter $1 > \varepsilon > 0$ given as $\frac{u}{v}$, with $p, q, u, v$
       positive integers
Output: Nonnegative integer $N_{\text{approx}}$
1  Compute a triangulation $\mathcal{T} = \{T_1, \ldots, T_{v-3}\}$ for the container polygon;
2  $k = \log\left(\max_{i \in \{1,\ldots,v-3\}} (a_i b_i)\right) + 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)$;
3  for $i := 1$to $v - 3$ do
4      Compute some $h_i$ with $\sin\gamma_i \geq h_i \geq \sin\gamma_i - 2^{-k}$;
5  end
6  Calculate $z = \max_{i \in \{1,\ldots,v-3\}} \left(\frac{(8(a_i+b_i+c_i)+7)(a_i b_i h_i)+(a_i+b_i+c_i)^2}{(a_i b_i h_i)^2}\right)$;
7  if $r < z \cdot \frac{1}{\varepsilon}$ then
8      Compute $N_{\text{approx}}$ with the exact algorithm;
9  end
10 else
11     Compute some $g$ with $\sqrt{12} \leq g \leq \sqrt{12} + 2^{-k}$;
12     for $i := 1$to $v - 3$ do
13         Calculate $n_{\text{approx}_i} = \left\lceil \frac{1}{g}\left(h_i \frac{a_i b_i}{2} r^2 - 3(a_i + b_i + c_i) r\right) \right\rceil$;
14     end
15     Compute $N_{\text{approx}} = \sum_{i=1}^{v-3} n_{\text{approx}_i}$;
16 end
17 return $N_{\text{approx}}$;
```

Lemma 5.10. *Let $n_{\max_i}$ be the maximum number of unit disks that can be packed in $T_i \in \mathcal{T}$. Then $N_{\max} \geq \sum_{i=1}^{v-3} n_{\max_i} \geq \sum_{i=1}^{v-3} \left\lceil \frac{1}{\sqrt{12}}\left(\frac{a_i b_i \sin\gamma_i}{2} r^2 - 3(a_i + b_i + c_i) r\right) \right\rceil$.*

Since $n_{\text{approx}_i} \leq n_{\max_i}$ as described in the previous section, Lemma 5.10 directly implies that $N_{\text{approx}} \leq N_{\max}$ and therefore N_{approx} unit disks can surely be packed into the scaled polygon. Next, we will prove the approximation factor of the algorithm.

Lemma 5.11. *$N_{approx} \geq (1 - \varepsilon) N_{\max}$.*

Proof. First, we need a lower bound on N_{approx} if not the optimal value is returned and therefore we will give lower bounds for the n_{approx_i}. Since $k = \log\left(\max_i (a_i b_i)\right) + 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)$, we get

$$\begin{aligned} h_i \geq h_i - 2^{-k} &= \sin\gamma_i - \frac{1}{2^{\log\left(\max_i (a_i b_i)\right) + 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)}} \\ &\geq \sin\gamma_i - \frac{1}{2^{\log((a_i b_i)) + 2(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)}} = \sin\gamma_i - \frac{1}{a_i b_i r^2}, \end{aligned}$$

for all $i = 1, \dots, v-3$. Analogously, we get $g \geq \sqrt{12} + \frac{1}{a_i b_i r^2}$ for all $i = 1, \dots, v-3$. As in the proof of Lemma 5.7, this gives $n_{\text{approx}_i} \geq \frac{1}{\sqrt{12}}\left(\frac{a_i b_i \sin\gamma_i}{2} r^2 - 3(a_i+b_i+c_i)r\right) - 1$. So, together we get

$$N_{\text{approx}} \geq \sum_{i=1}^{v-3} n_{\text{approx}_i} \geq \sum_{i=1}^{v-3} \left(\frac{1}{\sqrt{12}} \left(\frac{a_i b_i \sin\gamma_i}{2} r^2 - 3(a_i+b_i+c_i)r \right) - 1 \right).$$

Next, we need an upper bound for $N_{\max}$. Consider an optimal packing of $N_{\max}$ disks together with the triangulation $\mathcal{T}$. Observe, that we cannot directly apply Fejes Tóths Theorem 2.1 to the container since it is only stated for convex regions. Therefore, we consider the container split up into triangles but have to be careful since we might cut packed disks into pieces when dividing the polygon into triangles. Let C_i be the set of disks in the optimal packing with their disk centers in T_i. Let d be the inradius of T_i. Then define T_i' to be the triangle similar to T_i with the same center of the incircle but with inradius $d+1$. Then, the disks in C_i are completely contained in T_i' (see Fig. 5.2).

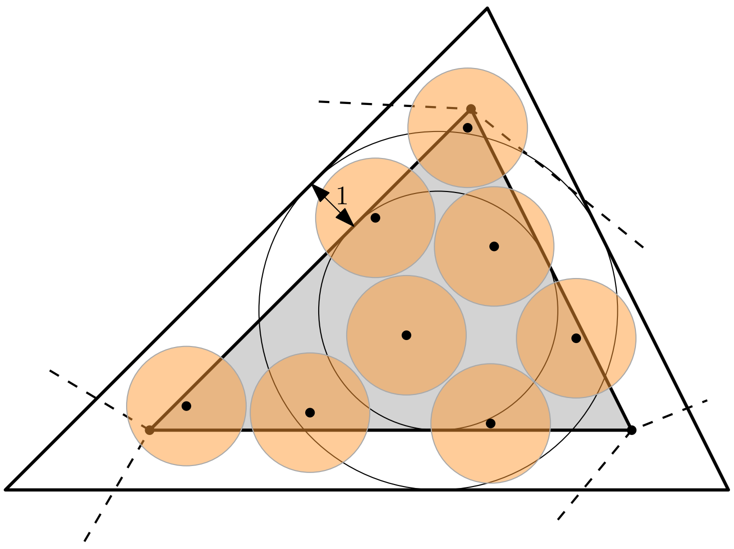


Figure 5.2: All disks with center in the grey triangle T_i of the triangulation are contained in the larger triangle T_i' that is similar to T_i and has the same incircle center and incircle radius enlarged by one.

The area of T_i' can be obtained in the following way. The inradius d of T_i is given by the well known formula

$$\begin{aligned} d &= \frac{2 \cdot \text{area}(T_i)}{\text{perimeter}(T_i)} \\ &= \frac{a_i b_i \sin\gamma_i \cdot r^2}{(a_i+b_i+c_i)r}. \end{aligned} \tag{5.4}$$

T_i' is a scaled copy of T_i so the following equality must hold for some constant λ:

$$d+1=\frac{\lambda^2 a_i b_i \sin\gamma_i \cdot r}{\lambda(a_i+b_i+c_i)}.$$

It follows that $\lambda = 1+\frac{a_i+b_i+c_i}{a_i b_i \sin\gamma_i \cdot r}$ and therefore

$$\begin{aligned}
\operatorname{area}(T_i') &= \lambda^2 \operatorname{area}(T_i)\\
&= \left(1+\frac{a_i+b_i+c_i}{a_i b_i \sin\gamma_i \cdot r}\right)^2 \frac{a_i b_i \sin\gamma_i}{2} r^2\\
&= \left(1+\frac{2(a_i+b_i+c_i)}{a_i b_i \sin\gamma_i \cdot r}+\frac{(a_i+b_i+c_i)^2}{a_i^2 b_i^2 \sin^2\gamma_i \cdot r^2}\right)\frac{a_i b_i \sin\gamma_i}{2} r^2\\
&= \frac{a_i b_i \sin\gamma_i}{2} r^2 + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_i b_i \sin\gamma_i}.
\end{aligned}$$

Observe, that the inradius of T_i' is by Eq. (5.4)

$$\frac{a_i b_i \sin\gamma_i \cdot r^2}{(a_i+b_i+c_i)r}+1 \geq \frac{a_i b_i \sin\gamma_i}{(a_i+b_i+c_i)} \cdot \frac{(8(a_i+b_i+c_i)+7)(a_i b_i h_i)+(a_i+b_i+c_i)^2}{(a_i b_i h_i)^2 \cdot \varepsilon}+1,$$

by Line 7 in Algorithm 5.3. Hence,

$$\frac{a_i b_i \sin\gamma_i \cdot r^2}{(a_i+b_i+c_i)r}+1 \geq 9.$$

So, at least four unit disks can be packed into T_i'. Hence, we can apply Theorem 2.1 to T_i' and get

$$|C_i| \leq \frac{1}{\sqrt{12}}\left(\frac{a_i b_i \sin\gamma_i}{2} r^2 + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_i b_i \sin\gamma_i}\right).$$

Since for every disk in the optimal packing there exists an i such that its center is contained in C_i, we have

$$N_{\max} \leq \sum_{i=1}^{v-3} |C_i| \leq \sum_{i=1}^{v-3} \frac{1}{\sqrt{12}}\left(\frac{a_i b_i \sin\gamma_i}{2} r^2 + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_i b_i \sin\gamma_i}\right). \tag{5.5}$$

In order to complete the proof, we will show that $n_{\text{approx}_i} \geq (1-\varepsilon)|C_i|$ for all $1 \leq i \leq v-3$.

$$\begin{aligned}
\frac{n_{\text{approx}_i}}{|C_i|} &\geq \frac{\frac{1}{\sqrt{12}}\left(\frac{a_ib_i\sin\gamma_i}{2}r^2 - 3(a_i+b_i+c_i)r\right) - 1}{\frac{1}{\sqrt{12}}\left(\frac{a_ib_i\sin\gamma_i}{2}r^2 + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i}\right)} \\
&= \frac{\frac{1}{\sqrt{12}}\left(\frac{a_ib_i\sin\gamma_i}{2}r^2 - 4(a_i+b_i+c_i)r + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i} - \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i}\right) - 1}{\frac{1}{\sqrt{12}}\left(\frac{a_ib_i\sin\gamma_i}{2}r^2 + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i}\right)} \\
&= 1 - \frac{\frac{1}{\sqrt{12}}\left(4(a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i}\right) + 1}{\frac{1}{\sqrt{12}}\left(\frac{a_ib_i\sin\gamma_i}{2}r^2 + (a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i}\right)} \\
&> 1 - \frac{\frac{1}{\sqrt{12}}\left(4(a_i+b_i+c_i)r + \frac{(a_i+b_i+c_i)^2}{2a_ib_i\sin\gamma_i}\right) + 1}{\frac{1}{\sqrt{12}}\frac{a_ib_i\sin\gamma_i}{2}r^2} \\
&= 1 - \left(\frac{8(a_i+b_i+c_i)}{a_ib_i\sin\gamma_i\cdot r} + \frac{(a_i+b_i+c_i)^2}{(a_ib_i\sin\gamma_i\cdot r)^2} + \frac{2\sqrt{12}}{a_ib_i\sin\gamma_i\cdot r^2}\right) \\
&\geq 1 - \frac{\left(8(a_i+b_i+c_i) + 2\sqrt{12}\right)a_ib_i\sin\gamma_i + (a_i+b_i+c_i)^2}{(a_ib_i\sin\gamma_i)^2 r},
\end{aligned}$$

since we can assume $r \geq 1$ as described above. Furthermore, the approximate solution is only computed for $r \geq \frac{(8(a_i+b_i+c_i)+7)a_ib_i\sin\gamma_i+(a_i+b_i+c_i)^2}{(a_ib_i\sin\gamma_i)^2}$, so we get

$$\frac{n_{\text{approx}_i}}{|C_i|} > 1-\varepsilon.$$

This immediately gives

$$N_{\text{approx}} \geq \sum_{i=1}^{v-3} n_{\text{approx}_i} \geq (1-\varepsilon)\sum_{i=1}^{v-3}|C_i| \geq (1-\varepsilon)N_{\max},$$

by Eq. (5.5). □

It remains to analyze the running time of the algorithm.

Lemma 5.12. *Algorithm 5.3 can be realized in time polynomial in the input size for fixed ε.*

Proof. The analysis of the running time is similar to the one for triangular containers. If the algorithm computes the optimal value for the maximal number of disks that can be packed by solving first-order-formulas as described in Section 2.2.1, we have $r = \mathcal{O}\left(\frac{1}{\varepsilon}\right)$ (see Lines 7 and 8 in Algorithm 5.3), since $a_i, b_i, c_i, \sin\gamma_i$ are independent of the input for all $i = 1, \ldots, v-3$ and therefore constants. So, in this case, the running time is polynomial in the input size by Corollary 2.3. Now we study the case where the algorithm returns the approximate solution N_{approx}. Initially, assume the container polygon is given together with a triangulation $\mathcal{T} = \{T_1, \dots, T_{v-3}\}$ and the triangle T_i is given by its side length a_i, b_i, c_i. Then, the algorithm computes constantly often approximations n_{approx_i} analogously to the algorithm for triangles described in the previous section. Since the running time of the algorithm for triangles is polynomial in the input size, the running time for the algorithm described in this section is polynomial in the input size, too. Unfortunately, if the triangulation of the container polygon is not given, we have to compute it. Since v is a constant, we can compute the triangles in constant time but it may happen, that the side lengths of the triangles have no rational values. In this case, we have to use the same trick as for the angles γ_i and $\sqrt{12}$, namely computing the side lengths with suitable precision depending on r. In this way, we can achieve overall polynomial running time in the input size. □

Altogether we have the following result.

Theorem 5.13. *Algorithm 5.3 is a PTAS for calculating the maximal number of unit disks that can be packed into a fixed simple polygon scaled by a given factor r.*

5.3 Packing Unit Balls into Spheres

In this section, we use the ideas from packing unit disks into circles to find the maximum number $n_{\max}$ of unit balls that can be packed into a container sphere of radius r. The exact solution can again be obtained by solving first-order-formulas as discussed in Section 2.2.1. As for the 2D-problem, the algorithm solves the problem exactly for small radii and calculates an approximation for bigger radii using the density of an optimal packing of infinitely many balls in 3D proven by Hales et al. [29, 30]. Basically, the 3D-algorithm is the same as the 2D-algorithm (Algorithm 5.1), but of course it uses other constants.

We will do the analysis of the algorithm in the same order as for the 2D-algorithm. First, we show that n_{approx} balls can indeed be packed by giving a lower bound for $n_{\max}$ and proving that n_{approx} is smaller than this bound. Then, we will give an upper bound for $n_{\max}$ and use it to show that the algorithm computes a $(1-\varepsilon)$-approximation. Finally, we analyze the running time to show that the algorithm is truly a PTAS.

Lemma 5.14. $n_{\max} \geq \left\lceil \frac{\pi}{\sqrt{18}}(r-4)^3 \right\rceil$.

Proof. A packing with optimal density of infinitely many unit balls in 3D is the face-centered cubic packing (see Section 6.2 for details). The volume of a cell in the Voronoi-Diagram of the ball centers in the FCC-packing is $4\sqrt{2}$ (see e.g. [29]). Such a Voronoi-cell is contained in a sphere of radius 2 [29]. Now, we consider the intersection of the FCC-packing with a container sphere. Since the diameter of a cell is at most 4, every cell that intersects a sphere with radius $r-4$ centered at the origin is contained in a sphere with radius r centered at the origin. Therefore, to get a lower bound for $n_{\max}$ we divide the

Algorithm 5.4:

```
Input: Number $r \geq 2$ given as $\frac{p}{q}$, parameter $\varepsilon > 0$ given as $\frac{u}{v}$, with $p, q, u, v$
       positive integers
Output: Nonnegative integer $n_{\text{approx}}$
1  if $r < 32992 \cdot \frac{1}{\varepsilon}$ then
2      Compute $n_{\text{approx}}$ with the exact algorithm;
3  end
4  else
5      $k = 3(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor) + 6$ ;
6      compute some $a$ with $\pi \geq a \geq \pi - 2^{-k}$;
7      compute some $b$ with $\sqrt{2} \leq b \leq \sqrt{2} + 2^{-k}$;
8      $n_{\text{approx}} = \left\lceil \frac{a(r-4)^3}{3b} \right\rceil$;
9  end
10 return $n_{\text{approx}}$;
```

volume of the sphere with radius $r-4$ by the volume of a Voronoi-cell:

$$n_{\max} \geq \left\lceil \frac{\frac{4}{3}\pi}{4\sqrt{2}}(r-4)^3 \right\rceil = \left\lceil \frac{\pi}{\sqrt{18}}(r-4)^3 \right\rceil.$$

$\square$

If n_{approx} is not computed with the exact algorithm, we know that $n_{\text{approx}} \leq \frac{\pi}{3\sqrt{2}}(r-4)^3$, since $a \leq \pi$ and $b \geq \sqrt{2}$. So following Lemma 5.14, n_{approx} unit balls can surely be packed. Next, we need an upper bound for $n_{\max}$. The formulation of Kepler's conjecture which has been proven by Hales et al. is as follows: For any ball packing there exists a constant c such that the volume of the intersection of the union of the unit balls with a container sphere of radius r is at most $\frac{\pi}{\sqrt{18}} + \frac{c}{r}$ times the volume of the container sphere. For our purpose, we need a slightly stronger statement, namely that c is universal, i. e., independent of the particular packing. Hales et al. state in their paper [30] that this is possible but do not show the solution. We do this in the following chapter (Chapter 6) by modifying Hales's proof from [29] and come up with the constant $c = 24373$ (Theorem 6.24). This bound also takes balls only partially contained in the container into account.

The actual maximum ratio between the volume of the union of balls completely contained in the container sphere and the volume of the container sphere is $\frac{n_{\max}\frac{4}{3}\pi}{\frac{4}{3}\pi r^3}$. Together, this gives

$$\frac{n_{\max}\frac{4}{3}\pi}{\frac{4}{3}\pi r^3} \leq \frac{\pi}{\sqrt{18}} + 24373\frac{1}{r},$$

which implies

$$n_{\max} \le \frac{\pi}{\sqrt{18}} r^3 + 24373 r^2.$$

Hence, we have shown the following lemma.

Lemma 5.15. *For $r \ge 1$ it holds that $n_{\max} \le \frac{\pi}{\sqrt{18}} r^3 + 24373 r^2$.*

Now, we use the upper bound to prove the approximation ratio of Algorithm 5.4.

Lemma 5.16. $n_{approx} \ge (1-\varepsilon) n_{\max}$

Proof. The proof is analogous to the proof of Lemma 5.3. As before, we compute bounds for a and b. Since $k = 3(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor) + 6$, we get

$$a \ge \pi - 2^{-k} = \pi - \frac{1}{2^{3(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor)+6}} \ge \pi - \frac{1}{2^{3(\log p - \log q)} \cdot 64} = \pi - \frac{1}{64 r^3},$$

similarly, we get

$$b \le \sqrt{2} + 2^{-k} \le \sqrt{2} + \frac{1}{64 r^3}.$$

n_{approx} is either the exact value or we have

$$n_{\text{approx}} \ge \left\lceil \frac{a}{3b} (r-4)^3 \right\rceil$$

With the bounds for a and b we get

$$\begin{aligned}
n_{\text{approx}} &\ge \frac{\pi - \frac{1}{64r^3}}{3\left(\sqrt{2} + \frac{1}{64r^3}\right)} (r-4)^3. \\
&= \left(\frac{\pi - \frac{1}{64r^3}}{3\left(\sqrt{2} + \frac{1}{64r^3}\right)} + \frac{\pi}{\sqrt{18}} - \frac{\pi}{\sqrt{18}} \right) (r-4)^3 \\
&= \frac{\pi}{\sqrt{18}} (r-4)^3 - \frac{\pi\left(\sqrt{18} + \frac{3}{64r^3}\right) - \sqrt{18}\left(\pi - \frac{1}{64r^3}\right)}{\sqrt{18}\left(\sqrt{18} + \frac{3}{64r^3}\right)} (r-4)^3 \\
&= \frac{\pi}{\sqrt{18}} (r-4)^3 - \frac{3\pi + \sqrt{18}}{18 \cdot 64 r^3 + 3\sqrt{18}} \left(r^3 - 12r^2 + 48r - 64\right) \\
&\ge \frac{\pi}{\sqrt{18}} (r-4)^3 - \frac{3\pi + \sqrt{18}}{1152 r^3} \left(r^3 + 48r\right) \\
&\ge \frac{\pi}{\sqrt{18}} (r-4)^3 - \frac{3\pi + \sqrt{18}}{1152 r^3} \left(49 r^3\right) && \text{(since } r \ge 1\text{)} \\
&= \frac{\pi}{\sqrt{18}} (r-4)^3 - \frac{49\left(3\pi + \sqrt{18}\right)}{1152} \\
&\approx \frac{\pi}{\sqrt{18}} (r-4)^3 - 0.59 < \frac{\pi}{\sqrt{18}} (r-4)^3 - 1. && (5.6)
\end{aligned}$$

Next, we will give the approximation factor using the bound from Eq. (5.6) and the upper bound for $n_{\max}$ given in Lemma 5.15:

$$\begin{aligned}
\frac{n_{\text{approx}}}{n_{\max}} &\geq \frac{\frac{\pi}{\sqrt{18}}(r-4)^3 - 1}{\frac{\pi}{\sqrt{18}}r^3 + 24373r^2} \\
&> \frac{\pi\left(r^3 - 12r^2 - 64\right)}{\pi r^3 + 24373\sqrt{18}r^2} && \text{(since } r \geq 1\text{)} \\
&= 1 - \frac{\left(12\pi + 24373\sqrt{18}\right)r^2 + 64\pi}{\pi r^3 + 24373\sqrt{18}r^2} \\
&\geq 1 - \frac{\left(76\pi + 24373\sqrt{18}\right)r^2}{\pi r^3 + 24373\sqrt{18}r^2} && \text{(also since } r \geq 1\text{)} \\
&> 1 - \frac{\left(76\pi + 24373\sqrt{18}\right)r^2}{\pi r^3} \\
&= 1 - \frac{\left(76\pi + 24373\sqrt{18}\right)}{\pi r}.
\end{aligned}$$

Since we calculate the optimal solution for $r < \frac{32992}{\varepsilon}$ we can assume $r \geq \frac{32992}{\varepsilon}$. Plugging in this bound for r gives

$$\frac{n_{\text{approx}}}{n_{\max}} > 1 - \varepsilon.$$

□

It remains to analyze the running time. Analogous to the 2D-algorithm $\lfloor \log p \rfloor + 1$, $\lfloor \log q \rfloor$, a, and b can be calculated in time polynomial in the input size. Therefore, the critical term in the calculation of the running time is the running time of the exact algorithm, which is polynomial in the input size by Corollary 2.3 for fixed ε. So, we get the following lemma.

Lemma 5.17. *Algorithm 5.4 can be realized with running time polynomial in the inputsize for fixed ε.*

Lemmas 5.14, 5.16 and 5.17 directly imply the following theorem.

Theorem 5.18. *Algorithm 5.4 is a PTAS for computing the maximum number of unit balls that can be packed into a sphere with given radius r.*

5.4 Packing Unit Disks into Thick Convex Containers

In this section, we will again pack unit disks into polygons. This time, the container polygon is part of the input and not part of the problem description with the extra requirement that the container polygon is c-thick by Definition 3.12 for a constant c. Instead of Definition 3.12, we will use the following definition in this section after we have proven that they are equivalent for convex polygons.

Definition 5.19 (c-chunkiness)**.** *Let P be a polygon, A its area and U its perimeter. We call P c-chunky if and only if $A \geq c \cdot U^2$.*

Lemma 5.20. *If a convex polygon P is c-chunky then there exists a c' such that P is c'-thick. If P is c-thick then there exists a c' such that P is c'-chunky.*

Proof. Observe that the perimeter U of P is at least twice the diameter d of P. Hence, $A \geq cU^2$ implies $A \geq 4cd^2$. Consider the smallest enclosing circle of P with radius s. Since the smallest enclosing circle is defined by at most three points of P, $s \leq \frac{d}{\sqrt{3}}$ (consider an equilateral triangle). For convex polygons, the perimeter is smaller than the perimeter of any enclosing circle. Hence, $A \geq cd^2$ implies $A \geq c \cdot \frac{3}{4\pi^2}U^2$. □

The idea of the algorithm is analogous to the one for packing unit disks into a circle (see Section 5.1). For small areas of P, where the definition of small depends on ε, we compute the optimal solution using the approach given in Section 2.2.1. Otherwise, we multiply the area of P without a strip of width 3 at its boundary by the density of the optimal infinite packing for the plane. Again, we will need some correction terms. Algorithm 5.5 gives more details. We assume that the container polygon P is c-chunky for a constant c and given by the rational coordinates of its vertices $v_1, \dots v_l$.

Algorithm 5.5:

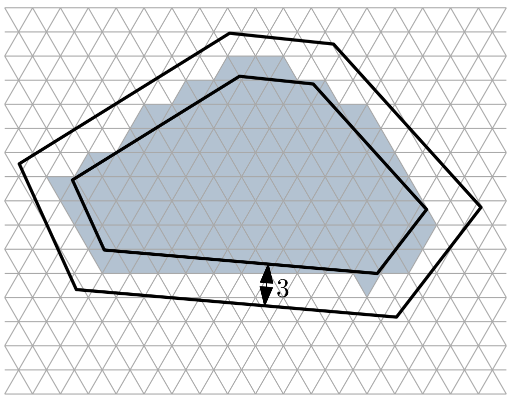
```
   Input: Convex polygon $P = (v_1, \dots, v_l)$, parameter $\varepsilon > 0$ given as $\frac{u}{v}$, with $u, v$
          positive integers
   Output: Nonnegative integer $n_{\text{approx}}$
 1 Compute area $A$ of $P$;
 2 if At least two unit disks can be packed into $P$ and $A \geq \frac{43}{c\varepsilon^2}$ then
 3     $k = \lfloor \log(A+1) \rfloor + 1$ ;
 4     compute some $a$ with $\sqrt{\frac{A}{c}} \leq a \leq \sqrt{\frac{A}{c}} + 2^{-k}$;
 5     compute some $b$ with $\sqrt{12} \leq b \leq \sqrt{12} + 2^{-k}$;
 6     $n_{\text{approx}} = \left\lceil \frac{A-3a}{b} \right\rceil$;
 7 end
 8 else
 9     Compute $n_{\text{approx}}$ with the exact algorithm;
10 end
11 return $n_{\text{approx}}$;
```

Since the idea of the algorithm is very similar to the one for packing unit disks into circles, the analysis is analogous. We start by giving a lower bound for $n_{\max}$ that shows that n_{approx} unit disks can surely be packed. Afterwards, we will prove the approximation factor of Algorithm 5.5 and then its running time.

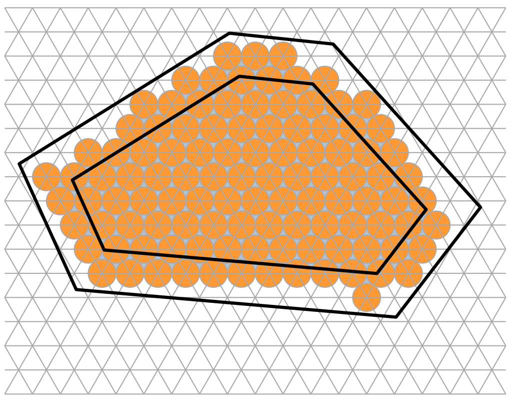
Lemma 5.21. $n_{\max} \geq \left\lceil \frac{A-3\sqrt{\frac{A}{c}}}{\sqrt{12}} \right\rceil$, *where A is the area of the c-chunky container polygon P.*

Proof. The proof is mostly analogous to the proof of Lemma 5.1 and additionally uses Definition 5.19.

Consider a polygon P' inside P that consists of all points that have distance at least 3 to the boundary of P together with the triangular grid of the optimal packing of unit disks in the plane (see Fig. 5.3i). Vertices of triangles at least partially contained in P' have distance at least one from the boundary of P. Hence, unit disks placed with their centers at these vertices are completely contained in P (see Fig. 5.3ii). Let A be the area of P and U its perimeter. Then, the area A' of P' can be approximated with $A' \geq A - 3U$ since P is convex. Using Definition 5.19 gives $A' \geq A - \sqrt{\frac{A}{c}}$. Since each triangle of the grid contains half a unit disk, dividing this approximation of A' by the area of a triangle $\sqrt{3}$ gives a lower bound for $2 \cdot n_{\max}$. □



(i) The vertices of triangles intersecting smaller polygon have distance at least one from the boundary of the container polygon.


(ii) Disks placed on the vertices of triangles intersecting the smaller polygon are completely contained in the container polygon.

Since $a \geq \sqrt{\frac{A}{c}}$ and $b \geq \sqrt{12}$, Lemma 5.21 implies that n_{approx} unit disks can surely be packed. Next, we prove the approximation factor of Algorithm 5.5.

Lemma 5.22. $n_{approx} \geq (1-\varepsilon)n_{\max}$

Proof. As in the many proofs of approximation factors before, we first determine upper bounds for a and b.

Since $k = \lfloor \log(A+1) \rfloor + 1$, we have

$$\begin{aligned} a &\leq \sqrt{\frac{A}{c}} + 2^{-k} = \sqrt{\frac{A}{c}} + \frac{1}{2^{\lfloor \log(A+1) \rfloor + 1}} \\ &\leq \sqrt{\frac{A}{c}} + \frac{1}{A+1}. \end{aligned}$$

Similarly, we get

$$b \le \sqrt{12} + \frac{1}{A+1}.$$

Now, as before, we use these bounds to obtain a lower bound for n_{approx} in case it is not calculated with the exact algorithm. Then,

$$\begin{aligned}
n_{\text{approx}} &= \left\lceil \frac{A-3a}{b} \right\rceil \\
&\ge \frac{A - 3\left(\sqrt{\frac{A}{c}} + \frac{1}{A+1}\right)}{\sqrt{12} + \frac{1}{A+1}} \\
&\ge \frac{A}{\sqrt{12} + \frac{1}{A+1}} - \frac{3\left(\sqrt{\frac{A}{c}} + \frac{1}{A+1}\right)}{\sqrt{12}} \\
&= \frac{A}{\sqrt{12}}\left(1 - \frac{\frac{1}{A+1}}{\sqrt{12} + \frac{1}{A+1}}\right) - \frac{3\sqrt{\frac{A}{c}}}{\sqrt{12}} - \frac{3\frac{1}{A+1}}{\sqrt{12}} \\
&\ge \frac{A - 3\sqrt{\frac{A}{c}}}{\sqrt{12}} - \left(\frac{A}{12(A+1)} + \frac{3}{\sqrt{12}(A+1)}\right) \\
&= \frac{A - 3\sqrt{\frac{A}{c}}}{\sqrt{12}} - \frac{A + 3\sqrt{12}}{12(A+1)} \\
&\ge \frac{A - 3\sqrt{\frac{A}{c}}}{\sqrt{12}} - 1. && (5.7)
\end{aligned}$$

By the upper bound $n_{\max} \ge \frac{A}{\sqrt{12}}$ that follows from Theorem 2.1 and the lower bound just given in Eq. (5.7), we get

$$\begin{aligned}
\frac{n_{\text{approx}}}{n_{\max}} &\ge \frac{\frac{A-3\sqrt{\frac{A}{c}}}{\sqrt{12}} - 1}{\frac{A}{\sqrt{12}}} \\
&= 1 - \frac{3}{\sqrt{c \cdot A}} - \frac{\sqrt{12}}{A}.
\end{aligned}$$

Observe that the convexity of P implies that $c \le 1$. Since at least two unit disks can be packed if we do not return the optimal value, we have $A \ge 1$. Hence,

$$\begin{aligned}
\frac{n_{\text{approx}}}{n_{\max}} &\ge 1 - \frac{3 + \sqrt{12}}{\sqrt{c \cdot A}} \ge 1 - \frac{6.5}{\sqrt{c \cdot A}} \\
&\ge 1 - \varepsilon,
\end{aligned}$$

since $A \ge \frac{43}{c\varepsilon^2}$ if we do not return the optimal solution. Summarizing, Algorithm 5.5 returns an $(1-\varepsilon)$-approximation if it does not return the optimal solution. □

It remains to analyze the running time.

Lemma 5.23. *Algorithm 5.5 can be implemented with polynomial running time.*

Proof. The area A of P in Line 1 can be computed with Gauss's well known area formula $\frac{1}{2}\left|\sum_{i=1}^{k}(y_i + y_{i+1})(x_i - x_{i+1})\right|$ where (x_i, y_i) are the coordinates of the v_i. Observe that the formula is polynomial and hence, the bit-representation of A is polynomial. The condition of the if-statement in Line 2 can be implemented by solving the decision problem if two unit disks can be packed into P as discussed in Section 2.2.1 (see Theorem 2.2) and a simple test if $A - \frac{43}{c\varepsilon^2} \geq 0$. Computing k means adding one to A and determining the bit length of its representation. Since the bit-representation of A is polynomial in the input size, so is k. Hence, computing a and b means computing a polynomial number of bits of $\sqrt{\frac{A}{c}}$ and $\sqrt{12}$ which can be done in polynomial time with techniques from [14]. Line 9 can be implemented in time polynomial in the input size if the maximum number of unit disks that can be packed into P is bounded from above by a constant (N in Corollary 2.3). Since this step is only performed for $A \in \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$, this is the case for fixed ε. Therefore, Algorithm 5.5 runs in polynomial time for fixed ε. □

Lemmas 5.21 to 5.23 together directly imply the following theorem.

Theorem 5.24. *Algorithm 5.5 is a PTAS for determining the maximum number of unit disks that can be packed into a convex c-thick polygon for a fixed constant c given by its vertices.*

Observe that the theorem above also holds if the container is not a polygon as long as its area can be computed in polynomial time and it can be described with polynomially many inequalities and polynomially bounded degree in order to apply Corollary 2.3.

5.5 Remarks, Conclusion, and Open Problems for Chapters 3 to 5

Packing Unit Disks is in $\exists\mathbb{R}$

In Chapters 3 and 5, we used the approach discussed in Section 2.2.1 to pack unit disks, i.e., we solved first-order-sentences of the reals containing $\exists$-quantifiers and an $\forall$-quantifiers. However, the $\forall$-quantifier is not necessary for all problems discussed. For example, the following sentence is true if and only if n unit disks can be packed into a circle of radius r:

$$(\exists x_1, \dots, x_n, y_1, \dots, y_n) \bigwedge_{i=1}^{n} \left(x_i^2 + y_i^2 \leq (r-1)^2\right) \\ \wedge \bigwedge_{1 \leq i < j \leq n} \left((x_i - x_j)^2 + (y_i - y_j)^2 \geq 4\right).$$

In the formula above (x_i, y_i) is the coordinate of the center of the ith disk, the first part ensures that the disks lie inside the container circle (their centers have distance at most $r-1$ from the origin) and the second part ensures that no two disks intersect in their interior (their centers have distance at least 2). Hence, the problem if n unit disks can be packed into a circle with radius r lies in $\exists\mathbb{R}$. When packing unit disks into containers different from circles, only the first part of the formula has to be adapted. So, for all container shapes where this is possible, such as convex polygons, the corresponding problem is in $\exists\mathbb{R}$ and hence lies in PSPACE[15]. The same holds obviously for packing spheres in 3D.

Even though these formulas can be solved faster by the algorithm of Basu et al.[12], using these formulas instead would not change the asymptotic overall running time of our algorithms.

Reverse Problem

Considering the reverse problem, i.e., given n, what is the smallest scaling factor of a given container such that n copies of an object can be packed into it, one might think it can be solved with the same approach. Let $\phi(r)$ be the formula in (2.1) with c describing the container scaled by r. At first sight, the following formula seems to solve the problem:

$$\exists r(\phi(r) \land \forall r'(r' < r) \Rightarrow \neg\phi(r')).$$

However, this formula is either true or false (it is probably mostly true) and the algorithm by Basu et al.[12] does not return an actual value for r. Even worse, for packing unit disks into a circle, already for small n (3,4,5,8,...) is the optimal r not a rational number. Omitting the first quantifier in the formula above and using quantifier elimination (e.g.[12]) would return a semialgebraic set. Unfortunately, it is not obvious how to extract a good approximation for r from that set, this might have potential for further research.

Conclusion and further Open Questions

Table 5.1 summarizes the results of the previous chapters.

The main observation for the results in Chapters 3 and 5 was that the maximum number of unit disks (or other objects that can be described with first-order-formulas) that can be packed into a container described by a first-order-formula can be computed by solving first-order-formulas of the reals. The idea is then to solve the problem for small containers (depending on ε) and either compose an approximation for bigger containers from these small solutions (Chapter 3) or to approximate it by using the packing with maximum density in the plane (Chapter 5).

In Chapter 3, the container is a fat parallelogram or triangle whereas the objects can not only be unit disks but also copies of a fixed or thick object and we can even allow rigid motions and not just translations. The algorithm highly uses the shape of the container, so it is not clear if and how this can be adapted to more general container shapes. In contrast to this, in Chapter 5 the approach generalizes from circles as container to scaled

container / objects	fat parallelogram	fat triangle	gen. triangle	scaled fixed polygon	thick convex container	sphere	gen. simple polygon with holes
unit disks	PTAS	PTAS	const. factor	PTAS	PTAS		NP-hard [27]
fixed objects	PTAS	PTAS					NP-hard [27]
thick objects	PTAS	PTAS					NP-hard [27]
unit spheres						PTAS	

Table 5.1: Summarized results of Chapters 3 to 5

fixed polygons, thick containers and even packing spheres in 3D into a ball. However, since its analysis highly depends on the theorem given by Fejes Tóth (Theorem 2.1) for the packing of unit disks in the plane with highest density (and in 3D on the equivalent theorem by Hales et al.), it is open if and how this result can be used to pack more general objects. Observe, that in Chapter 3 and Chapter 5, the container is either fat or thick or a scaled copy of a fixed container. In Chapter 4 we made a small step towards more general containers by packing unit disks into general triangles, where, in contrast to Chapter 5, the triangle is neither fixed nor need to be fat. We achieved a constant factor approximation, it remains open if also a PTAS is possible as for more restricted containers.

Anyhow, these are the first approximability results for this kind of problems to our knowledge. Since not much is known about the complexity of these problems in general, this and the fact that they can be solved by solving first-order-formulas hopefully helps understanding and researching them in the future.

Chapter 6

A Note on Hales' "Dense Sphere Packing: A Blueprint to Formal Proofs"

6.1 Introduction

In [29] Hales proves that for every packing of infinitely many unit balls into three dimensional space, there exists a constant c such that the *density* inside a sphere of radius r is bounded from above by $\pi/\sqrt{18} + c/r$. Here, the density is defined as the volume of the intersection of the packed unit balls with the container sphere of radius r divided by the volume of the container sphere. The famous Kepler conjecture states that the density tends to $\pi/\sqrt{18}$ when r tends to infinity which is implied by the density bound shown by Hales.

To make, for example, statements about bounds for finite packings inside a container, we need c to be independent of the packing, i.e., we want a statement of the form: There exists a constant c' such that for every infinite sphere packing the density inside a ball of radius r is upper bounded by $\pi/\sqrt{18} + c'/r$. If it holds for an infinite packing, then this statement would also hold for a finite packing.

First, we give some definitions that are necessary to understand the crucial lemmas from [29]. Then, we follow the proofs of the lemmas and calculate a constant c' independent of the packing. Instead of using $\mathcal{O}$-Notation as in [29], we give more detailed calculations to be able to give an actual value for c'. When we cite lemmas or definitions from [29] (with occasional slight modifications) we give the corresponding number of the lemma or definition in [29] parenthesized. We try to give definitions as closely as possible to the point where we use them. This chapter is thought as an supplement to [29], therefore, we stick as closely as possible to the notation of Hales.

6.2 Main Lemmas

Let us denote by vol the Lebesgue measure on Euclidean space $\mathbb{R}^3$.

In the sequel, let $V \subset \mathbb{R}^3$ be a point set that induces a packing of infinitely many unit spheres, i.e., the points in V are the centers of the spheres and thus have pairwise distance at least 2. Let $\mathrm{B}(\mathbf{p}, r)$ be the open sphere centered at point $\mathbf{p}$ with radius r. Let $V(\mathbf{p}, r) = V \cap \mathrm{B}(\mathbf{p}, r)$. The *density* $\delta(V, \mathbf{p}, r)$ inside a container sphere centered at point $\mathbf{p}$ with radius r is defined by

$$\delta(V, \mathbf{p}, r) = \frac{\mathrm{vol}(\mathrm{B}(\mathbf{p}, r) \cap \bigcup_{\mathbf{v} \in V} \mathrm{B}(\mathbf{v}, 1))}{\mathrm{vol}(\mathrm{B}(\mathbf{p}, r))}.$$

In [29], it is shown that the face-centered cubic (FCC) packing of unit spheres has optimal density when r tends to infinity. In the FCC-packing, the spheres are arranged in layers. In each layer, the spheres are arranged with their centers on a hexagonal grid where neighboring grid points have distance 2. The vertices of the second layer lie above centers of the triangles in the first layer. In the third layer, the vertices lie above centers of triangles in the first and in the second layer. These three layers are then repeated infinitely often. The layers are pushed together until the spheres touch. See the following picture for illustration. Consider the Voronoi diagram of the circle centers

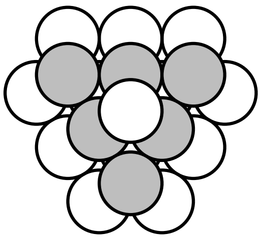


Figure 6.1: Three layers of the FCC-packing

in a FCC-packing. The Voronoi cells are rhombic dodecahedra with volume $4\sqrt{2}$ which yields a density of the packing of $\pi/\sqrt{18}$.

Let $\Omega(V, \mathbf{v})$ denote the cell of $\mathbf{v}$ in the Voronoi diagram of V and, more generally, let $\Omega(V, \underline{\mathbf{u}})$ be the intersection of the Voronoi cells for points in the list $\underline{\mathbf{u}} = [\mathbf{u}_0; \mathbf{u}_1, \ldots]$.

Definition 6.1 (Definition 6.11). *A function* $\mathrm{G}\colon V \to \mathbb{R}$ *on a set* $V \subset \mathbb{R}^3$ *is* negligible *if there is a constant* c_1 *such that for all* $r \geq 1$*,*

$$\sum_{\mathbf{v} \in \mathrm{B}(\mathbf{0}, r) \cap V} \mathrm{G}(\mathbf{v}) \leq c_1 r^2.$$

A function $\mathrm{G}\colon V \to \mathbb{R}$ *is* FCC-compatible *if for all* $\mathbf{v} \in V$

$$4\sqrt{2} \leq \mathrm{vol}(\Omega(V, \mathbf{v})) + \mathrm{G}(\mathbf{v}).$$

FCC-compatible means that the volume of every Voronoi cell is close to the volume of those in the FCC-packing (or larger). Then, *negligible* means, that the error is small.

Now we are ready to look at a lemma in [29] from which the Kepler conjecture follows under certain assumptions.

Lemma 6.2 (Lemma 6.13 from[29]). *If there exists a negligible FCC-compatible function* $\mathrm{G}\colon V \to \mathbb{R}$ *for a saturated packing* V*, then there exists a constant* $c = c(V)$ *such that for all* $r \geq 1$

$$\delta(V, \mathbf{0}, r) \leq \frac{\pi}{\sqrt{18}} + \frac{c}{r}.$$

Since this lemma is not sufficient to show the Kepler conjecture, two more lemmas are required. Therefore, we need two more definitions.

Definition 6.3 (Definition 6.34). *If* $\underline{\mathbf{u}} = [\mathbf{u}_0; \ldots; \mathbf{u}_k]$ *is a list of points in* $\mathbb{R}^n$*, then let* $\mathrm{h}(\underline{\mathbf{u}})$ *be the circumradius of its point set* $\{\mathbf{u}_0; \ldots; \mathbf{u}_k\}$*.*

Definition 6.4 (Definition 6.88). *Set*

$$h_0 = 1.26.$$

Let $\mathrm{L}\colon \mathbb{R} \to \mathbb{R}$ *be the piecewise linear function*

$$\mathrm{L}(h) = \begin{cases} \frac{h_0 - h}{h_0 - 1} & , h \leq h_0 \\ 0 & , h \geq h_0. \end{cases}$$

We call a packing *saturated* if no unit sphere can be added to the packing.

Lemma 6.5 (Lemma 6.95). *For any saturated packing* V *and any* $\mathbf{u}_0 \in V$*,*

$$\sum_{\mathbf{u}_1 \in V : \mathrm{h}([\mathbf{u}_0, \mathbf{u}_1]) \leq h_0} \mathrm{L}(\mathrm{h}([\mathbf{u}_0; \mathbf{u}_1])) \leq 12. \tag{6.1}$$

We will not discuss the proof of this lemma since it is irrelevant to calculate the constant c' explained in Section 6.1. It is a computer proof anyways.

Lemma 6.6 (Lemma 6.97). *Inequality (6.1) implies that for every saturated packing* V*, there exists a negligible FCC-compatible function* $\mathrm{G}\colon V \to \mathbb{R}$*.*

Lemmas 6.2, 6.5 and 6.6 together imply the Kepler conjecture. As mentioned above, we are interested in replacing the constant c in Lemma 6.2 by a constant c' that is independent of the packing V. In the proof of Lemma 6.2 in [29] it is shown that

$$\delta(V, \mathbf{0}, r) \leq \frac{\pi}{\sqrt{18}} \left(1 + \frac{3}{r}\right)^3 + c_1 \frac{(r+1)^2}{r^3 4\sqrt{2}}, \tag{6.2}$$

where c_1 is the constant from Definition 6.1 for the FCC-compatible negligible function $\mathrm{G}\colon V \to \mathbb{R}$ that exists by assumption. So, what we need to show in order to find c' is the following: There exists a constant d independent of the function G and therefore of the packing V such that we can set $c_1 = d$ in Definition 6.1 and Lemma 6.6 still holds.

In the following, we derive such a constant d, reproduce the proof of Lemma 6.6, and show that the constructed FCC-compatible function also fulfills the stronger definition of negligible for $c_1 = d$.

6.3 Proof of Lemma 6.6

For the following function, we will show that it is FCC-compatible and negligible:

$$\mathrm{G}(\mathbf{u}) = -\operatorname{vol}(\Omega(V,\mathbf{u})) + 8m_1 - \sum_{\mathbf{v}\in V\setminus\{\mathbf{u}\}} 8m_2\,\mathrm{L}(\mathrm{h}([\mathbf{u};\mathbf{v}])) \tag{6.3}$$

for constants m_1 and m_2 defined later and the function L as described in Definition 6.4.

To prove that this function is FCC-compatible and negligible, another lemma is used. For this lemma we need more definitions.

In the following, if we talk about cells, we mean so called *Marchal cells.* We refer the reader to Definition 6.51 in [29] since we will not need most of the definition for the calculations made here. Cells are defined by four points in V and a number $0 \le k \le 4$ and are denoted by $\operatorname{cell}(\underline{\mathbf{u}}, k)$ ("k-cell") where $\underline{\mathbf{u}}$ is a list of four points with some extra property to be explained later. 4-cells are always tetrahedra. The cells form a partition of $\mathbb{R}^3$. For a cell $X \neq \emptyset$, let $V(X) = X \cap V$ for a saturated packing V (Definition 6.62 and Lemma 6.63 in [29]).

The following three definitions build upon each other.

Definition 6.7 (Definition 3.7)**.** *A set C is* r-radial *at center* $\mathbf{v}$ *if the two conditions* $C \subset \mathrm{B}(\mathbf{v}, r)$ *and* $\mathbf{v}+\mathbf{u} \in C$ *imply* $\mathbf{v} + t\mathbf{u} \in C$ *for all t satisfying* $0 < \|\mathbf{u}\|t < r$*. A set C is* eventually radial *at center* $\mathbf{v}$ *if* $C \cap \mathrm{B}(\mathbf{v}, r)$ *is r-radial at center* $\mathbf{v}$ *for some* $r > 0$.

Definition 6.8 (Definition 3.11)**.** *When C is measurable and eventually radial at center* $\mathbf{v}$*, define the* solid angle *of C at* $\mathbf{v}$ *to be*

$$\operatorname{sol}(C,\mathbf{v}) = 3\frac{\operatorname{vol}(C \cap \mathrm{B}(\mathbf{v}, r))}{r^3},$$

where r is as in the definition of eventually radial. By Lemma 3.10 in [29], this yields the same value when replacing r by r' for any $0 \le r' \le r$*.*

Definition 6.9 (Definition 6.66)**.** *Define the* total solid angle *of a cell X to be*

$$\operatorname{tsol}(X) = \sum_{\mathbf{v}\in V(X)} \operatorname{sol}(X,\mathbf{v}).$$

We will use this definition later. The following seven definitions build upon each other.

Definition 6.10 (Definition 5.43). *Recall that a set $A \subset \mathbb{R}^n$ is* affine *if for every $\mathbf{v}, \mathbf{w} \in A$ and every $t \in \mathbb{R}$,*

$$t\mathbf{v} + (1-t)\mathbf{w} \in A.$$

Recall that the affine hull *of $P \subset \mathbb{R}^n$ (denoted $\mathrm{aff}(P)$) is the smallest affine set containing P. The* affine dimension *of P (written $\mathrm{dimaff}(P)$) is $\mathrm{card}(S) - 1$, where S is a set of smallest cardinality such that $P \subset \mathrm{aff}(S)$. In particular, the affine dimension of the empty set is -1. [...]*

Definition 6.11 (Definition 6.18). *Let V be a saturated packing. When $k = 0, 1, 2, 3$, let $\underline{V}(k)$ be the set of lists $\underline{\mathbf{u}} = [\mathbf{u}_0; \ldots; \mathbf{u}_k]$ of length $k+1$ with $\mathbf{u}_i \in V$ such that*

$$\mathrm{dimaff}(\Omega(V, [\mathbf{u}_0; \ldots; \mathbf{u}_j])) = 3 - j$$

for all $0 < j \le k$. Set $\underline{V}(k) = \emptyset$ for $k > 3$.

Definition 6.12 (Definition 6.24). *Let V be a saturated packing and let $\underline{\mathbf{u}} = [\mathbf{u}_0; \ldots; \mathbf{u}_k] \in \underline{V}(k)$ for some k. Define points $\omega_j = \omega_j(V, \underline{\mathbf{u}}) \in \mathbb{R}^3$ by recursion over $j \le k$*

$$\begin{aligned} \omega_0 &= \mathbf{u}_0 \\ \omega_{j+1} &= \text{ the closest point to } \omega_j \text{ on } \Omega(V, [\mathbf{u}_0; \ldots; \mathbf{u}_{j+1}]), \end{aligned}$$

set $w(V, \underline{\mathbf{u}}) = \omega_k(V, \underline{\mathbf{u}})$, when $\underline{\mathbf{u}} \in \underline{V}(k)$. The set V is generally fixed and is dropped from the notation.

Let $\mathrm{conv}\{\mathbf{v}_0, \ldots, \mathbf{v_n}\}$ denote the convex hull of the point set $\{\mathbf{v}_0, \ldots, \mathbf{v_n}\}$.

Definition 6.13 (Definition 6.51). *Let V be a saturated packing. Let $\underline{\mathbf{u}} = [\mathbf{u}_0; \ldots; \mathbf{u}_3] \in \underline{V}(3)$. Define $\xi(\underline{\mathbf{u}})$ as follows. If $\sqrt{2} \le \mathrm{h}([\mathbf{u}_0; \ldots; \mathbf{u}_2])$, then let $\xi(\underline{\mathbf{u}}) = \omega([\mathbf{u}_0; \ldots; \mathbf{u}_2])$. If $\mathrm{h}([\mathbf{u}_0; \ldots; \mathbf{u}_2]) < \sqrt{2} \le \mathrm{h}(\underline{\mathbf{u}})$, define $\xi(\underline{\mathbf{u}})$ to be the unique point in*

$$\mathrm{conv}\{\omega([\mathbf{u}_0; \ldots; \mathbf{u}_2]), \omega(\underline{\mathbf{u}})\}$$

at distance $\sqrt{2}$ from $\mathbf{u}_0$. [...]

Definition 6.14 (Definition 2.66). *When $\mathbf{v}_0 \ne \mathbf{v}_1$, write $\mathrm{dih}_V(\{\mathbf{v}_0, \mathbf{v}_1\}, \{\mathbf{v}_2, \mathbf{v}_3\})$ for the angle $\gamma \in [0, \pi]$ formed by*

$$\begin{aligned} \overline{\mathbf{w}}_2 &= (\mathbf{w}_1 \cdot \mathbf{w}_1)\mathbf{w}_2 - (\mathbf{w}_1 \cdot \mathbf{w}_2)\mathbf{w}_1 \text{ and} \\ \overline{\mathbf{w}}_3 &= (\mathbf{w}_1 \cdot \mathbf{w}_1)\mathbf{w}_3 - (\mathbf{w}_1 \cdot \mathbf{w}_3)\mathbf{w}_1, \end{aligned}$$

where $\mathbf{w}_i = \mathbf{v}_i - \mathbf{v}_0$.

Definition 6.15 (Definition 6.67). *Let $E(X)$ be the set of* extremal edges *of the k-cell X in a saturated packing V. More precisely, let*

$$E(X) = \{\{\mathbf{u}_i, \mathbf{u}_j\} : \mathbf{u}_i \ne \mathbf{u}_j \in V(X)\}.$$

In particular, $E(X)$ is empty for 0 and 1-cells and contains $\binom{k}{2}$ pairs when $2 \le k \le 4$.

Definition 6.16 (Definition 6.68). *Let V be a saturated packing. Let X be a k-cell, where $2 \leq k \leq 4$. Let $\varepsilon \in E(X)$. We define the dihedral angle* $\mathrm{dih}(X, \varepsilon)$ *of X along ε as follows. Explicitly, if X is a null set, then set* $\mathrm{dih}(X, \varepsilon) = 0$. *Otherwise, choose* $\underline{\mathbf{u}} = [\mathbf{u}_0; \mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_3] \in \underline{V}(3)$ *such that $X = cell(\underline{\mathbf{u}}, k)$ and $\varepsilon = \{\mathbf{u}_0, \mathbf{u}_1\}$. Set* $\mathrm{dih}(X, \varepsilon) = \mathrm{dih}_V(\{\mathbf{u}_0, \mathbf{u}_1\}, \{\mathbf{v}, \mathbf{w}\})$*, where*

$$\{\mathbf{v}, \mathbf{w}\} = \begin{cases} \{\xi(\underline{\mathbf{u}}), \omega(\underline{\mathbf{u}})\} & , k = 2 \\ \{\mathbf{u}_2, \xi(\underline{\mathbf{u}})\} & , k = 3 \\ \{\mathbf{u}_2, \mathbf{u}_3\} & , k = 4. \end{cases}$$

This is independent of the choice of $\underline{\mathbf{u}}$ defining X.

Now, we need just two more definitions before we can state the lemma mentioned above. The second one builds upon all the previously given definitions.

Definition 6.17 (Definition 6.70). *Define the following constants [...]:*

$$\begin{aligned} \mathrm{sol}_0 &= 3 \arccos\left(\frac{1}{3}\right) - \pi \\ \tau_0 &= 4\pi - 20\,\mathrm{sol}_0 \\ m_1 &= \mathrm{sol}_0\, 2\frac{\sqrt{2}}{\tau_0} \approx 1.012 \\ m_2 &= (6\,\mathrm{sol}_0 - \pi)\frac{\sqrt{2}}{6\tau_0} \approx 0.0254 \end{aligned}$$

[...]

Definition 6.18 (Definition 6.79). *For any cell X of a saturated packing, define the function $\gamma(X, *)$ on $\{f \colon \mathbb{R} \to \mathbb{R}\}$ by*

$$\gamma(X, f) = \mathrm{vol}(X) - \left(\frac{2m_1}{\pi}\right) \mathrm{tsol}(X) + \left(\frac{8m_2}{\pi}\right) \sum_{\varepsilon \in E(X)} \mathrm{dih}(X, \varepsilon) f(h(\varepsilon)).$$

Now, we can state the lemma.

Lemma 6.19 (Lemma 6.86). *Let f be any bounded, compactly supported function. Set*

$$\mathrm{G}(\mathbf{u}_0, f) = -\,\mathrm{vol}(\Omega(V, \mathbf{u}_0)) + 8m_1 - \sum_{\mathbf{u} \in V \setminus \{\mathbf{u}_0\}} 8m_2 f(\mathrm{h}([\mathbf{u}_0; \mathbf{u}])).$$

If

$$\sum_{\mathbf{v} \in V \setminus \{\mathbf{u}\}} f(\mathrm{h}([\mathbf{u}; \mathbf{v}])) \leq 12,$$

then $\mathrm{G}(*, f)$ *is FCC-compatible. Moreover, if there exists a constant* c_0 *such that for all* $r \geq 1$

$$\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X, f) \geq c_0 r^2,$$

then $\mathrm{G}(*, f)$ *is negligible.* [1]

This Lemma almost implies that there is a FCC-compatible negligible function. Using this Lemma, it is sufficient to show that Lemma 6.5 implies the existence of suitable G and f in Lemma 6.19 to prove Lemma 6.6. We will turn to this later. Now, we focus on the proof of Lemma 6.19. Since we are only interested in the constant in the definition of negligible, we will only outline this part of the proof here. The idea of the proof of negligibility is as follows. If one can show that

$$-\sum_{\mathbf{u} \in V(\mathbf{0},r)} \mathrm{G}(\mathbf{u}, f) \geq \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X, f) + c_2 r^2 \tag{6.4}$$

for some constant c_2. Then by the assumption in Lemma 6.19

$$-\sum_{\mathbf{u} \in V(\mathbf{0},r)} \mathrm{G}(\mathbf{u}, f) \geq (c_0 + c_2) r^2, \tag{6.5}$$

which directly implies, that $\mathrm{G}(*, f)$ is negligible for $c_1 = -(c_0 + c_2)$ in Definition 6.1. Since we are only interested in the independence of c_1 of the packing V, we will now focus on calculating the constant c_2 and thereby showing that c_2 is independent of the packing. We will get an estimate on c_0 later satisfying the conditions in Lemma 6.19.

6.3.1 Calculation of c_2

In this section we go through the proof of Lemma 6.19 and show that

$$-\sum_{\mathbf{u} \in V(\mathbf{0},r)} \mathrm{G}(\mathbf{u}, f) \geq \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X, f) + c_2 r^2$$

for some constant c_2 independent of the packing V.

Observe that the equalities

$$-\sum_{\mathbf{u} \in V(\mathbf{0},r)} \mathrm{G}(\mathbf{u}, f) =$$
$$\sum_{\mathbf{u} \in V(\mathbf{0},r)} \mathrm{vol}(\Omega(V, \mathbf{u})) - \sum_{\mathbf{u} \in V(\mathbf{0},r)} 8m_1 + \sum_{\mathbf{u} \in V(\mathbf{0},r)} \sum_{\mathbf{v} \in V \setminus \{\mathbf{u}\}} 8m_2 f(\mathrm{h}([\mathbf{u}; \mathbf{v}]))$$

[1]In the sequel, by slight abuse of notation $\sum_{X \subset \mathrm{B}(\mathbf{0},r)}$ refers to the summation only over all Marchal cells $X \subset \mathrm{B}(\mathbf{0}, r)$.

and

$$\sum_{X\subset \mathrm{B}(\mathbf{0},r)} \gamma(X,f) =$$

$$\sum_{X\subset \mathrm{B}(\mathbf{0},r)} \mathrm{vol}(X) - \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \left(\frac{2m_1}{\pi}\right) \mathrm{tsol}(X) + \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \left(\frac{8m_2}{\pi}\right) \sum_{\varepsilon\in E(X)} \mathrm{dih}(X,\varepsilon) f(h(\varepsilon))$$

have each three summands and we will relate them in this order. So, we start by showing that

$$\sum_{\mathbf{u}\in V(\mathbf{0},r)} \mathrm{vol}(\Omega(V,\mathbf{u})) \geq \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \mathrm{vol}(X) - \frac{56}{3}\pi r^2. \tag{6.6}$$

By Lemma 6.7 in [29], $\Omega(V,\mathbf{v}) \subset \mathrm{B}(\mathbf{v},2)$, so we have

$$\begin{aligned}\sum_{\mathbf{u}\in V(\mathbf{0},r)} \mathrm{vol}(\Omega(V,\mathbf{u})) &\geq \mathrm{vol}(\mathrm{B}(\mathbf{0},r-2)) \\ &= \mathrm{vol}(\mathrm{B}(\mathbf{0},r)) - \mathrm{vol}(\mathrm{B}(\mathbf{0},r)\setminus \mathrm{B}(\mathbf{0},r-2)) \\ &\geq \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \mathrm{vol}(X) - \left(\frac{4}{3}\pi r^3 - \frac{4}{3}\pi (r-2)^3\right),\end{aligned}$$

since Marchal cells form a partition of space as mentioned above. So we get

$$\sum_{\mathbf{u}\in V(\mathbf{0},r)} \mathrm{vol}(\Omega(V,\mathbf{u})) \geq \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \mathrm{vol}(X) - 8\pi r^2 - \frac{32}{3}\pi.$$

Since negligible is only defined for $r \geq 1$, we can assume $r \geq 1$ here. So we have

$$\sum_{\mathbf{u}\in V(\mathbf{0},r)} \mathrm{vol}(\Omega(V,\mathbf{u})) \geq \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \mathrm{vol}(X) - \frac{56}{3}\pi r^2,$$

which proves Eq. (6.6).

Next, we will show that

$$-\sum_{\mathbf{u}\in V(\mathbf{0},r)} 8m_1 \geq -\left(\frac{2m_1}{\pi}\right) \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \mathrm{tsol}(X) - m_1\cdot 2240 r^2. \tag{6.7}$$

First, we need to estimate the diameter of a Marchal cell. For a cell $X = \mathrm{cell}(\underline{\mathbf{u}},k)$ for $\underline{\mathbf{u}} = [\mathbf{u}_0;\dots] \in \underline{V}(3)$ it holds that

$$X = \mathrm{cell}(\underline{\mathbf{u}},k) \subset \Omega(V,\mathbf{u}_0) \cup \dots \cup \Omega(V,\mathbf{u}_{k-1})$$

(see the proof of Lemma 6.63 in [29]). By Lemma 6.7, $\Omega(V,\mathbf{v}) \subset \mathrm{B}(\mathbf{v},2)$, so we get

$$X \subset \mathrm{B}(\mathbf{u}_0,2) \cup \dots \cup \mathrm{B}(\mathbf{u}_{k-1},2).$$

By Definition 6.11, the Voronoi cells $\Omega(V, \mathbf{u}_0), \ldots, \Omega(V, \mathbf{u}_{k-1})$ share at least one point and so the spheres $\mathrm{B}(\mathbf{u}_0, 2), \ldots, \mathrm{B}(\mathbf{u}_{k-1}, 2)$ do. Therefore, we can conclude that the diameter of a Marchal cell is at most 4.

Now,

$$\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \mathrm{tsol}(X) = \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \sum_{\mathbf{v} \in V(X)} \mathrm{sol}(X, \mathbf{v})$$

by definition. Since the diameter of each cell is at most 4, we can rewrite this as

$$\begin{aligned}\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \mathrm{tsol}(X) &= \sum_{X \subset \mathrm{B}(\mathbf{0},r+4)} \sum_{\mathbf{u} \in V(X)} \mathrm{sol}(X, \mathbf{u}) - \sum_{X \subset \mathrm{B}(\mathbf{0},r+4) : X \not\subseteq \mathrm{B}(\mathbf{0},r)} \sum_{\mathbf{u} \in V(X)} \mathrm{sol}(X, \mathbf{u}) \\ &\geq \sum_{\mathbf{v} \in V(\mathbf{0},r)} \sum_{X : \mathbf{v} \in V(X)} \mathrm{sol}(X, \mathbf{v}) - \sum_{\mathbf{u} \in V(\mathbf{0},r+4) \setminus V(\mathbf{0},r-4)} \sum_{X : \mathbf{u} \in V(X)} \mathrm{sol}(X, \mathbf{u}),\end{aligned}$$

since $V(X) = X \cap V$ and a cell with diameter at most 4 that is not completely contained in a sphere of radius r cannot intersect the sphere with the same center of radius $r-4$. Next, we use that the solid angles around one point sum up to 4π. Furthermore, to estimate the number of points $\mathbf{u} \in V(\mathbf{0}, r+4) \setminus V(\mathbf{0}, r-4)$, we use the following volume argument. Each $\mathbf{u} \in V(\mathbf{0}, r+4) \setminus V(\mathbf{0}, r-4)$ is the center of a packed unit sphere and therefore the volume of those unit spheres sum up to less than the volume of $\mathrm{B}(\mathbf{0}, r+5) \setminus \mathrm{B}(\mathbf{0}, r-5)$. Vice versa dividing $\mathrm{vol}(\mathrm{B}(\mathbf{0}, r+5) \setminus \mathrm{B}(\mathbf{0}, r-5))$ by the volume of a unit sphere yields an upper bound on the number of points, so we get

$$\begin{aligned}\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \mathrm{tsol}(X) &\geq \sum_{\mathbf{v} \in V(\mathbf{0},r)} 4\pi - \left(\frac{\frac{4}{3}\pi(r+5)^3 - \frac{4}{3}\pi(r-5)^3}{\frac{4}{3}\pi} \right) 4\pi \\ &= \sum_{\mathbf{v} \in V(\mathbf{0},r)} 4\pi - (30r^2 + 250) 4\pi \\ &\geq \sum_{\mathbf{v} \in V(\mathbf{0},r)} 4\pi - 1120\pi r^2,\end{aligned}$$

since $r \geq 1$.

By rearranging, we obtain

$$-\left(\frac{2m_1}{\pi}\right) \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \mathrm{tsol}(X) - 2240 m_1 r^2 \leq -\sum_{\mathbf{v} \in V(\mathbf{0},r)} 8m_1,$$

proving Eq. (6.7).

Finally, we show that

$$\sum_{\mathbf{u} \in V(\mathbf{0},r)} \sum_{\mathbf{v} \in V \setminus \{\mathbf{u}\}} 8m_2 f(\mathrm{h}([\mathbf{u}; \mathbf{v}])) \geq \frac{8m_2}{\pi} \sum_{X \in \mathrm{B}(\mathbf{0},r)} \sum_{\varepsilon \in E(X)} \mathrm{dih}(X, \varepsilon) f(\mathrm{h}(\varepsilon)). \tag{6.8}$$

For any $\varepsilon \in E(X)$ it holds that $\varepsilon \subset V(X) \subset X$ by Definition 6.15, and Lemma 6.63 in [29]. This implies

$$\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \sum_{\varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon) f(\mathrm{h}(\varepsilon)) \leq \sum_{\varepsilon=\{\mathbf{u},\mathbf{v}\} \subset \mathrm{B}(\mathbf{0},r)} \sum_{X : \varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon) f(\mathrm{h}(\varepsilon))$$
$$= \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r)} 2\pi f(\mathrm{h}(\varepsilon)),$$

since the dihedral angle around an edge sums up to 2π.[2]When summing over ordered pairs, each edge appears twice, so we have to divide by two and get

$$\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \sum_{\varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon) f(\mathrm{h}(\varepsilon)) \leq \sum_{\mathbf{u} \in V(\mathbf{0},r)} \sum_{\mathbf{v} \in V(\mathbf{0},r) \setminus \{\mathbf{u}\}} \pi f(\mathrm{h}(\mathbf{u},\mathbf{v})).$$

So, for the last summand we get by rearrangement

$$\frac{8m_2}{\pi} \sum_{X \in \mathrm{B}(\mathbf{0},r)} \sum_{\varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon) f(\mathrm{h}(\varepsilon)) \leq \sum_{\mathbf{u} \in V(\mathbf{0},r)} \sum_{\mathbf{v} \in V(\mathbf{0},r) \setminus \{\mathbf{u}\}} 8m_2 f(\mathrm{h}(\mathbf{u},\mathbf{v})),$$

showing Eq. (6.8).

Combining Eqs. (6.6) to (6.8), we have

$$\begin{aligned}
&-\mathrm{G}(\mathbf{u},f)\\
&= \sum_{\mathbf{u} \in V(\mathbf{0},r)} \mathrm{vol}(\Omega(V,\mathbf{u})) - \sum_{\mathbf{u} \in V(\mathbf{0},r)} 8m_1 + \sum_{\mathbf{u} \in V(\mathbf{0},r)} \sum_{\mathbf{v} \in V \setminus \{\mathbf{u}\}} 8m_2 f(\mathrm{h}([\mathbf{u};\mathbf{v}]))\\
&\geq \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \mathrm{vol}(X) - \frac{56}{3}\pi r^2 - \left(\frac{2m_1}{\pi}\right) \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \mathrm{tsol}(X) - m_1 \cdot 2240 r^2 +\\
&\frac{8m_2}{\pi} \sum_{X \in \mathrm{B}(\mathbf{0},r)} \sum_{\varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon) f(\mathrm{h}(\varepsilon))\\
&= \sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X,f) - \left(\frac{56}{3} + m_1 \cdot 2240\right) r^2,
\end{aligned}$$

i.e. Eq. (6.4) holds with $c_2 = -\frac{56}{3} - m_1 \cdot 2240$.

As mentioned above, we only need to show that the conditions in Lemma 6.19 hold for some f to prove Lemma 6.6. The first condition holds for $f = \mathrm{L}$ by Lemma 6.5. It remains to show that the second condition also holds for $f = \mathrm{L}$, namely $\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X,\mathrm{L}) \geq c_0 r^2$. Then, we will have showed that the function $\mathrm{G}(*,\mathrm{L})$ is FCC-compatible and negligible for $c_1 = -(c_0 + c_2)$ in Definition 6.1. For c_2 we already showed that it is independent of the packing, now, we will do so for c_0, i.e. prove that $\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X,\mathrm{L}) \geq c_0 r^2$ holds for a c_0 independent of the packing.

[2]Here, we follow the argumentation of Hales [29]. The author noticed that in Lemma 6.69 in [29], it is required that $\mathrm{h}(\varepsilon) < \sqrt{2}$ to have that the dihedral angles sum up to 2π. Since Lemma 6.19 is only used for $f = \mathrm{L}$ and $\mathrm{L}(\mathrm{h}(\varepsilon)) = 0$ for $\mathrm{h}(\varepsilon) \geq \sqrt{2}$, this does not cause any problems.

6.3.2 Calculation of c_0

In this section, we will show that

$$\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X, f) \geq c_0 r^2 \tag{6.9}$$

holds for $f = \mathrm{L}$ for a constant c_0 independent of the packing that defines the cells X. We will need more definitions for this proof and we will try to give them as closely as possible to the point where they are used.

Definition 6.20 (Definition 6.70 and 6.88)**.** *Set*

$$h_+ = 1.3254.$$

Let $\mathrm{M}\colon \mathbb{R} \to \mathbb{R}$ *be the piecewise polynomial function*

$$\mathrm{M}(h) = \begin{cases} \frac{\sqrt{2}-h}{\sqrt{2}-1} \frac{h_+ - h}{h_+ - 1} \frac{17h - 9h^2 - 3}{5} & , h \leq \sqrt{2} \\ 0 & , h > \sqrt{2}. \end{cases}$$

Let $h_- \approx 1.23175$ *be the unique root of the quadratic polynomial* $\mathrm{M}(h) - \mathrm{L}(h)$ *that lies in the interval* $[1.231, 1.232]$.

Definition 6.21 (Definition 6.89)**.** *A* critical edge ε *of a saturated packing* V *is an unordered pair that appears as an element of* $E(X)$ *for some* k*-cell* X *of the packing* V *such that* $h(\varepsilon) \in [h_-, h_+]$*. Let* $\mathrm{EC}(X)$ *be the set of critical edges that belong to* $E(X)$*. If* X *is any cell such that* $\mathrm{EC}(X)$ *is not empty, let the* weight $\mathrm{wt}(X)$ *of* X *be* $1/\operatorname{card}(\mathrm{EC}(X))$.

Now, we can start with estimating $\sum_{X\subset \mathrm{B}(\mathbf{0},r)} \gamma(X,\mathrm{L})$.

$$\begin{aligned}
\sum_{X\subset \mathrm{B}(\mathbf{0},r)} \gamma(X,\mathrm{L}) &= \sum_{X\subset \mathrm{B}(\mathbf{0},r):\mathrm{EC}(X)\neq\emptyset} \gamma(X,\mathrm{L}) + \underbrace{\sum_{X\subset \mathrm{B}(\mathbf{0},r):\mathrm{EC}(X)=\emptyset} \gamma(X,\mathrm{L})}_{\geq 0 \text{ by Lemma 6.92 in [29]}} \\
&\geq \sum_{X\subset \mathrm{B}(\mathbf{0},r):\mathrm{EC}(X)\neq\emptyset} \gamma(X,\mathrm{L}) \\
&= \sum_{X\subset \mathrm{B}(\mathbf{0},r):\mathrm{EC}(X)\neq\emptyset} \left(\gamma(X,\mathrm{L}) \underbrace{\sum_{\varepsilon\in\mathrm{EC}(X)} \mathrm{wt}(X)}_{=1 \text{ by Definition 6.21}} \right) \\
&= \sum_{X\subset \mathrm{B}(\mathbf{0},r):\mathrm{EC}(X)\neq\emptyset} \sum_{\varepsilon\in\mathrm{EC}(X)} \gamma(X,\mathrm{L})\,\mathrm{wt}(X) \\
&= \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X:\varepsilon\in\mathrm{EC}(X)} \gamma(X,\mathrm{L})\,\mathrm{wt}(X) \\
&\quad - \underbrace{\sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X:\varepsilon\in\mathrm{EC}(X), X\not\subseteq \mathrm{B}(\mathbf{0},r)} \gamma(X,\mathrm{L})\,\mathrm{wt}(X)}_{=:\alpha} \qquad (6.10)
\end{aligned}$$

Next, we will show that the term α is bounded from above by a constant times r^2.

$$\begin{aligned}
\alpha &= \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X:\varepsilon\in\mathrm{EC}(X), X\not\subseteq \mathrm{B}(\mathbf{0},r)} \gamma(X,\mathrm{L})\,\mathrm{wt}(X) \\
&\leq \sum_{X\subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} \sum_{\varepsilon\in\mathrm{EC}(X)} \gamma(X,\mathrm{L})\,\mathrm{wt}(X), \qquad (6.11)
\end{aligned}$$

since the diameter of X is at most 4 and X intersects the boundary of $\mathrm{B}(\mathbf{0},r)$. By Eq. (6.11),

$$\alpha \leq \sum_{X\subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} \left(\gamma(X,\mathrm{L}) \underbrace{\sum_{\varepsilon\in\mathrm{EC}(X)} \mathrm{wt}(X)}_{=1 \text{ by Definition 6.21}} \right). \qquad (6.12)$$

Next, we plug in the definition of $\gamma(X, L)$. By Eq. (6.12),

$$\begin{aligned}\alpha &\leq \sum_{X \subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} \left(\mathrm{vol}(X) - \underbrace{\left(\frac{2m_1}{\pi}\right)\mathrm{tsol}(X)}_{\geq 0 \text{ by definition}} + \left(\frac{8m_2}{\pi}\right) \sum_{\varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon)\,\mathrm{L}(h(\varepsilon)) \right) \\ &\leq \sum_{X \subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} \mathrm{vol}(X) + \sum_{X \in \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} \left(\frac{8m_2}{\pi}\right) \sum_{\varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon)\,\mathrm{L}(h(\varepsilon)).\end{aligned} \tag{6.13}$$

Since Marchal cells are a partition of the space, the volume of the cells contained in a spherical shell sums up to at most the volume of the spherical shell. Furthermore, it holds for any $\varepsilon \in E(X)$ that $\varepsilon \subset V(X) \subset X$, so we get from Eq. (6.13)

$$\alpha \leq \frac{4}{3}\pi(r+4)^3 - \frac{4}{3}\pi(r-4)^3 + \frac{8m_2}{\pi} \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} \sum_{X : \varepsilon \in E(X)} \mathrm{dih}(X,\varepsilon)\,\mathrm{L}(h(\varepsilon)). \tag{6.14}$$

Since the dihedral angle for edges ε with $\mathrm{h}(\varepsilon) < \sqrt{2}$ sums up to 2π (see Lemma 6.69 in [29]) and for $\mathrm{h}(\varepsilon) \geq \sqrt{2}$ the factor $\mathrm{L}(\mathrm{h}(\varepsilon)) = 0$, we obtain from Eq. (6.14)

$$\alpha \leq 32\pi r^2 + \frac{512}{3}\pi + \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4)} 16 m_2\, \mathrm{L}(h(\varepsilon)). \tag{6.15}$$

Now, we want to estimate the number of edges such that $\mathrm{L}(\mathrm{h}(\varepsilon)) > 0$. First, we give an upper bound on the number of points in V inside the spherical shell as follows. By decreasing the inner radius and increasing the outer radius by 1, all unit spheres packed with centers in the original shell are completely contained in the enlarged shell. Then, we divide the volume of the enlarged shell by the volume of a unit sphere. Next, we want to give an upper bound on the number of points in V with distance at most 2.52 from a given point, since for longer edges $\mathrm{L}(\mathrm{h}(\varepsilon)) = 0$. We do this in a similar way as before by a volume argument. By multiplying these two values, we count each edge twice, so we have to divide by 2. In addition, the function L is upper bounded by 1.26/0.26. So, by Eq. (6.15)

$$\begin{aligned}\alpha &\leq 32\pi r^2 + \frac{512}{3}\pi + \underbrace{\frac{\frac{4}{3}\pi(r+5)^3 - \frac{4}{3}\pi(r-5)^3}{\frac{4}{3}\pi}}_{\geq |V \cap (\mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4))|} \cdot \underbrace{\frac{\frac{4}{3}\pi \cdot 3.52^3}{\frac{4}{3}\pi}}_{\geq |V \cap \mathrm{B}(\mathbf{v},2.52)| \text{ for any } \mathbf{v}} \cdot \frac{1}{2} \cdot 16 m_2 \cdot \frac{126}{26} \\ &\leq 32\pi r^2 + \frac{512}{3}\pi + \left(30r^2 + 250\right) \cdot 3.52^3 \cdot 8 \cdot \frac{63}{13} \cdot 0.0255 \\ &\leq 1394.1 r^2 + 11315.6.\end{aligned}$$

As mentioned before, we can assume $r \geq 1$ and obtain

$$\alpha \leq 12710 r^2. \tag{6.16}$$

Now, we know from Eqs. (6.10) and (6.16) that

$$\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X,L) \geq \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r)} \sum_{X : \varepsilon \in \mathrm{EC}(X)} \gamma(X,L)\,\mathrm{wt}(X) - 12710^2. \tag{6.17}$$

To proceed with the estimation, we need two more definitions.

Definition 6.22 (Definition 6.90)**.** *Set*

$$\beta_0(h) = 0.005\left(1 - \frac{(h-h_0)^2}{(h_+ - h_0)^2}\right).$$

If X is a 4-cell with exactly two critical edges and if those edges are opposite, then set

$$\beta(\varepsilon, X) = \beta_0(\mathrm{h}(\varepsilon)) - \beta_0(\mathrm{h}(\varepsilon')), \text{ where } \mathrm{EC}(X) = \{\varepsilon, \varepsilon'\}.$$

Otherwise, for all other edges in all other cells, set $\beta(\varepsilon, X) = 0$.

Definition 6.23 (Definition 6.91)**.** *Let V be a saturated packing. Let $\varepsilon \in \mathrm{EC}(X)$ be a critical edge of a k-cell X of V for some $2 \leq k \leq 4$. A* cell cluster *is the set*

$$\mathrm{CL}(\varepsilon) = \{X : \varepsilon \in \mathrm{EC}(X)\}$$

of all cells around ε. Define

$$\Gamma(\varepsilon) = \sum_{X \in \mathrm{CL}(\varepsilon)} \gamma(X,L)\,\mathrm{wt}(X) + \beta(\varepsilon, X).$$

Using these definitions, we can rewrite Eq. (6.17) as follows:

$$\begin{aligned}\sum_{X \subset \mathrm{B}(\mathbf{0},r)} \gamma(X,L) &\geq \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r)} \sum_{X : \varepsilon \in \mathrm{EC}(X)} \gamma(X,L)\,\mathrm{wt}(X) - 12710^2 \\ &= \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r)} \left(\Gamma(\varepsilon) - \sum_{X \in \mathrm{CL}(\varepsilon)} \beta(\varepsilon, X)\right) - 12710 r^2 \\ &= \sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r)} \Gamma(\varepsilon) - \underbrace{\sum_{\varepsilon \subset \mathrm{B}(\mathbf{0},r)} \sum_{X \in \mathrm{CL}(\varepsilon)} \beta(\varepsilon, X)}_{=:\zeta} - 12710 r^2.\end{aligned} \tag{6.18}$$

Next, we will upper bound the term ζ.

$$\begin{aligned}
\zeta &= \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X\in \mathrm{CL}(\varepsilon)} \beta(\varepsilon,X) \\
&= \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X\in \mathrm{CL}(\varepsilon):X\subset \mathrm{B}(\mathbf{0},r)} \beta(\varepsilon,X) + \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X\in \mathrm{CL}(\varepsilon):X\not\subseteq \mathrm{B}(\mathbf{0},r)} \beta(\varepsilon,X) \\
&= \sum_{X\subset \mathrm{B}(\mathbf{0},r)} \sum_{\varepsilon\in \mathrm{EC}(X)} \beta(\varepsilon,X) + \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{X\in \mathrm{CL}(\varepsilon):X\not\subseteq \mathrm{B}(\mathbf{0},r)} \beta(\varepsilon,X) \\
&= 0 + \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{\substack{X\in \mathrm{CL}(\varepsilon):X\not\subseteq \mathrm{B}(\mathbf{0},r)\\ \wedge X \text{ is 4-cell with} |\mathrm{EC}(X)|=2}} \beta(\varepsilon,X),
\end{aligned} \tag{6.19}$$

since $\beta(\varepsilon,X) > 0$ only for 4-cells with exactly two critical edges ε,ε' and $\beta(\varepsilon,X) + \beta(\varepsilon',X) = 0$. Furthermore, $\beta(\varepsilon,X) \le 0.005$, so we get from Eq. (6.19)

$$\zeta \le \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \sum_{\substack{X\in \mathrm{CL}(\varepsilon):X\not\subseteq \mathrm{B}(\mathbf{0},r)\\ \wedge X \text{ is 4-cell with } |\mathrm{EC}(X)|=2}} 0.005. \tag{6.20}$$

Next, we want to exchange the inner and outer sum. Since the inner sum is zero for non-critical edges and the cells intersect the boundary of $\mathrm{B}(\mathbf{0},r)$, we get from Eq. (6.20)

$$\begin{aligned}
\zeta &\le \sum_{\substack{X\subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4):\\ X \text{ is a 4-cell with } |\mathrm{EC}(X)|=2}} \sum_{\varepsilon\in \mathrm{EC}(X)} 0.005 \\
&\le \sum_{\substack{X\subset \mathrm{B}(\mathbf{0},r+4)\setminus \mathrm{B}(\mathbf{0},r-4):\\ X \text{ is a 4-cell}}} 0.01.
\end{aligned} \tag{6.21}$$

Now, we will estimate the number of points in V inside the sperical shell as before. A 4-cell is the convex hull of four points in V with circumradius at most $\sqrt{2}$ (see Definition 6.51 in [29]). Therefore, for a fixed point $\mathbf{v} \in V$ all points that can form a 4-cell with $\mathbf{v}$ must lie inside a ball of radius $2\sqrt{2}$ centered at $\mathbf{v}$. As before, we calculate an upper bound on the number of points in V inside a ball of radius $2\sqrt{2}$. Then, the number of subsets of size three that can be formed with these points is an upper bound on the number of 4-cells a point in V can be part of. By multiplying these numbers, we count each 4-cell 4 times, so we divide by 4. So, we get from Eq. (6.21)

$$\begin{aligned}
\zeta &\le \frac{\frac{4}{3}\pi(r+5)^3 - \frac{4}{3}\pi(r-5)^3}{\frac{4}{3}\pi} \cdot \binom{\left\lfloor \frac{\frac{4}{3}\pi(2\sqrt{2}+1)^3}{\frac{4}{3}\pi} \right\rfloor}{3} \cdot \frac{1}{4} \cdot 0.01 \\
&= \frac{(30r^2+250)}{4} \cdot \binom{\left\lfloor \left(2\sqrt{2}+1\right)^3 \right\rfloor}{3} \cdot 0.01 \\
&= \left(7.5r^2 + 62.5\right) \cdot 27720 \cdot 0.01 \\
&= 2079r^2 + 17325.
\end{aligned}$$

Again, we can assume $r \geq 1$, so

$$\zeta \leq 19404r^2. \tag{6.22}$$

Now, we plug this into Eq. (6.18) and obtain

$$\begin{aligned}\sum_{X\subset \mathrm{B}(\mathbf{0},r)} \gamma(X,L) &\geq \sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \Gamma(\varepsilon) - 19404r^2 - 12710r^2 \\ &= \underbrace{\sum_{\varepsilon\subset \mathrm{B}(\mathbf{0},r)} \Gamma(\varepsilon)}_{\geq 0 \text{ by Theorem 6.93 in [29]}} \quad -32114r^2 \\ &\geq -32114r^2,\end{aligned}$$

i.e. Eq. (6.9) holds for $c_0 = -32114$.

We showed that both conditions in Lemma 6.19 hold and therefore the function $\mathrm{G}(*,\mathrm{L})$ is FCC-compatible and negligible (see Eq. (6.5) and the explanation thereafter) for

$$\begin{aligned} c_1 = -(c_0 + c_2) &= \frac{56}{3} + m_1 \cdot 2240 + 32114 \\ &\leq \frac{56}{3} + 1.013 \cdot 2240 + 32114 \\ &\leq 34402.\end{aligned}$$

By Eq. (6.2), the constant in Lemma 6.2 only depends on constants and c_1. So it is independent of the packing since we showed that c_1 is. Therefore, we turn now to Eq. (6.2) and will later plug in c_1 as calculated above:

$$\begin{aligned}\delta(V,\mathbf{0},r) &\leq \frac{\pi}{\sqrt{18}}\left(1+\frac{3}{r}\right)^3 + c_1\frac{(r+1)^2}{r^3 4\sqrt{2}} \\ &= \frac{\pi}{\sqrt{18}}\left(1+\frac{9}{r}+\frac{27}{r^2}+\frac{27}{r^3}\right) + c_1\frac{r^2+2r+1}{r^3 4\sqrt{2}} \\ &= \frac{\pi}{\sqrt{18}} + \frac{\pi}{\sqrt{18}}\left(\frac{9}{r}+\frac{27}{r^2}+\frac{27}{r^3}\right) + c_1\left(\frac{1}{r4\sqrt{2}}+\frac{2}{r^2 4\sqrt{2}}+\frac{1}{r^3 4\sqrt{2}}\right).\end{aligned}$$

Since $r \geq 1$, we have $\frac{1}{r^3} \leq \frac{1}{r^2} \leq \frac{1}{r}$ and get

$$\begin{aligned}\delta(V,\mathbf{0},r) &\leq \frac{\pi}{\sqrt{18}} + \frac{63\pi}{\sqrt{18}r} + \frac{c_1}{\sqrt{2}r} \\ &= \frac{\pi}{\sqrt{18}} + \frac{21\pi + c_1}{\sqrt{2}r} \\ &= \frac{\pi}{\sqrt{18}} + \frac{21\pi + 34402}{\sqrt{2}} \cdot \frac{1}{r}.\end{aligned}$$

Summarizing, we showed that the constant in Lemma 6.2 does not depend on the particular packing V but only on the constant for the assumed existing FCC-compatible negligible function. Then, we showed that there is a FCC-compatible negligible function for which the definition for negligible holds for a constant independent of the packing. So, we can state the main result of this work as follows.

Theorem 6.24. *For a saturated packing V and all $r \geq 1$ it holds that*

$$\delta(V, \mathbf{0}, r) \leq \frac{\pi}{\sqrt{18}} + \frac{21\pi + 34402}{\sqrt{2}} \cdot \frac{1}{r} \leq \frac{\pi}{\sqrt{18}} + 24373 \cdot \frac{1}{r}.$$

Chapter 7

Packing 2D-Disks into a 3D-Container

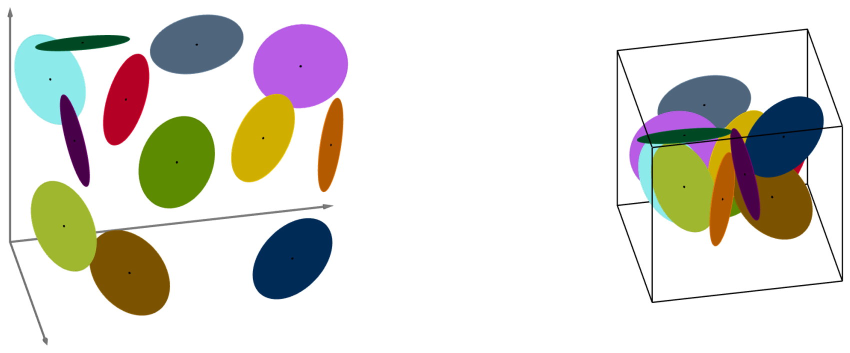
In three dimensions, most previous results on packing problems are concerned with packing "regular" objects like axis-parallel boxes, see e.g. [36]. In particular, approximation algorithms for packing rectangular cuboids or convex polyhedra into minimum volume rectangular cuboids or convex containers under rigid motions are known [9]. Whether this is possible under only translations remains open.

In this chapter, we give a positive answer for a restricted set of possible objects, namely disks of unit radius and axis-parallel box containers (see Fig. 7.1) by describing a constant-factor approximation algorithm. Currently, our approximation factor is forbiddingly high but it should be of theoretical interest that the problem, which is neither known to be NP-hard nor if its corresponding decision problem is in NP, can be approximated in polynomial time at all.

Overview

We say that two disks are *identical* (modulo translation) if their normal vectors are multiples of each other. Note that the normal vectors do not need to be normalized, i.e., they can have a length different from 1. Anyhow, they define a plane (the plane to which they are perpendicular) and therefore a disk. We say that two disks are in a position where they *touch* if their intersection contains only one point and that two disks *intersect* if their intersection consists of more than one point. By *nonoverlapping*, we mean a placement of disks such that no two disks intersect whereas it is allowed that two disks touch. The main problem we study in this work is then defined as follows:

Definition 7.1 (DiskPacking). *Given a set of nonidentical unit disks in $\mathbb{R}^3$ by their normal vectors in $\mathbb{Z}^3$, the goal is to find*



(i) Input (ii) Solution

Figure 7.1: Example for disk-packing

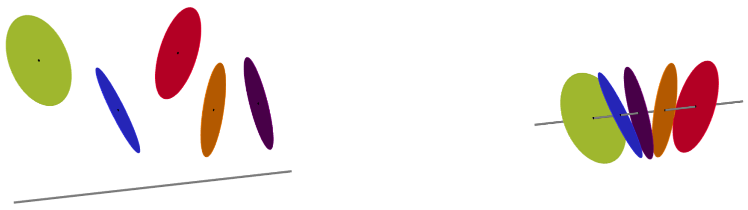
- *an axis-parallel box of minimum volume such that all disks can be packed without overlapping under translation inside the box*
- *and the actual packing of the disks inside the box.*

For an example, see Fig. 7.1.

Definition 7.2 (length of an ordering, DiskStabbing)**.** *Let O be a given ordering of a set of unit disks and $\mathbf{s} \in \mathbb{R}^3$. We define the length of O in the following way. If the disks are placed nonoverlappingly with their centers in order O on a line supported by $\mathbf{s}$ such that consecutive disks touch, we call the distance from the first disk center to the center of the last disk* the length of the ordering O *with respect to* $\mathbf{s}$.

For the DiskStabbing-problem, we are given a set of nonidentical unit disks by their normal vectors in $\mathbb{Z}^3$ and an additional vector $\mathbf{s} \in \mathbb{Z}^3$ defining the direction of a line. A solution to the DiskStabbing-problem is an ordering of the disks with minimum length.

See Figures 7.2i and 7.2ii for 3D-disk-stabbing and Fig. 7.3 for 2D-disk-stabbing.



(i) Input (ii) Solution

Figure 7.2: Example for 3D-disk-stabbing

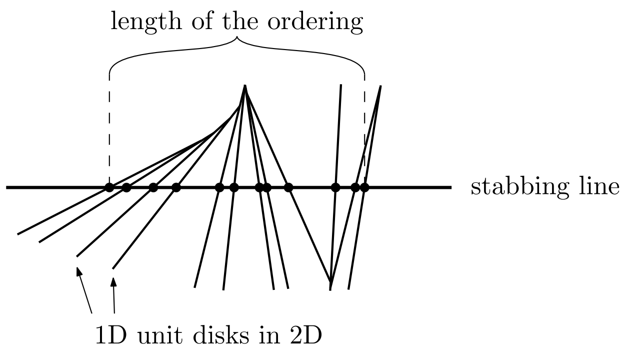


Figure 7.3: Example for 2D-disk-stabbing. Here, the unit disks are unit line segments. Note the length of the ordering.

In Section 7.1, we show that DiskPacking can be approximated by a constant factor in polynomial time. We will reduce approximating DiskPacking by a constant factor to approximating DiskStabbing by a constant factor. DiskStabbing then again will be reduced to finding the shortest Hamiltonian path in a complete weighted graph. In general, approximating the shortest Hamiltonian path is hard, but if the graph satisfies the triangle inequality, the problem admits a polynomial time approximation. In Section 7.2 we address more general inputs like squares. These results are complemented by Section 7.3, where we show that there is no container of constant size in which all unit disks can be packed.

7.1 Approximation Algorithms

In the following, we first define the distance between two given disks stabbed by a given line, which will be used as weights in a complete graph on the disks. Then, we use this graph to show how to reduce DiskStabbing to finding the shortest Hamiltonian path in a complete weighted graph so as to obtain a constant-factor approximation algorithm. Afterwards, we will use this approximation algorithm to compute a constant-factor approximation for DiskPacking.

7.1.1 Measuring the Distance Between Disks

Definition 7.3. *Given a vector* $\mathbf{s}$ *and two disks* D_1 *and* D_2 *in* $\mathbb{R}^3$*, the distance* $\mathrm{h}_s(D_1, D_2)$ *between* D_1 *and* D_2 *with respect to* $\mathbf{s}$ *is defined by the distance of the centers of* D_1 *and* D_2 *when placed with their centers on a line supported by the vector* $\mathbf{s}$ *such that* D_1 *and* D_2 *touch. For the special case, that* D_1 *and* D_2 *are identical, we define* $\mathrm{h}_s(D_1, D_2) = 0$.

Fix a vector $\mathbf{s}$ in $\mathbb{Z}^3$. We show that h_s forms a metric on unit disks and that h_s^2 can be computed easily.

Lemma 7.4. *For any* $\mathbf{s} \in S^2$, h_s *is a metric on the set of unit disks (modulo translation).*

Proof. If D_1 and D_2 are not identical, it is clear that $h_s(D_1, D_2) > 0$. Otherwise, $h_s(D_1, D_2) = 0$ by definition.

Symmetry also can easily be observed: assume that D_1 and D_2 are stabbed in that order so that they touch by a line with direction $\mathbf{s}$ through the origin. Then a point reflection about the origin will preserve the orientation of the disks (invert their normal vectors) and the distance of their centers, whereas their order on the stabbing line is reversed.

The triangle inequality holds obviously, if at least two of three disks are identical. Hence, we focus on the case where all three disks are different. We will show the triangle inequality by showing that if three disks D_1, D_2, and D_3 with centers $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ respectively are stabbed in that order so that D_1 touches D_2 and D_2 touches D_3 then D_3 cannot intersect into D_1 (they may touch in a point contained in D_2), i.e., the distance from the center of D_1 to the center of D_3 is at least $h_s(D_1, D_3)$. This is done by contradiction: assume that D_1 and D_3 intersect, i.e., $D_1 \cap D_3$ is a non-degenerate line segment meaning that $h_s(D_1, D_2) + h_s(D_2, D_3) < h_s(D_1, D_3)$. Let $\mathbf{v}$ be any point in the interior of $D_1 \cap D_3$. This implies that the distances from $\mathbf{v}$ to $\mathbf{c_1}$ and from $\mathbf{v}$ to $\mathbf{c_3}$ are both less than one. Observe that if $\mathbf{v}$ cannot be contained in the interior of D_2, since otherwise D_1 would intersect D_2.

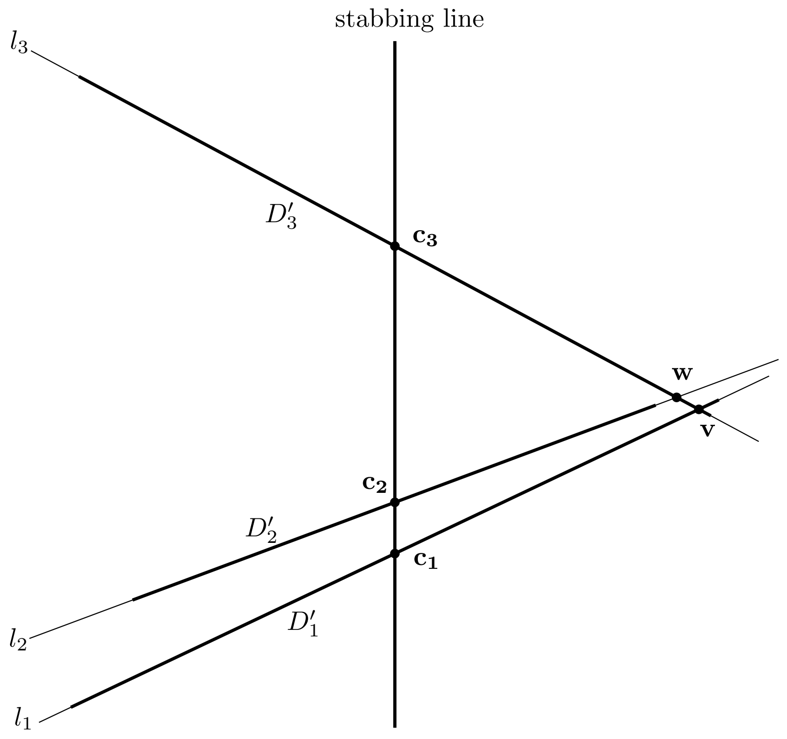


Figure 7.4: Situation in plane E. Length of D'_1, D'_2, D'_3 not drawn to scale.

The following proof setup is illustrated in Fig. 7.4. Consider the plane E spanned by $\mathbf{v}, \mathbf{c_1}, \mathbf{c_3}$. Observe that $\mathbf{c_2}$ lies in that plane since $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ lie on the stabbing line. Let

D'_1, D'_2, D'_3 be the intersections of D_1, D_2, D_3 with E and l_1, l_2, l_3 the lines supporting D'_1, D'_2, D'_3 respectively. D'_1, D'_2, D'_3 are line segments of length two since the centers of the disks are contained in E. Without loss of generality, we assume that E is the xy-plane, the y-axis is the stabbing line, $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ have increasing y-coordinates in this order, $\mathbf{c_1}$ lies at the origin, and $\mathbf{v}$ has positive x-coordinate. Let m_1, m_2, m_3 be the slopes of l_1, l_2, l_3 respectively. Observe that $m_1 > m_3$ for the setting described before. For three pairwise distinct points $\mathbf{p}, \mathbf{q}, \mathbf{r} \in E$, we denote by $\overline{\mathbf{pq}}$ the line segment with endpoints $\mathbf{p}, \mathbf{q}$, by $|\overline{\mathbf{pq}}|$ its length, and by $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ the triangle with vertices $\mathbf{p}, \mathbf{q}, \mathbf{r}$. Let $\mathbf{w}$ be the intersection of l_2 and either $\overline{\mathbf{c_1 v}}$ or $\overline{\mathbf{c_3 v}}$. Observe that l_2 intersects $\triangle(\mathbf{c_1}, \mathbf{c_3}, \mathbf{v})$ exactly twice, at $\mathbf{c_2}$ and $\mathbf{w}$. Furthermore, $|\overline{\mathbf{c_2 w}}| \geq 1$ since otherwise D_2 would intersect D_1 or D_3. We will contradict this inequality in the following case distinction.
Case 1: $m_2 \in [m_3, m_1]$

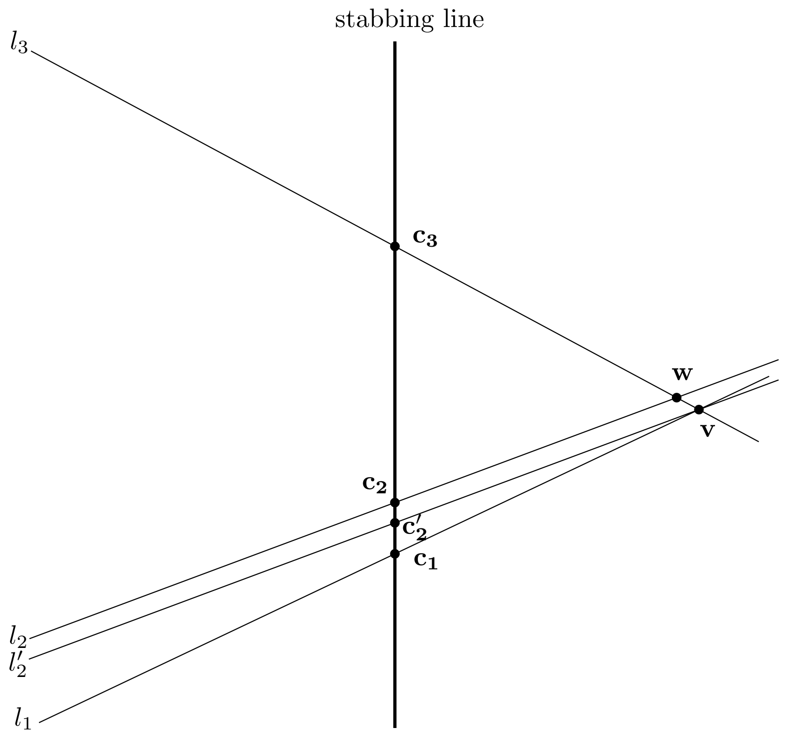


Figure 7.5: The distance from $\mathbf{c_2}$ to $\mathbf{w}$ is at most the distance from $\mathbf{c'_2}$ to $\mathbf{v}$ which is less than one.

Consider the line l'_2 passing through $\mathbf{v}$ and parallel to l_2, intersecting the stabbing line at $\mathbf{c_2}'$ (see Fig. 7.5), and notice that $|\overline{\mathbf{c_2 w}}| \leq |\overline{\mathbf{c'_2 v}}|$. However, since $\overline{\mathbf{c_1 v}}$ is the longest edge in $\triangle(\mathbf{c_1}, \mathbf{c'_2}, \mathbf{v})$ or $\overline{\mathbf{c_3 v}}$ is the longest edge in $\triangle(\mathbf{c_3}, \mathbf{c'_2}, \mathbf{v})$, $|\overline{\mathbf{c'_2 v}}| \leq \max(|\overline{\mathbf{c_1 v}}|, |\overline{\mathbf{c_3 v}}|) < 1$, contradicting our assumption.
Case 2: $m_2 > m_1$
We summarize the following setting in Fig. 7.6. The case constraint $m_2 > m_1$ implies that there is an intersection of l_1 and l_2 with negative x-coordinate. Let $\mathbf{u}$ be this

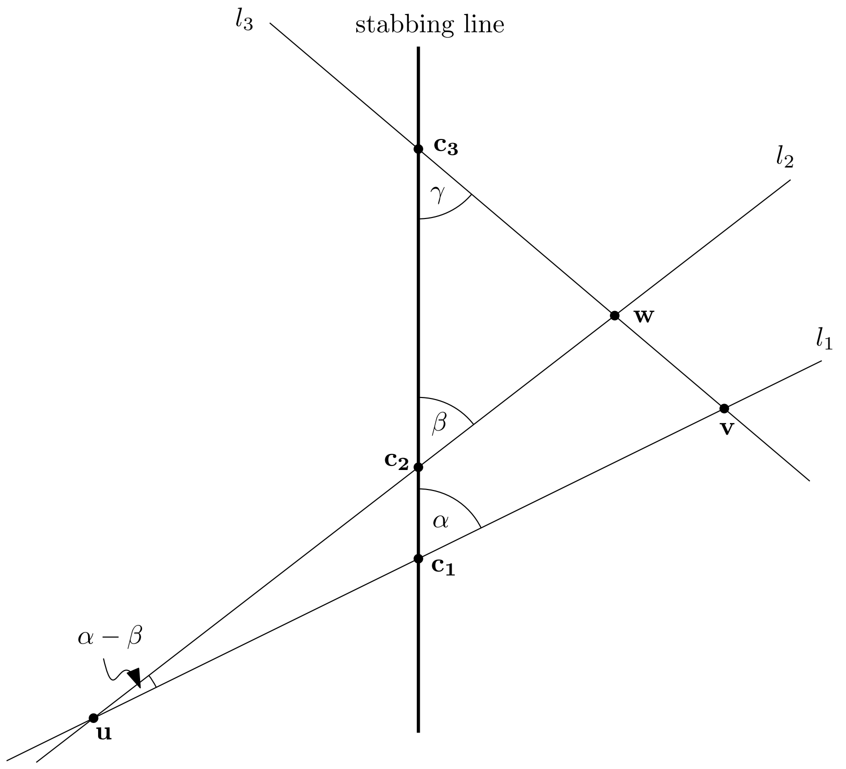


Figure 7.6: Situation in E when $\overline{\mathbf{c_1c_3}}$ is longest edge in $\triangle(\mathbf{c_1}, \mathbf{c_3}, \mathbf{v})$ and $m_2 > m_1$

intersection point. Furthermore, $\mathbf{w}$ has to lie on the line segment $\overline{\mathbf{c_3v}}$. Let α be the inner angle at $\mathbf{c_1}$ and γ be the inner angle at $\mathbf{c_3}$ in the triangle $\triangle(\mathbf{c_1}, \mathbf{c_3}, \mathbf{v})$. Let β be the inner angle at $\mathbf{c_2}$ in $\triangle(\mathbf{c_2}, \mathbf{c_3}, \mathbf{w})$. Since $|\overline{\mathbf{c_2w}}| \geq 1$, the longest side of $\triangle(\mathbf{c_1}, \mathbf{c_3}, \mathbf{v})$ has length at least 1, and since other sides have length less than 1, the longest side should be $\overline{\mathbf{c_1c_3}}$. This implies that

$$0 < \alpha, \gamma < \pi/2, \tag{7.1}$$

and together with $m_2 > m_1$ that $\mathbf{u}$ has a negative y-coordinate. We do another case distinction depending on α and γ.

Case 2.1: $\gamma \leq \alpha$

We will show that the following function f gives an upper bound for $|\overline{\mathbf{c_2w}}|$ depending on β:

$$\mathrm{f}(\beta) := \frac{(\sin(\alpha+\gamma) - \sin(\alpha-\beta))\sin\gamma}{\sin(\beta+\gamma)\sin\alpha}. \tag{7.2}$$

Furthermore, we will see that f is strictly monotone increasing for $\beta \in [0, \alpha]$ and we have

$0 < \beta < \alpha$ from the case constraint $m_2 > m_1$. Together, this implies

$$\begin{aligned}|\overline{\mathbf{c_2 w}}| &< \mathrm{f}(\alpha)\\ &= \frac{(\sin(\alpha+\gamma) - \sin(\alpha-\alpha))\sin\gamma}{\sin(\alpha+\gamma)\sin\alpha}\\ &= \frac{\sin\gamma}{\sin\alpha} \leq 1,\end{aligned}$$

since $0 < \gamma \leq \alpha < \pi/2$ by the case constraint.

It remains to show that $\mathrm{f}(\beta)$ is an upper bound for $|\overline{\mathbf{c_2 w}}|$ and that f is strictly monotone increasing for $\beta \in [0, \alpha]$. We start with the former.

We first apply the sine law in $\triangle(\mathbf{c_2}, \mathbf{c_3}, \mathbf{w})$:

$$|\overline{\mathbf{c_2 w}}| = \frac{|\overline{\mathbf{c_2 c_3}}|\sin(\gamma)}{\sin(\beta+\gamma)} = \frac{(|\overline{\mathbf{c_1 c_3}}| - |\overline{\mathbf{c_1 c_2}}|)\sin(\gamma)}{\sin(\beta+\gamma)}. \tag{7.3}$$

Recall that $|\overline{\mathbf{c_3 v}}| < 1$. From the sine law in $\triangle(\mathbf{c_1}, \mathbf{c_3}, \mathbf{v})$ we get:

$$|\overline{\mathbf{c_1 c_3}}| = \frac{|\overline{\mathbf{c_3 v}}|\sin(\alpha+\gamma)}{\sin(\alpha)} < \frac{\sin(\alpha+\gamma)}{\sin(\alpha)}. \tag{7.4}$$

Recall that $\max(|\overline{\mathbf{c_1 u}}|, |\overline{\mathbf{c_2 u}}|) \geq 1$ and notice that $|\overline{\mathbf{c_2 u}}| > |\overline{\mathbf{c_1 u}}|$ since $\mathbf{u}$ has a negative y-coordinate. We get an inequality for $|\overline{\mathbf{c_1 c_2}}|$ by the sine law in $\triangle(\mathbf{c_1}, \mathbf{c_2}, \mathbf{u})$:

$$|\overline{\mathbf{c_1 c_2}}| = \frac{|\overline{\mathbf{c_2 u}}|\sin(\alpha-\beta)}{\sin(\alpha)} \geq \frac{\sin(\alpha-\beta)}{\sin(\alpha)}. \tag{7.5}$$

Plugging Eqs. (7.4) and (7.5) into Eq. (7.3) gives

$$\begin{aligned}\frac{|\overline{\mathbf{c_2 w}}|}{\sin(\gamma)} &= \frac{(|\overline{\mathbf{c_1 c_3}}| - |\overline{\mathbf{c_1 c_2}}|)\sin(\gamma)}{\sin(\beta+\gamma)}\\ &< \frac{(\sin(\alpha+\gamma) - \sin(\alpha-\beta))\sin\gamma}{\sin(\beta+\gamma)\sin\alpha}\\ &= \mathrm{f}(\beta).\end{aligned}$$

Now, we show that f(β) is strictly monotonically increasing:

$$\begin{aligned}
\mathrm{f}'(\beta) &= \frac{d\mathrm{f}(\beta)}{d\beta} \\
&= \frac{\sin\gamma}{\sin\alpha} \cdot \frac{\sin(\beta+\gamma)\cos(\alpha-\beta) - (\sin(\alpha+\gamma) - \sin(\alpha-\beta))\cos(\beta+\gamma)}{\sin^2(\beta+\gamma)} \\
&= \frac{\sin\gamma}{\sin\alpha} \cdot \\
&\quad \frac{\sin(\beta+\gamma)\cos(\alpha-\beta) + \cos(\beta+\gamma)\sin(\alpha-\beta) - \sin(\alpha+\gamma)\cos(\beta+\gamma)}{\sin^2(\beta+\gamma)} \\
&= \frac{\sin\gamma}{\sin\alpha} \cdot \frac{\sin(\beta+\gamma+\alpha-\beta) - \sin(\alpha+\gamma)\cos(\beta+\gamma)}{\sin^2(\beta+\gamma)} \\
&= \frac{\sin\gamma}{\sin\alpha} \cdot \frac{\sin(\alpha+\gamma)(1-\cos(\beta+\gamma))}{\sin^2(\beta+\gamma)}.
\end{aligned}$$

Since $0 < \alpha, \gamma < \pi/2$ (Eq. (7.1)), we have $\sin\gamma / \sin\alpha > 0$ and $\sin(\alpha+\gamma) > 0$. Since $0 < \beta \leq \alpha$, we have $\sin^2(\beta+\gamma) > 0$ and $\cos(\beta+\gamma) < 1$. Together, we have $\mathrm{f}'(\beta) > 0$ for $\beta \in [0, \alpha]$. This completes the argument for this case.

Case 2.2: $\gamma > \alpha$

We have a situation as shown in Fig. 7.7. We adjust the situation in the following way in order to apply case 2.1. Let $\mathbf{c_3'}$ be the second point at distance $|\overline{\mathbf{c_1 v}}|$ from $\mathbf{v}$ on the stabbing line (the first is $\mathbf{c_1}$), forming an isosceles triangle $\triangle(\mathbf{c_1}, \mathbf{c_3'}, \mathbf{v})$. Let $\mathbf{w}'$ be the intersection point of l_2 with $\overline{\mathbf{c_3' v}}$. Observe that since $\alpha < \gamma$, we have $|\overline{\mathbf{c_1 v}}| > |\overline{\mathbf{c_3 v}}|$. Therefore, $\mathbf{c_3'}$ has a larger y-coordinate than $\mathbf{c_3}$ and $\overline{\mathbf{c_2 w}'}$ is longer than $\overline{\mathbf{c_2 w}}$ Hence, showing $|\overline{\mathbf{c_2 w}'}| < 1$ will complete the proof. Analogously to the proof in case 2.1 when using $\mathbf{c_3'}$ instead of $\mathbf{c_3}$ and $\mathbf{w}'$ instead of $\mathbf{w}$, we get the desired result $|\overline{\mathbf{c_2 w}'}| < 1$.

Case 3: $m_2 < m_3$

This case is symmetric to case 2 and can therefore be handled analogously.

In all cases, we showed $|\overline{\mathbf{c_2 w}}| < 1$. This concludes the proof. □

As explained before, we show next, that $h_s^2(D_1, D_2)$ can be easily computed, i.e., we show the following theorem.

Lemma 7.5. *Given two disks D_1 and D_2 by their normal vectors $\mathbf{n_1}, \mathbf{n_2}$ in $\mathbb{Z}^3$, $\mathrm{h}_s^2(D_1, D_2)$ is a rational number whose numerator and denominator can be computed in time polynomial in the length of the bit-representations of the coordinates of $\mathbf{n_1}$, $\mathbf{n_2}$, and $\mathbf{s}$.*

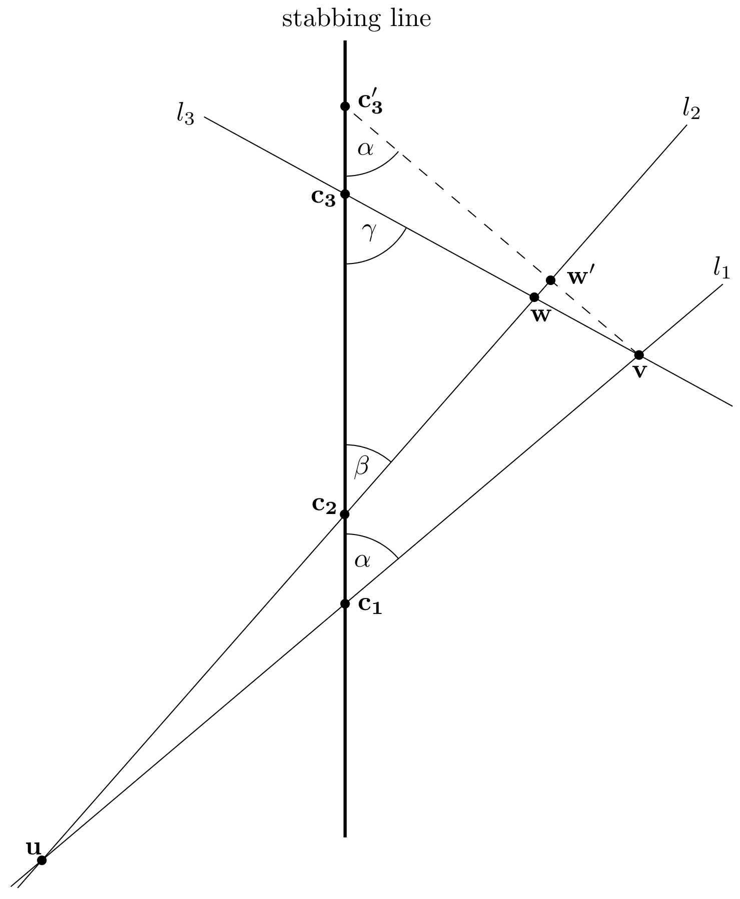


Figure 7.7: Situation in E when $\overline{\mathbf{c_1 c_3}}$ is longest edge in $\triangle(\mathbf{c_1}, \mathbf{c_3}, \mathbf{v})$, $m_2 > m_1$, and $\gamma > \alpha$.

Proof. Let

$$\mathbf{n_1} = \begin{pmatrix} n_{1,x} \\ n_{1,y} \\ n_{1,z} \end{pmatrix}$$

$$\mathbf{n_2} = \begin{pmatrix} n_{2,x} \\ n_{2,y} \\ n_{2,z} \end{pmatrix},$$

$$\mathbf{s} = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}.$$

We assume without loss of generality that D_1 and D_2 are not identical. Place the center of D_1 at the origin and denote the line in direction $\mathbf{s}$ through the origin by l. Place D_2 with its center $\mathbf{p}$ on l at distance $\mathrm{h}_s(D_1, D_2)$ from the origin, i.e., D_1 and D_2 touch.

Since $\mathbf{p}$ lies on l, its coordinates have the form $\mathbf{p} = (h \cdot s_x, h \cdot s_y, h \cdot s_z)$. We can assume without loss of generality that $h > 0$ since h_s is symmetric by Lemma 7.4. Note that in this setting, $h = \frac{\mathrm{h}_s(D_1,D_2)}{|\mathbf{s}|}$. Hence, we will show how to obtain h^2 in order to compute $(\mathrm{h}_s(D_1, D_2))^2$. Let E_1, E_2 be the planes containing D_1, D_2 respectively and g the intersection line between E_1 and E_2.

E_1, E_2 can be described by the following equations:

$$E_1 : \qquad \mathbf{n_1} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0,$$

$$E_2 : \qquad \mathbf{n_2} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = h \cdot (\mathbf{n_2} \cdot \mathbf{s}).$$

The intersection line g is the set of solutions for this system of equations. It can be described in parameter form with parameter $t \in \mathbb{R}$:

$$g : \begin{pmatrix} x \\ y \\ z \end{pmatrix} = h \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} + t \cdot \begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix}.$$

The components of the vector $\mathbf{k} = \begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix}$ can easily be obtained by computing the cross product of the normal vectors $\mathbf{n_1}$ and $\mathbf{n_2}$. $h \cdot (a_x, a_y, a_z)$ is a point on g and can be obtained in the following way. Note that there is at least one plane among the x-y-plane, the y-z-plane, and the x-z-plane such that g is not parallel to this plane, i.e., g intersects this plane. Let without loss of generality this plane be the x-y-plane. Hence, we set $z = 0$ in the system of equations above (given by the planes spanned by the two disks) and solve it to obtain values for x and y. Then, we have $ha_x = x, ha_y = y$ and $ha_z = 0$. Note that the formulas for $k_x, k_y, k_z, a_x, a_y, a_z$ are fractions of quadratic polynomials of bounded length consisting only of the components of $\mathbf{n_1}$, $\mathbf{n_2}$, and $\mathbf{s}$.

Recall, that D_1 and D_2 are stabbed by the line l in direction $\mathbf{s}$ and they touch. This implies that there is exactly one point on g that has distance exactly one to one of the centers of D_1, D_2 (the origin and $\mathbf{p}$) and distance at most one to the other.

Let $d_1(t)$ be the squared distance of the origin to the point on g given by t. Likewise, let $d_2(t)$ be the squared distance of $\mathbf{p}$ to the point on g given by t. Let $\mathbf{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$. So,

$$\begin{aligned} d_1(t) =& (h \cdot \mathbf{a} + t \cdot \mathbf{k})^2 \\ =& h^2 \cdot \mathbf{a} \cdot \mathbf{a} + 2ht \cdot \mathbf{a} \cdot \mathbf{k} + t^2 \cdot \mathbf{k} \cdot \mathbf{k}, \qquad (7.6) \\ d_2(t) =& (h \cdot \mathbf{a} + t \cdot \mathbf{k} - h \cdot \mathbf{s})^2 \\ =& h^2 \cdot (\mathbf{a} - \mathbf{s}) \cdot (\mathbf{a} - \mathbf{s}) + 2ht \cdot ((\mathbf{a} - \mathbf{s}) \cdot \mathbf{k}) + t^2 \cdot \mathbf{k} \cdot \mathbf{k}. \qquad (7.7) \end{aligned}$$

Since the coefficients of t^2 are equal in d_1, d_2, their graphs are identical modulo translation. Therefore, there are three possibilities to meet the requirement explained before, i.e., that there exists exactly one value t^* such that $d_1(t^*) = 1$ and $d_2(t^*) \leq 1$ or $d_2(t^*) = 1$ and $d_1(t^*) \leq 1$ (see Fig. 7.8 for examples).

1. $\min_t(d_1(t)) = d_1(t^*) = 1$ and $d_2(t^*) \leq 1$. This means that D_1 touches D_2 in its interior (g is a tangent of D_1 and intersects D_2).
2. $\min_t(d_2(t)) = d_2(t^*) = 1$ and $d_1(t^*) \leq 1$. This means that D_2 touches D_1 in its interior (g is a tangent of D_2 and intersects D_1).
3. $d_1(t^*) = d_2(t^*) = 1$. This means that D_1 and D_2 touch at their boundaries. In this case, it is possible that $D_1 \cap g$ and $D_2 \cap g$ are line segments of positive length. Note that it is crucial to test 1 and 2 first since in these cases it is possible to place D_1, D_2 such that there exist t', t'' with $d_1(t') = d_2(t') = 1$ and $d_1(t'') < 1$ **and** $d_2(t'') < 1$, i.e., D_1 and D_2 intersect.

Note that in any other case the disks intersect (there is more than one t that fulfills the requirement) or do not touch at all (no t fulfills the requirement). In the following, we explain how to check the possibilities and obtain h^2.

First, we check if there is an h such that the minimum of d_1 is one and lies above d_2. To obtain the minimum of d_1, we set its derivative to zero and get

$$t_{\min_1} = -\frac{h \cdot \mathbf{a} \cdot \mathbf{k}}{\mathbf{k} \cdot \mathbf{k}}.$$

Now, we plug this formula into $d_1(t_{\min_1}) = 1$ and get:

$$\begin{aligned} 1 =& h^2 \cdot \mathbf{a} \cdot \mathbf{a} - 2h \frac{h \cdot \mathbf{a} \cdot \mathbf{k}}{\mathbf{k} \cdot \mathbf{k}} \cdot \mathbf{a} \cdot \mathbf{k} \\ &+ \left(\frac{h \cdot \mathbf{a} \cdot \mathbf{k}}{\mathbf{k} \cdot \mathbf{k}}\right)^2 \cdot \mathbf{k} \cdot \mathbf{k} \\ =& h^2 \cdot \left(\mathbf{a} \cdot \mathbf{a} - \frac{(\mathbf{a} \cdot \mathbf{k})^2}{\mathbf{k} \cdot \mathbf{k}}\right). \end{aligned} \tag{7.8}$$

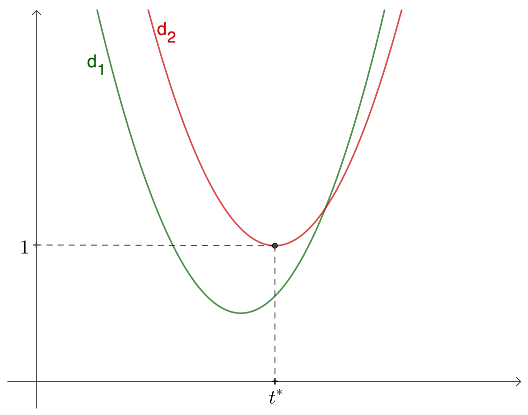
In case $\mathbf{a} \cdot \mathbf{a} - \frac{(\mathbf{a} \cdot \mathbf{k})^2}{\mathbf{k} \cdot \mathbf{k}} = 0$, we have

$$\mathbf{a} \cdot \mathbf{a} = \frac{(\mathbf{a} \cdot \mathbf{k})^2}{\mathbf{k} \cdot \mathbf{k}},$$

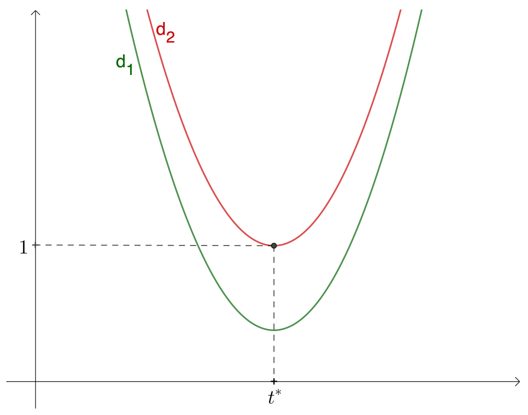
which is equivalent to

$$|\mathbf{a}|^2 = \frac{(|\mathbf{a}||\mathbf{k}| \cos(\mathbf{a}, \mathbf{k}))^2}{|\mathbf{k}|^2},$$

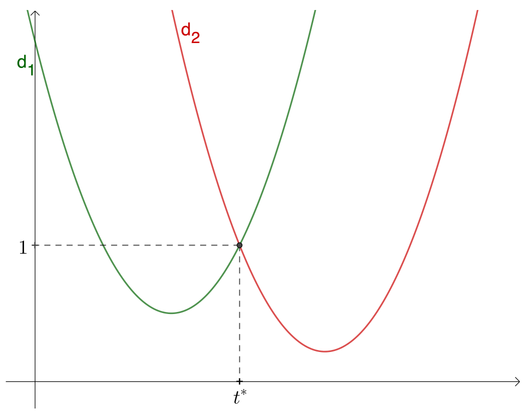
so $\cos(\mathbf{a}, \mathbf{k}) = 1$, which implies that g contains the center of D_1. In this case, we proceed with checking if there is a t^* with $\min_t(d_2(t)) = d_2(t^2) = 1$ and $d_1(t^*) \leq 1$.



(i) One example for $\min_t(d_2(t)) = d_1(t^*) = 1$ and $d_1(t^*) \leq 1$.



(ii) Another example for $\min_t(d_2(t)) = d_1(t^*) = 1$ and $d_1(t^*) \leq 1$.



(iii) An example for $d_1(t^*) = d_2(t^*) = 1$.

Figure 7.8: Examples for d_1 and d_2 if D_1 and D_2 touch.

If $\mathbf{a} \cdot \mathbf{a} - \frac{(\mathbf{a} \cdot \mathbf{k})^2}{\mathbf{k} \cdot \mathbf{k}} \neq 0$, we obtain immediately from Eq. (7.8)

$$h^2 = \frac{1}{\mathbf{a} \cdot \mathbf{a} - \frac{(\mathbf{a} \cdot \mathbf{k})^2}{\mathbf{k} \cdot \mathbf{k}}}. \tag{7.9}$$

Observe that this value can be computed in polynomial time from the input integers. Now, we need to check if $d_2(t_{\min_1}) \leq 1$. We have

$$\begin{aligned} d_2(t_{\min_1}) =& h^2(\mathbf{a} - \mathbf{s})^2 - 2h\frac{h \cdot \mathbf{a} \cdot \mathbf{k}}{\mathbf{k} \cdot \mathbf{k}} \cdot (\mathbf{a} - \mathbf{s}) \cdot \mathbf{k} + \left(\frac{h \cdot \mathbf{a} \cdot \mathbf{k}}{\mathbf{k} \cdot \mathbf{k}}\right)^2 k^2 \\ =& h^2\left((\mathbf{a} - \mathbf{s})^2 - 2\frac{\mathbf{a} \cdot \mathbf{k}}{\mathbf{k} \cdot \mathbf{k}} \cdot (\mathbf{a} - \mathbf{s}) \cdot \mathbf{k} + \frac{(\mathbf{a} \cdot \mathbf{k})^2}{\mathbf{k} \cdot \mathbf{k}}\right). \end{aligned}$$

We can now insert Eq. (7.9) and check in a straight forward manner if this term evaluates to at most one.

If not, we proceed with the second possibility, i.e., we perform the same steps with d_1, d_2 exchanged. If the minimum of d_2 for the obtained value of h^2 does not lie above d_1, we proceed with the third possibility.

Therefore, set $d_1(t_\chi) = d_2(t_\chi) = 1$. Since $h > 0$, $d_1(t_\chi) = d_2(t_\chi)$ gives

$$h \cdot \mathbf{a} \cdot \mathbf{a} + 2t_\chi \cdot \mathbf{a} \cdot \mathbf{k} = h \cdot (\mathbf{a} - \mathbf{s})^2 + 2t_\chi \cdot (\mathbf{a} - \mathbf{s}) \cdot \mathbf{k},$$

so,

$$\mathbf{a} \cdot \mathbf{a} - (\mathbf{a} - \mathbf{s})^2 = 2t_\chi \cdot ((\mathbf{a} - \mathbf{s}) \cdot \mathbf{k} - \mathbf{a} \cdot \mathbf{k}),$$

which is equivalent to

$$((2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}) \cdot h = -\mathbf{s} \cdot \mathbf{k} \cdot 2t_\chi. \tag{7.10}$$

Note that $\mathbf{s} \cdot \mathbf{k} = 0$ would imply $(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s} = 0$ as well, since we know that $h > 0$. Hence, all coefficients in d_1 are equal to the corresponding ones in d_2 (see Eqs. (7.6) and (7.7)), i.e., the functions d_1 and d_2 are equal. This was already handled when checking if there exists a t^* such that $\min_t(d_1(t)) = d_1(t^*) = 1$ and $d_2(t^*) \leq 1$. So, we obtain

$$t_\chi = -\frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{2 \cdot \mathbf{s} \cdot \mathbf{k}} \cdot h.$$

Hence, we get

$$\begin{aligned} d_1(t_\chi) &= h^2 \cdot \mathbf{a} \cdot \mathbf{a} + 2h \cdot \left(-\frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{2 \cdot \mathbf{s} \cdot \mathbf{k}} \cdot h\right) \cdot \mathbf{a} \cdot \mathbf{k} + \left(\frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{2 \cdot \mathbf{s} \cdot \mathbf{k}} \cdot h\right)^2 \cdot \mathbf{k} \cdot \mathbf{k} \\ &= h^2 \cdot \left(\mathbf{a} \cdot \mathbf{a} - \frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{\mathbf{s} \cdot \mathbf{k}} \cdot \mathbf{a} \cdot \mathbf{k} + \left(\frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{2 \cdot \mathbf{s} \cdot \mathbf{k}}\right)^2 \cdot \mathbf{k} \cdot \mathbf{k}\right). \end{aligned}$$

Observe that t_χ describes the point on g with equal distance to the origin and $\mathbf{p}$. This distance can only be zero if the origin and $\mathbf{p}$ are identical, i.e., $h = 0$. Since we know, that $h > 0$, we can conclude that $\mathbf{a} \cdot \mathbf{a} - \frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{\mathbf{s} \cdot \mathbf{k}} \cdot \mathbf{a} \cdot \mathbf{k} + \left(\frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{2 \cdot \mathbf{s} \cdot \mathbf{k}}\right)^2 \cdot \mathbf{k} \cdot \mathbf{k} > 0$. So, setting $d_1(t_\chi) = 1$ gives

$$h^2 = \frac{1}{\mathbf{a} \cdot \mathbf{a} - \frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{\mathbf{s} \cdot \mathbf{k}} \cdot \mathbf{a} \cdot \mathbf{k} + \left(\frac{(2\mathbf{a} - \mathbf{s}) \cdot \mathbf{s}}{2 \cdot \mathbf{s} \cdot \mathbf{k}}\right)^2 \cdot \mathbf{k} \cdot \mathbf{k}}.$$

Altogether, we obtained for every possibility how D_1 and D_2 can touch a formula for h^2 that is a rational function of constant degree in the input integers. Hence, we can compute the numerator and denominator of the rational number h^2 in polynomial time. This completes the proof.

$\square$

```
Algorithm 7.1: Approximation algorithm for disk-stabbing
  Input: $n$ unit disks given by their normal vectors, vector $\mathbf{s}$
  Output: Ordering of the $n$ disks
1 Generate a complete weighted graph $G$ with $n$ vertices:
2 Set the weight of the edge $(i, j)$ to $\mathrm{h}_s(D_i, D_j)$ for all $1 \leq i, j \leq n, i \neq j$;
3 Approximate the shortest Hamiltonian path on the graph with the tree
    heuristic;
4 return the ordering of the overall shortest path;
```

7.1.2 Disk-stabbing Approximation

We determine an approximate solution for the DiskStabbing-problem in Algorithm 7.1. The idea is to consider a complete weighted graph, where the vertices correspond to the disks and the weight of an edge (D_1, D_2) is $h_s(D_1, D_2)$. A Hamiltonian path in this graph corresponds to an ordering of the disks.

Theorem 7.6. *Algorithm 7.1 computes a 2-approximation for disk-stabbing in polynomial time.*

Proof. By Lemma 7.4 the triangle inequality holds in G. Let $O = D_{i_1}, D_{i_2}, \ldots D_{i_n}$ be the optimal ordering for the input instance and OPT the length of O when stabbed by a line in direction $\mathbf{s}$ in order O. Then, there is a path in G from vertex i_1 to i_n visiting each vertex exactly once, i.e., a Hamiltonian path, of length OPT. Now, we can utilize the tree heuristic (see e.g. [17]), which finds in polynomial time a Hamiltonian path of length at most 2 OPT with no specified starting and ending point in a graph that satisfies the triangle inequality. To implement the tree heuristic in polynomial time, we use for example Kruskals Algorithm to build an MST on the graph. Therefore, we have to sort the edges by length. The resulting ordering is equal when using instead of the actual values the squared values of the $\mathrm{h}_s(D_i, D_j)$ that we can compute in polynomial time by Lemma 7.5. Hence, we can compute an ordering of length at most 2 OPT in polynomial time. □

In the next section, we will use Algorithm 7.1 to approximate disk-packing.

7.1.3 Disk-packing Approximation

The idea for the approximation algorithm for disk-packing is as follows. We divide the disks into three subsets corresponding to the three axes such that the disks are almost orthogonal to the assigned axis. Then, we use Algorithm 7.1 to compute disk-stabbings of the three sets on the corresponding axes. The result can be interpreted as three containers, of which one is possibly very wide, one very high and the third very deep. The other two dimensions are relatively small. The last step is to divide these three boxes into pieces and arrange those pieces such that they form one single axis-parallel box. To describe the details of the algorithm, we use the following definitions.

We define $w_{\max}, d_{\max}, h_{\max}$ to be the maximum extent of any disk in x-,y-, and z-direction respectively and, thus, the minimum width, depth, and height any container for the disks must have. Let $\tilde{w}_{\max}$ be an approximation of $w_{\max}$ with $w_{\max} \leq \tilde{w}_{\max} \leq \left(1 + \frac{1}{2^k}\right) w_{\max}$ for an integer k defined later and define $\tilde{d}_{\max}, \tilde{h}_{\max}$ analogously. Let the width and depth of the final box be $w_{\text{box}} = a \cdot \tilde{w}_{\max}$ and $d_{\text{box}} = a \cdot \tilde{d}_{\max}$, respectively, for a constant $a > 1$ to be defined later. Algorithm 7.2 gives the details of the idea described above. For an illustration see Fig. 7.9.

Algorithm 7.2: Approximation algorithm for disk-packing

```
Input: n unit disks given by their normal vectors
Output: nonoverlapping packing of the disks into an axis-parallel box
1  Partition the $n$ disks into three sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ according to the axis their normal
     vectors form the smallest angle with;
2  Call Algorithm 7.1 for the disks in $\mathcal{X}$ and vector $(1, 0, 0)$. Approximate the
     distances of consecutive disks in the returned ordering with a factor
     $\left(1 + \frac{1}{2^{k+1}}\right)$ by rational numbers with bounded length and obtain an
     approximate length of the ordering $L_x$ from it. This can be interpreted as a
     packing of the disks in $\mathcal{X}$ into an axis-parallel box of width $W = L_x + \tilde{w}_{\max}$,
     depth $\tilde{d}_{\max}$, and height $\tilde{h}_{\max}$;
3  Analogously to Step 2 apply Algorithm 7.1 for the disks in $\mathcal{Y}$ and $\mathcal{Z}$ giving
     lengths $L_y$ and $L_z$, respectively. This can be seen as packing $\mathcal{Y}$ and $\mathcal{Z}$ into
     boxes of dimensions $\tilde{w}_{\max} \times D \times \tilde{h}_{\max}$ and $\tilde{w}_{\max} \times \tilde{d}_{\max} \times H$, respectively,
     where $D = L_y + \tilde{d}_{\max}$ and $H = L_z + \tilde{h}_{\max}$;
4  Divide the box obtained for $\mathcal{X}$ into pieces of width $(a - 1)\,\tilde{w}_{\max}$;
5  Assign each disk to the piece where the point in the disk with smallest
     x-coordinate lies;
6  Enlarge each piece from width $(a - 1)\,\tilde{w}_{\max} = w_{\text{box}} - \tilde{w}_{\max}$ to width $w_{\text{box}}$ s.t.
     all disks that are assigned to a piece are completely contained in that piece;
7  Divide the box obtained for $\mathcal{Y}$ into pieces of depth $\tilde{d}_{\text{box}}$ analogously to Steps 4
     to 6;
8  Divide the box obtained for $\mathcal{Z}$ into $\lfloor a \rfloor^2$ pieces of width $\tilde{w}_{\max}$ and depth $\tilde{d}_{\max}$;
9  Analogously to Steps 5 and 6, enlarge the height of each piece from step 8 by
     $\tilde{h}_{\max}$;
10 Arrange all pieces into a box of width $w_{\text{box}}$ and depth $d_{\text{box}}$, so that the pieces
     containing disks of $\mathcal{X}$ form $\left\lceil \left\lceil \frac{W}{w_{\text{box}} - \tilde{w}_{\max}} \right\rceil / \left\lfloor \frac{d_{\text{box}}}{\tilde{d}_{\max}} \right\rfloor \right\rceil$ layers of height $\tilde{h}_{\max}$, the
     pieces containing disks of $\mathcal{Y}$ form $\left\lceil \left\lceil \frac{D}{d_{\text{box}} - \tilde{d}_{\max}} \right\rceil / \left\lfloor \frac{w_{\text{box}}}{\tilde{w}_{\max}} \right\rfloor \right\rceil$ layers of height $\tilde{h}_{\max}$,
     and the pieces containing disks from $\mathcal{Z}$ form one layer of height
     $H / \left( \left\lfloor \frac{w_{\text{box}}}{\tilde{w}_{\max}} \right\rfloor \left\lfloor \frac{d_{\text{box}}}{\tilde{d}_{\max}} \right\rfloor \right) + \tilde{h}_{\max}$ (See Fig. 7.9 for an example);
11 return the resulting box with the packed disks;
```

To analyze Algorithm 7.2 we first give a bound on W, D, and H as defined in Al-

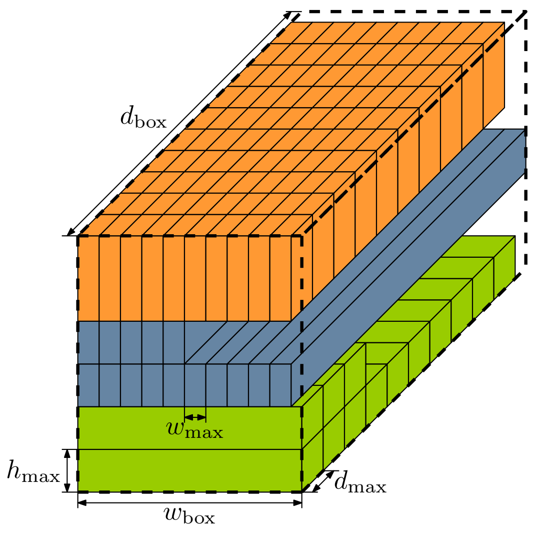


Figure 7.9: Example container for, e.g., $a = 10.5$. The green boxes are the enlarged pieces obtained by dividing the container-box computed by Algorithm 7.1 for the disks in $\mathcal{X}$. Here, they form two layers. The blue boxes contain disks from $\mathcal{Y}$ and the orange boxes contain disks from $\mathcal{Z}$.

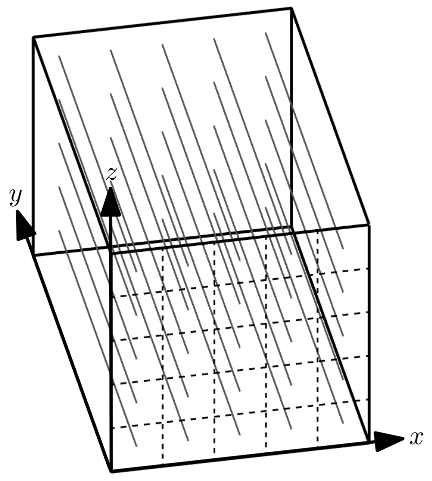
gorithm 7.2. Observe that the angle between the normal vector of a disk and the axis it gets stabbed by in Algorithm 7.2 is maximal when it is equal to the angle with the other two axis. Vector $\mathbf{v} = (1,1,1)$ forms the same angle with all three axis. The x-axis can be represented by the vector $\mathbf{x} = (1,0,0)$ and let φ be the maximum angle between a normal vector and the axis the corresponding disk gets stabbed by, i.e., the angle between $\mathbf{v}$ and $\mathbf{x}$. Then $\cos\varphi = \frac{(1,1,1)\cdot(1,0,0)}{\|(1,1,1)\|} = \frac{1}{\sqrt{3}}$. Hence, the maximum angle between the normal vector of any disk and the axis it gets stabbed by is $\varphi = \arccos(\frac{1}{\sqrt{3}})$.

Lemma 7.7. *It holds that*

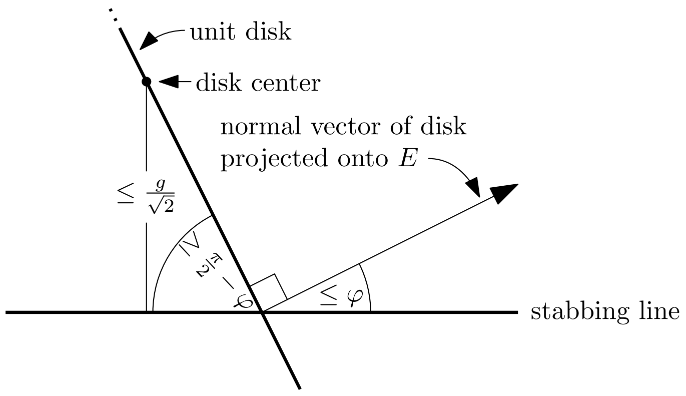
$$W \leq \left(130 + 66 \cdot \frac{1}{2^k}\right) \cdot \frac{\text{OPT}}{d_{\max} h_{\max}},$$
$$D \leq \left(130 + 66 \cdot \frac{1}{2^k}\right) \cdot \frac{\text{OPT}}{w_{\max} h_{\max}},$$
$$H \leq \left(130 + 66 \cdot \frac{1}{2^k}\right) \cdot \frac{\text{OPT}}{w_{\max} d_{\max}},$$

where OPT *is the volume of an optimal container.*

Proof. Consider an optimal container with width W_{OPT}, depth D_{OPT}, and height H_{OPT} and let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be the partition of disks into subsets as in Algorithm 7.2. Furthermore consider a square grid of grid cells with side length g on the xz-plane and lines parallel to the y-axis through the grid cell centers (see Fig. 7.10i for illustration). Then, each point has distance at most $\frac{g}{\sqrt{2}}$ to the closest line. So, in the optimal packing, every disk in $\mathcal{Y}$ is stabbed by a line in a point of distance at most $\frac{g}{\sqrt{2}\sin\left(\frac{\pi}{2}-\varphi\right)}$ from the disk center if g is small enough, i.e., $cg < 1$, where $c = \frac{1}{\sqrt{2}\sin\left(\frac{\pi}{2}-\varphi\right)} = \sqrt{\frac{3}{2}}$. See Fig. 7.10ii for illustration. Therefore, each disk in $\mathcal{Y}$ contains a disk of radius $1 - cg$ stabbed by a



(i) Optimal container with grid and line segments stabbing the disks.


unit disk
disk center
normal vector of disk projected onto E
$\leq \frac{g}{\sqrt{2}}$
$\geq \frac{\pi}{2} - \varphi$
$\leq \varphi$
stabbing line

(ii) Distance of a disk center to the stabbing line. The figure shows the cross section with the plane E that contains the stabbing line and the center of the disk.

Figure 7.10: Tools for disk-packing.

line through its center. So, by placing the $\left\lceil\frac{H_{\mathrm{OPT}}}{g}\right\rceil\left\lceil\frac{W_{\mathrm{OPT}}}{g}\right\rceil$ line segments of length D_{OPT} that are the intersection of the container and the lines behind each other so that they touch, we get a solution to the disk-stabbing problem for the disks in $\mathcal{Y}$ but with radius $1 - cg$. By stretching this solution by $1/(1 - cg)$, we get a solution for disks of radius 1 of length $\left\lceil\frac{H_{\mathrm{OPT}}}{g}\right\rceil\left\lceil\frac{W_{\mathrm{OPT}}}{g}\right\rceil D_{\mathrm{OPT}} \cdot \frac{1}{1-cg}$. Let $L_{\mathrm{OPT}_{\mathcal{Y}}}$ be the length of an optimal solution for the disk-stabbing problem for the disks in $\mathcal{Y}$. Then, this length can be at most the length of the solution just described, i.e.,

$$\begin{aligned}
L_{\mathrm{OPT}_{\mathcal{Y}}} &\leq \left\lceil \frac{H_{\mathrm{OPT}}}{g} \right\rceil \left\lceil \frac{W_{\mathrm{OPT}}}{g} \right\rceil D_{\mathrm{OPT}} \cdot \frac{1}{1-cg} \\
&\leq \frac{(H_{\mathrm{OPT}}+g)(W_{\mathrm{OPT}}+g)D_{\mathrm{OPT}}}{g^2(1-cg)} \\
&= \frac{H_{\mathrm{OPT}}W_{\mathrm{OPT}}D_{\mathrm{OPT}} + gW_{\mathrm{OPT}}D_{\mathrm{OPT}} + gH_{\mathrm{OPT}}D_{\mathrm{OPT}} + g^2 D_{\mathrm{OPT}}}{g^2(1-cg)} \\
&= \frac{1}{g^2(1-cg)}\left(\mathrm{OPT} + \frac{g\,\mathrm{OPT}}{H_{\mathrm{OPT}}} + \frac{g\,\mathrm{OPT}}{W_{\mathrm{OPT}}} + \frac{g^2\,\mathrm{OPT}}{H_{\mathrm{OPT}}W_{\mathrm{OPT}}}\right).
\end{aligned}$$

As observed earlier, we have $H_{\mathrm{OPT}} \geq h_{\max}$ and $W_{\mathrm{OPT}} \geq w_{\max}$. So, we get

$$\begin{aligned}
L_{\mathrm{OPT}_{\mathcal{Y}}} &\leq \frac{\mathrm{OPT}}{g^2(1-cg)}\left(1 + \frac{g}{h_{\max}} + \frac{g}{w_{\max}} + \frac{g^2}{h_{\max}w_{\max}}\right) \\
&= \frac{\mathrm{OPT}}{g^2(1-cg)} \cdot \frac{h_{\max}w_{\max} + gw_{\max} + gh_{\max} + g^2}{h_{\max}w_{\max}}.
\end{aligned}$$

Since the extent of a disk in any direction is at most two, we have $h_{max}, w_{\max} \leq 2$. This gives

$$\begin{aligned}
L_{\mathrm{OPT}_{\mathcal{Y}}} &\leq \frac{4+4g+g^2}{g^2(1-cg)} \cdot \frac{\mathrm{OPT}}{w_{\max}h_{\max}} \\
&= \frac{(g+2)^2}{g^2(1-cg)} \cdot \frac{\mathrm{OPT}}{w_{\max}h_{\max}}. \qquad (7.11)
\end{aligned}$$

Since we use Algorithm 7.1 to compute a disk-stabbing solution for $\mathcal{Y}$, we get by Theorem 7.6

$$\begin{aligned}
D &= L_x + \tilde{d}_{\max} \\
&\leq 2\left(1 + \frac{1}{2^{k+1}}\right) \cdot L_{\mathrm{OPT}_{\mathcal{Y}}} + 1 + \frac{1}{2^k} \cdot d_{\max},
\end{aligned}$$

where the extra term $\tilde{d}_{\max}$ comes from the fact that the length of a disk-stabbing is defined as the distance of the center of the first disk to the center of the last disk and we are interested in the total depth of the packing.

By inequality (7.11),

$$\begin{aligned} D &\le \left(2+\frac{1}{2^k}\right)\frac{(g+2)^2}{g^2(1-cg)}\cdot\frac{\mathrm{OPT}}{w_{\max}h_{\max}}+\left(1+\frac{1}{2^k}\right)\cdot d_{\max} \\ &\le \left(\left(2+\frac{1}{2^k}\right)\frac{(g+2)^2}{g^2(1-cg)}+\left(1+\frac{1}{2^k}\right)\right)\cdot\frac{\mathrm{OPT}}{w_{\max}h_{\max}}, \end{aligned}$$

since $d_{\max} \le D_{\mathrm{OPT}} = \frac{\mathrm{OPT}}{W_{\mathrm{OPT}}H_{\mathrm{OPT}}} \le \frac{\mathrm{OPT}}{w_{\max}h_{\max}}$. Optimizing for g (e.g. with programs like MAPLE) yields $g = \sqrt{\frac{1}{3}\left(27+4\sqrt{6}\right)}-3$ (≈ 0.5022) and $\frac{(g+2)^2}{g^2(1-cg)} \approx 64.49$. Hence, we obtain a factor of approximately $130 + 66\cdot\frac{1}{2^k}$. The calculations for W and H are analogous. This implies the lemma. $\square$

Now, we are ready to state the main theorem of this chapter.

Theorem 7.8. *Algorithm 7.2 computes a constant-factor approximation for disk-packing in polynomial time.*

Proof. The container computed by Algorithm 7.2 is a box with base area $w_{\mathrm{box}}\cdot d_{\mathrm{box}}$ and height

$$\left\lceil\frac{\left\lceil\frac{W}{w_{\mathrm{box}}-\tilde{w}_{\max}}\right\rceil}{\left\lfloor\frac{d_{\mathrm{box}}}{\tilde{d}_{\max}}\right\rfloor}\right\rceil\tilde{h}_{\max}+\left\lceil\frac{\left\lceil\frac{D}{d_{\mathrm{box}}-\tilde{d}_{\max}}\right\rceil}{\left\lfloor\frac{w_{\mathrm{box}}}{\tilde{w}_{\max}}\right\rfloor}\right\rceil\tilde{h}_{\max}+\frac{H}{\left(\left\lfloor\frac{w_{\mathrm{box}}}{\tilde{w}_{\max}}\right\rfloor\left\lfloor\frac{d_{\mathrm{box}}}{\tilde{d}_{\max}}\right\rfloor\right)}+\tilde{h}_{\max} \tag{7.12}$$

(see step 10 in Algorithm 7.2). Define

$$\begin{aligned} t_1 &:= w_{\mathrm{box}}\cdot d_{\mathrm{box}}\left(\left\lceil\frac{\left\lceil\frac{W}{w_{\mathrm{box}}-\tilde{w}_{\max}}\right\rceil}{\left\lfloor\frac{d_{\mathrm{box}}}{\tilde{d}_{\max}}\right\rfloor}\right\rceil\tilde{h}_{\max}\right), \\ t_2 &:= w_{\mathrm{box}}\cdot d_{\mathrm{box}}\left(\left\lceil\frac{\left\lceil\frac{D}{d_{\mathrm{box}}-\tilde{d}_{\max}}\right\rceil}{\left\lfloor\frac{w_{\mathrm{box}}}{\tilde{w}_{\max}}\right\rfloor}\right\rceil\tilde{h}_{\max}\right), \\ t_3 &:= w_{\mathrm{box}}\cdot d_{\mathrm{box}}\left(\frac{H}{\left(\left\lfloor\frac{w_{\mathrm{box}}}{\tilde{w}_{\max}}\right\rfloor\left\lfloor\frac{d_{\mathrm{box}}}{\tilde{d}_{\max}}\right\rfloor\right)}+\tilde{h}_{\max}\right), \end{aligned}$$

i.e., the volume of the container-box is $V = t_1+t_2+t_3$. To estimate the ratio of V to the volume of an optimal container OPT, we consider t_1, t_2, t_3 separately.

$$\begin{aligned}
t_1 &= a\tilde{w}_{\max} \cdot a\tilde{d}_{\max}\left(\left\lceil \frac{\left\lceil \frac{W}{a\tilde{w}_{\max}-\tilde{w}_{\max}} \right\rceil}{\left\lfloor \frac{a\tilde{d}_{\max}}{\tilde{d}_{\max}} \right\rfloor} \right\rceil \tilde{h}_{\max}\right) && \text{(by def. of } w_{\text{box}}, d_{\text{box}}) \\
&= a^2\tilde{w}_{\max}\tilde{d}_{\max}\tilde{h}_{\max}\left\lceil \frac{\left\lceil \frac{W}{(a-1)\tilde{w}_{\max}} \right\rceil}{\lfloor a \rfloor} \right\rceil \\
&\leq a^2\tilde{w}_{\max}\tilde{d}_{\max}\tilde{h}_{\max}\left(\frac{\left(\frac{W}{(a-1)\tilde{w}_{\max}}+1\right)}{(a-1)}+1\right) \\
&\leq a^2\tilde{w}_{\max}\tilde{d}_{\max}\tilde{h}_{\max}\left(\frac{\left(\frac{\left(130+66\cdot\frac{1}{2^k}\right)\mathrm{OPT}}{(a-1)\tilde{w}_{\max}d_{\max}h_{\max}}+1\right)}{(a-1)}+1\right) && \text{(by Lemma 7.7)} \\
&\leq a^2\left(1+\frac{1}{2^k}\right)^3 w_{\max}d_{\max}h_{\max}\left(\frac{\left(\frac{\left(130+66\cdot\frac{1}{2^k}\right)\mathrm{OPT}}{(a-1)w_{\max}d_{\max}h_{\max}}+1\right)}{(a-1)}+1\right)
\end{aligned}$$

by the definition of $\tilde{w}_{\max}, \tilde{d}_{\max}, \tilde{h}_{\max}$. So,

$$\begin{aligned}
t_1 &\leq a^2\left(1+\frac{1}{2^k}\right)^3\left(\frac{130+66\cdot\frac{1}{2^k}}{(a-1)^2}\,\mathrm{OPT}+\left(\frac{1}{a-1}+1\right)w_{\max}d_{\max}h_{\max}\right) \\
&\leq a^2\left(1+\frac{1}{2^k}\right)^3\left(\frac{130+66\cdot\frac{1}{2^k}}{(a-1)^2}+\frac{1}{a-1}+1\right)\mathrm{OPT}, && (7.13)
\end{aligned}$$

since $w_{\max}d_{\max}h_{\max} \leq W_{\mathrm{OPT}}D_{\mathrm{OPT}}H_{\mathrm{OPT}} = \mathrm{OPT}$. Analogoulsy, we get for t_2

$$t_2 \leq a^2\left(1+\frac{1}{2^k}\right)^3\left(\frac{130+66\cdot\frac{1}{2^k}}{(a-1)^2}+\frac{1}{a-1}+1\right)\mathrm{OPT}\,. \qquad (7.14)$$

For t_3, we get

$$\begin{aligned}
t_3 &= a\tilde{w}_{\max} \cdot a\tilde{d}_{\max}\left(\frac{H}{\left(\left\lfloor \frac{a\tilde{w}_{\max}}{\tilde{w}_{\max}} \right\rfloor \left\lfloor \frac{a\tilde{d}_{\max}}{\tilde{d}_{\max}} \right\rfloor\right)}+\tilde{h}_{\max}\right) && \text{(def. of } w_{\text{box}}, d_{\text{box}}) \\
&\leq a^2\tilde{w}_{\max}\tilde{d}_{\max}\left(\frac{H}{(a-1)^2}+\tilde{h}_{\max}\right) \\
&\leq a^2\tilde{w}_{\max}\tilde{d}_{\max}\left(\frac{\left(130+66\cdot\frac{1}{2^k}\right)\cdot\mathrm{OPT}}{(a-1)^2 w_{\max}d_{\max}}+\tilde{h}_{\max}\right) && \text{(by Lemma 7.7)} \\
&\leq a^2\left(1+\frac{1}{2^k}\right)^2 w_{\max}d_{\max}\left(\frac{\left(130+66\cdot\frac{1}{2^k}\right)\cdot\mathrm{OPT}}{(a-1)^2 w_{\max}d_{\max}}+\left(1+\frac{1}{2^k}\right)h_{\max}\right)
\end{aligned}$$

by the definitions of $\tilde{w}_{\max}, \tilde{d}_{\max}, \tilde{h}_{\max}$. So,

$$t_3 \le a^2 \left(1 + \frac{1}{2^k}\right)^3 \left(\frac{130 + 66 \cdot \frac{1}{2^k}}{(a-1)^2} + 1\right) \mathrm{OPT}\,. \tag{7.15}$$

Eqs. (7.13) to (7.15) immediately give the following estimation for the volume V of the container computed by Algorithm 7.2:

$$V \le a^2 \left(1 + \frac{1}{2^k}\right)^3 \left(\frac{3 \cdot \left(130 + 66 \cdot \frac{1}{2^k}\right)}{(a-1)^2} + \frac{2}{a-1} + 3\right) \mathrm{OPT}\,.$$

Obviously, the factor of OPT is monotonically decreasing if k increases (at a cost in running time as we will see afterwards). Hence, we can chose a value for k, for example $k = 7$ and optimize for a afterwards by computing the derivative and its real root. For $k = 7$ we get the root at approximately $a \approx 5.9852$ and an approximation factor of approximately 692.

It remains to analyze the running time of Algorithm 7.2. Step 1 runs in polynomial time since each disk gets assigned to the axis corresponding to the component with largest absolute value in its normal vector. The calls of Algorithm 7.1 in steps 2 and 3 run in polynomial time by Theorem 7.6. The computation of the approximate distance of two disks in the stabbing can be done in time polynomial in the input and k with techniques from [14]. Note that the extent of a disk D in x-, y-, or z-direction is $\mathrm{h}_{(1,0,0)}((1,0,0), D), \mathrm{h}_{(0,1,0)}((0,1,0), D), \mathrm{h}_{(0,0,1)}((0,0,1), D)$ respectively which can be approximated in the same way. These approximate values can also be used to obtain the division into the smaller boxes in Steps 4, 7 and 8. So, Algorithm 7.2 can be implemented with polynomial running time. □

7.2 Other objects

Observe that our approximation algorithm can be extended to any arbitrary fixed planar shape A, provided that A can be enclosed by some disk D (i.e., is bounded) and contains some disk d (i.e., it has nonempty interior). More precisely, if we are given a finite set of congruent copies of A in three dimensions we can approximate the smallest axis-parallel box into which it can be packed by translations.

This can be done by just applying our algorithm to the corresponding set of copies of D. Since it gives a constant-factor approximation of the optimal packing of the D's it also gives an approximation of the optimal packing of the d's. Observe however, that the approximation factor is multiplied by r^3 where r is the ratio between the radii of D and d. Since the optimal packing of the A's provides some packing of the d's its container must be at least as large from which we obtain an approximation for the A's.

Notice however, that the approximation factor obtained this way depends on the shape of A. For standard shapes such as squares ($r = \sqrt{2}$), equilateral triangles ($r = 2$) etc., we can directly compute it from the approximation factor of Algorithm 7.2.

7.3 Unbounded containers are necessary

At first glance, the question may seem strange whether all, uncountably many, unit disks in three-space can be packed into a finite volume container. However, in dimension two, obviously all unit length line segments can be packed nonoverlapping into a rectangle of area 2, as Fig. 7.11 shows (There are even smaller containers). Observe that no two distinct segments intersect in interior points.

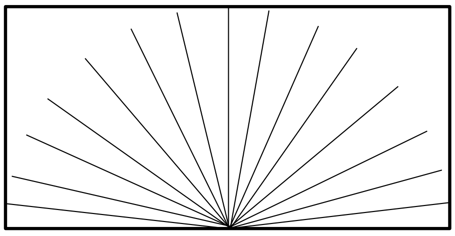


Figure 7.11: How to pack all unit length segments into a container of area 2.

However, as a corollary of our previous results we will conclude that there is no bounded size container into which all unit disks can be packed. More precisely, we will show that even for a subset of all disks there is no bounded size container into which all unit disks from that subset can be packed. To do so, we need the following lemma.

Lemma 7.9. *Let D_1 and D_2 be two disks whose normal vectors form an angle ξ smaller than $\frac{\pi}{2}$. Then their distance $h_{\mathbf{s}}(D_1, D_2)$ when stabbed by a line in direction $\mathbf{s}$ is at least $\sin \xi$ for any $\mathbf{s} \in S^2$.*

Proof. Let $\mathbf{c_1}$ and $\mathbf{c_2}$ be the positions of the two disk centers on the line. Consider the shortest path P from $\mathbf{c_1}$ to $\mathbf{c_2}$ on the planes E_1 containing D_1 and E_2 containing D_2. It is easy to see that P contains only one bend on the intersection line l of E_1 and E_2. We refer to this point by $\mathbf{b}$ (see Fig. 7.12). Observe that $\mathbf{c_1}$ or $\mathbf{c_2}$ must have distance at least 1 to $\mathbf{b}$ since otherwise D_1 and D_2 would intersect in $\mathbf{b}$. Furthermore, P forms an angle η of at least ξ at $\mathbf{b}$. To see that, suppose $\mathbf{c_1}$ is fixed and $\mathbf{c_2}$ can move parallel to l inside E_2. Observe that the smallest angle P can form in this way is ξ.

Now, we consider the triangle formed by $\mathbf{c_1}$, $\mathbf{c_2}$, and $\mathbf{b}$, so $|\overline{\mathbf{c_1 c_2}}| = h_s(D_1, D_2))$. Let without loss of generality the distance $|\overline{\mathbf{c_1 b}}|$ of $\mathbf{c_1}$ to $\mathbf{b}$ be at least 1. Then, within the plane through $\mathbf{c_1}, \mathbf{c_2}$, and $\mathbf{b}$ we have the situation shown in Fig. 7.12. Consider the ray in this plane emanating from $\mathbf{b}$ that has an angle of ξ with the line segment $\overline{\mathbf{c_1 b}}$ which hits the line segment $\overline{\mathbf{c_1 c_2}}$ in some point $\mathbf{m}$ since $\eta \geq \xi$. By the law of sines, we have

$$\frac{|\overline{\mathbf{c_1 b}}|}{\sin \chi} = \frac{|\overline{\mathbf{c_1 m}}|}{\sin \xi}.$$

Hence,

$$h_s(D_1, D_2) = |\overline{\mathbf{c_1 c_2}}| \geq |\overline{\mathbf{c_1 m}}| = \frac{|\overline{\mathbf{c_1 b}}| \cdot \sin \xi}{\sin \chi} \geq \sin \xi,$$

where the last inequality holds since $|\overline{\mathbf{c_1 b}}| \geq 1$ and $\sin\chi \leq 1$. □

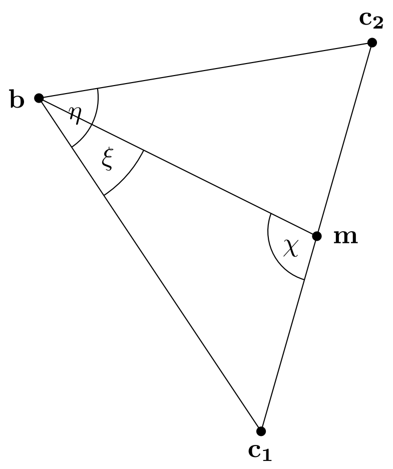


Figure 7.12: Triangle formed by $\mathbf{c_1}$, $\mathbf{c_2}$, and $\mathbf{b}$.

Theorem 7.10. *Packing a set of n unit disks requires a container of volume $\Omega(\sqrt{n})$ in the worst case.*

Proof. In the following, we will show that $\Omega(\sqrt{n})$ is a lower bound for the volume of the container constructed by Algorithm 7.2 which is within a constant factor of the optimal container. From that the theorem follows immediately.

We identify every unit disk with its normal vector in the upper half of a unit sphere S^2 centered at the origin. Observe that for every normal vector in the lower half (negative z-coordinate) of the unit sphere there is a vector in the upper half corresponding to the same disk.

Consider the projection parallel to the z-axis of a $c \times c$-square partitioned into a square grid of grid cells with side length ε in the xy-plane centered at the origin onto the upper half sphere, see Fig. 7.13. Choose the constant c to be sufficiently small so that all points contained in the projection correspond to disks contained in set $\mathcal{Z}$ in Algorithm 7.2 and their normal vectors pairwise form an angle of at most $\pi/2$. Note that the grid contains $n = \Omega(1/\varepsilon^2)$ points.

For any two grid-points $\mathbf{p_1}, \mathbf{p_2}$ corresponding to disks D_1 and D_2 it holds that $h_s(D_1, D_2) \geq \sin\xi$ with $\mathbf{s} = (0, 0, 1)$ and ξ is the angle between the two normal vectors by Lemma 7.9.

By construction, see Fig. 7.14, the projected points $\mathbf{p_1'}, \mathbf{p_2'} \in S^2$ have Euclidean distance $2 \cdot \sin(\xi/2)$ which is at least ε. So we have

$$\varepsilon \leq 2 \sin\frac{\xi}{2} \leq 2 \sin\xi.$$

So by Lemma 7.9, for any two disks D_1, D_2 corresponding to grid-points in the $c \times c$-square we have $h_s(D_1, D_2) \geq \varepsilon/2$. Therefore, no matter in which order these disks are

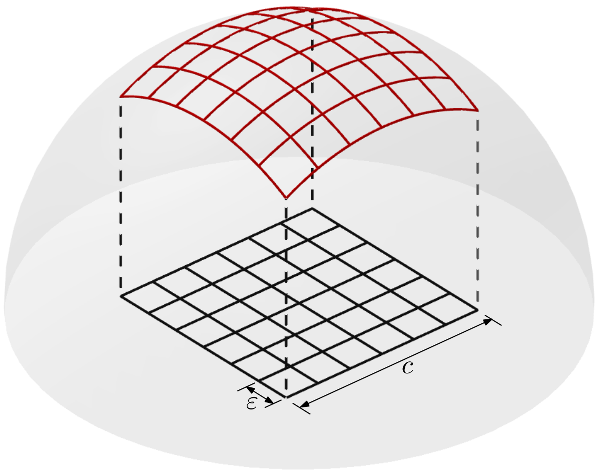


Figure 7.13: Projecting a grid onto the unit sphere.

stabbed they will occupy a segment of length $\Omega((1/\varepsilon^2)\cdot\varepsilon)$, i.e., $\Omega(1/\varepsilon)$ of the stabbing line which is $\Omega(\sqrt{n})$ since $n=\Omega(1/\varepsilon^2)$. From Lemma 7.7 it follows that this is also a lower bound for the volume of a container computed by Algorithm 7.2, and, since that is within a constant factor of the optimum, of any container for the set of disks. $\square$

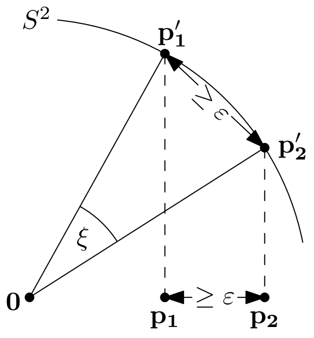


Figure 7.14: Two grid points $\mathbf{p_1},\mathbf{p_2}$ and their projection onto S^2 with center $\mathbf{0}$.

From Theorem 7.10, we obtain immediately the following corollary.

Corollary 7.11. *There is no bounded size container into which all unit disks can be packed.*

7.4 Conclusion and Open problems

Summarizing, we showed that there are constant factor approximation algorithm for DiskPacking and DiskStabbing and how to generalize these algorithms. Even though the approximation factor of these algorithms are forbiddingly high, it is interesting that they can be approximated within a constant factor from a theoretical point of view, since it is neither known if these problems are NP-hard nor in NP. It remains an open problem whether an optimal packing of disks of different radii can be efficiently approximated.

In particular, approximating the packing of arbitrarily oriented boxes or convex polyhedra seems to be much more difficult.

We showed further that not all unit disks in 3D can be packed into a constant size convex container which is in contrast to the 2D-case, where all unit line segments can easily be packed into a rectangle with area two. It is an interesting question, if this translates to higher dimensions.

II

Complexity of Packing

Chapter 8

No FPTAS for Packing Polygons

In this chapter, we discuss the inapproximability of packing polygons into minimum-area convex polygonal containers. From Theorem 2.6, we know that the decision problem whether a given set of polygons can be packed into a given polygonal container using only translations is in NP. By a simple reduction from PARTITION, it can be shown that this problem is NP-hard (see e.g. [5]), hence, it is NP-complete. In the following, we ask about the approximability of these kind of problems. There is a constant factor approximation by Alt, de Berg, and Knauer for packing convex polygons into a minimum area axis-parallel rectangle or convex container under translation[6]. It is known by a reduction form PARTITION almost identical to the one for NP-hardness, that STRIP-PACKING and BIN-PACKING cannot be approximated better than with a factor of $\frac{3}{2}$ unless $\mathsf{P} = \mathsf{NP}$. This means under the assumption that $\mathsf{P} \neq \mathsf{NP}$, that there is no PTAS for minimizing the area of a rectangular container for a given set of polygons when the width of the container is fixed. It is not clear if this also holds if the width of the container is not fixed. In the following, we will show a weaker result, i.e., that there is no FPTAS for minimizing the area of a convex container unless $\mathsf{P} = \mathsf{NP}$. In more detail, we study the following set of problems.

Definition 8.1 (*(o,m,c)*-minimum area container packing)**.** *An instance of the problem of* (o,m,c)-minimum area container packing *is a set of convex polygons of type* o *(e.g. rectangles or general polygons). The aim is to pack these polygons non-overlappingly into a convex container of type* c *(e.g. axis-parallel rectangles or convex polygons) of minimum area using only motions of type* m *(e.g. translations or rigid motions).*

We will first show that there is no FPTAS for packing axis-parallel rectangles under translation into an axis-parallel rectangle of minimum area ((axis-parallel rectangle, translation, axis-parallel rectangle)-minimum area container packing) unless $\mathsf{P} = \mathsf{NP}$.

This implies directly that the same holds for packing arbitrary polygons into an axis-parallel rectangle under translation ((polygon, tranlsation, axis-parallel rectanlge)-minimum area container packing). The proof will be by a reduction from the well studied strongly NP-hard problem 3-PARTITION.

Definition 8.2 (3-PARTITION)**.** *An instance of the 3-PARTITION-problem is a multiset $\mathcal{A}$ of $3n$ positive integers where the following holds: $\frac{B}{4} < a < \frac{B}{2}$ for all $a \in \mathcal{A}$ with $B = \frac{1}{n}\sum_{a\in\mathcal{A}} a$. $\mathcal{A}$ is in 3-PARTITION if and only if $\mathcal{A}$ can be partitioned into triplets $\{a_{j_1}, a_{j_2}, a_{j_3}\}$ with $a_{j_1} + a_{j_2} + a_{j_3} = B$.*

Observation 8.3. *We can assume without loss of generality that $B > 3$ since $a \geq 1$ for all $a \in \mathcal{A}$ and $B = 3$ is trivial.*

In Section 8.1, we give the reduction from 3-PARTITION and use it to prove the following theorem.

Theorem 8.4. *There can not be an FPTAS for packing axis-parallel rectangles under translation into a minimum-area rectangle unless $\mathsf{P} = \mathsf{NP}$.*

We will use the term YES-instance simultaneously for an instance $\mathcal{A} \in$ 3-PARTITION and the set of objects obtained by the reduction from $\mathcal{A}$. Analogously, we will use NO-instance. It should be clear from the context if we refer to a 3-PARTITION-instance or the set of objects obtained from a 3-PARTITION-instance by the reduction.

In Section 8.2 we will turn to other variants of the problem.

8.1 No FPTAS for Packing Rectangles under Translation into Minimum-Area Axis-Parallel Rectangles

Algorithm 8.1 computes from an instance of 3-PARTITION a set of rectangles that can be used as an instance for the (axis-parallel rectangle, translation, axis-parallel rectangle)-minimum area container packing problem. If an instance $\mathcal{A}$ is in 3-PARTITION, then the rectangles computed by Algorithm 8.1 on input $\mathcal{A}$ can be packed into a container with height $(n+1)B+n$ and width B (see Fig. 8.1i). Observe that the area of the container equals the total area of the rectangles packed, i.e. there are no parts of the container that are not covered by rectangles packed.

Now, consider an instance $\mathcal{A} \notin$ 3-PARTITION and let $\mathcal{R}$ be the output of Algorithm 8.1 on input $\mathcal{A}$. In any packing, all rectangles in $\mathcal{R}$ can be moved to the left until they either touch the boundary of the container or another rectangle and downwards in the same way and this does not increase the area of the container. Therefore, and since all rectangles have integer width and height, an optimal container for the rectangles in $\mathcal{R}$ has integer width and height. Observe that $\sum_{a\in\mathcal{A}'} a = B$ for $\mathcal{A}' \subseteq \mathcal{A}$ is only possible if $|\mathcal{A}'| = 3$ since $\frac{B}{4} < a < \frac{B}{2}$ by Definition 8.2. Since $\mathcal{R}$ was computed from an instance

Algorithm 8.1:

```
    Input: Multiset of $3n$ positive integers $\mathcal{A} = \{a_1, a_2, \dots, a_{3n}\}$
    Output: Multiset of $3n+1$ rectangles $\mathcal{R} = \{r_1, r_2, \dots, r_{3n+1}\}$ given by their
            height and width $r_i = (h_i, w_i)$
1   $B \leftarrow \frac{1}{n} \sum_{a \in \mathcal{A}} a$;
2   foreach $a_i \in \mathcal{A}$ do
3   |   $r_i \leftarrow (1, a_i)$;
4   end
5   $r_{3n+1} \leftarrow ((n+1)B, B)$;
6   return $\{r_1, r_2, \dots, r_{3n+1}\}$
```

$\mathcal{A} \notin$ 3-Partition, this implies that the rectangles in $\mathcal{R} \setminus \{r_{3n+1}\}$ cannot be partitioned into subsets such that the total sum of the width of rectangles in each subset equals B. Therefore, the rectangles in $\mathcal{R}$ cannot be packed above (or below) r_{3n+1} into a container with width B and height $(n+1)B+n$ like a YES-instance. Hence, any container has to be wider than B or higher than $(n+1)B+n$. So, for an instance $\mathcal{A}$ not in 3-Partition, the area of any container for $\mathcal{R}$ is at least

$$\min\{(n+1)B \cdot (B+1), ((n+1)B+n+1)B\} = (n+1)(B+1)B,$$

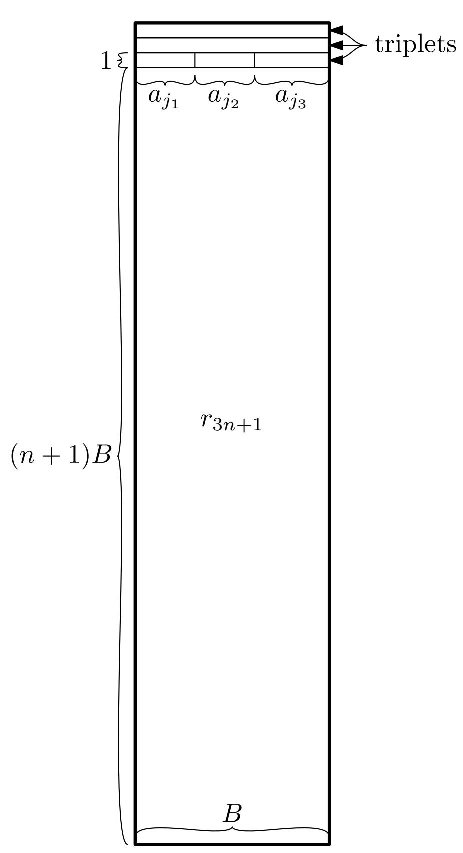
since r_{3n+1} still needs to be packed and the width and height of an optimal container for the instance are integers as observed earlier (see Fig. 8.1ii for illustration). Note, that this is an alternative proof for the NP-completeness of packing rectangles into the minimum-area axis-parallel rectangle under translation. Since 3-Partition is strongly NP-hard and the integers generated in the reduction are polynomially bounded from above by the maximum integer in the given 3-Partition-instance and the total size of the instance, this reduction shows indeed that packing rectangles into the minimum-area axis-parallel rectangle is strongly NP-hard. It follows from [28] that there cannot be an FPTAS for packing rectangles into the minimum-area axis-parallel rectangle under the assumption $\mathsf{P} \neq \mathsf{NP}$.

8.2 Other Variants of the Problem

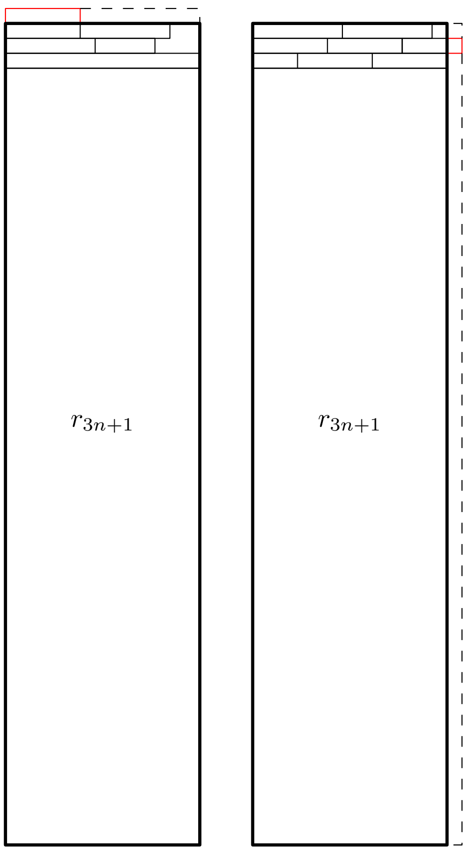
In the following, we will generalize the idea from above to show that there cannot exist an FPTAS for other variants of minimum area container packing unless $\mathsf{P} = \mathsf{NP}$. First, we make the crucial observation for our proofs:

Observation 8.5. *Given a container and a packing of axis-parallel rectangles into it under translation or rigid motions. If the container is convex and not a rectangle, its area is larger than the sum of the areas of the rectangles packed.*

First, we use this observation to show that there cannot be an FPTAS for packing rectangles under translation into a minimum-area convex container unless $\mathsf{P} = \mathsf{NP}$ and afterwards turn to packing under rigid motions.



(i) Optimal packing for $\mathcal{A} \in$ 3-PARTITION.



(ii) In a packing for $\mathcal{A} \notin$ 3-PARTITION the container needs to be wider or higher than the optimal container (bold rectangle).

Figure 8.1: Packing examples with $n = 3$

8.2.1 No FPTAS for Packing Rectangles under Translation into Minimum-Area Convex Container

Consider the rectangles obtained from a 3-Partition-instance by Algorithm 8.1 that are to be packed under translations into a minimum area convex container ((rectangle, translation, polygon)-minimum area container packing). The area of the optimal container for a YES-instance is as before the sum of the areas of the rectangles packed. Hence, if the container is not a rectangle, the given instance is a NO-instance by Observation 8.5. If the container is a rectangle, we can decide by its area if the given instance is a YES- or a NO-instance analogously to the proof in Section 8.1. Therefore, Algorithm 8.1 is a reduction from 3-Partition to (rectangle, translation, polygon)-minimum area container packing preserving strong NP-hardness as before. Again, this implies that there cannot be an FPTAS for (rectangle, translation, polygon)-minimum area container packing unless P = NP. Observe that this directly generalizes to (polygon, translation, polygon)-minimum area container packing.

8.2.2 No FPTAS for Packing Rectangles under Rigid Motions

First, we consider packing rectangles under rigid motions into a rectangle. Observe that since we allow to rotate the rectangles, we can assume without loss of generality that the container rectangle is axis-parallel. We will use a reduction from 3-Partition similar to Algorithm 8.1 to show that (rectangle, rigid motion, rectangle)-minimum area container packing is stronlgy NP-hard. The following observation will simplify the proof.

Observation 8.6. *Given an axis-parallel container rectangle and a set of rectangles packed into it under rigid motions. If there is at least one rectangle packed such that it is not axis-parallel, then the area of the container rectangle is larger than the sum of the areas of the packed rectangles.*

Consider a 3-Partition-instance. Intuitively, Observation 8.6 tells us that we can restrict the rectangles to be rotated not at all or by 90°. We adapt the reduction given in Algorithm 8.1 in such a way that rotating rectangles corresponding to integers in the given 3-Partition-instance and packing them above r_{3n+1} will result always in a container with height greater than the height of an optimal container. The idea is that rotation does not help to fit a NO-instance into the optimal container. In detail, the reduction works as follows: For every integer a in the given 3-Partition instance, we generate a rectangle with width $n \cdot a$ instead of a and height 1 as before. The extra rectangle defined in Line 5 in Algorithm 8.1 gets width nB and height $n(n+1)B$. Note that the dimensions of the rectangles are still polynomial in the size of the integers in the given 3-Partition-instance as before. The area of an optimal container for a YES-instance is as before the sum of the areas of the packed rectangles: $nB \cdot n((n+1)B+1)$. Observe that in such a packing no rectangle is rotated. Consider now an optimal packing of rectangles obtained by the described reduction from a 3-Partition-instance. If a rectangle is rotated other than by 90°, the area of the container is larger than the sum of the areas of the rectangles packed by Observation 8.6 and therefore we can tell from the container area that the given instance is a NO-instance. Therefore, we assume in the following that all rectangles are either not rotated at all or by 90°. Furthermore, we assume without loss of generality that r_{3n+1} is not rotated. Let $a_{\min} = \min(\mathcal{A})$. As soon as we rotate one rectangle and place it above the extra rectangle r_{3n+1} (see Fig. 8.2ii), the container has height at least

$$n(n+1)B + n \cdot a_{\min} > n(n+1)B + n$$

since $a_{\min}$ is greater than one by Observation 8.3 and Definition 8.2. So, as before, the container for a NO-instance has to be wider or higher than an optimal container for a YES-instance. Hence, the container for a NO-instance has area at least

$$\begin{aligned}&\min(nB(n(n+1)B+n+1), (nB+1)n(n+1)B)\\ &= nB(n(n+1)B+n+1).\end{aligned}$$

which is larger than the area of an optimal container for a YES-instance. Therefore, the reduction described before shows that (rectangle, rigid motion, rectangle)-minimum area container packing is strongly NP-hard and hence, there cannot be an FPTAS for it unless $\mathsf{P} = \mathsf{NP}$.

Analogously to the proof in Section 8.2.1, this proof generalizes to packing rectangles under rigid motions into convex polygons and hence, (rectangle, rigid motions, polygon)-minimum area container packing can also not have an FPTAS unless $\mathsf{P} = \mathsf{NP}$.

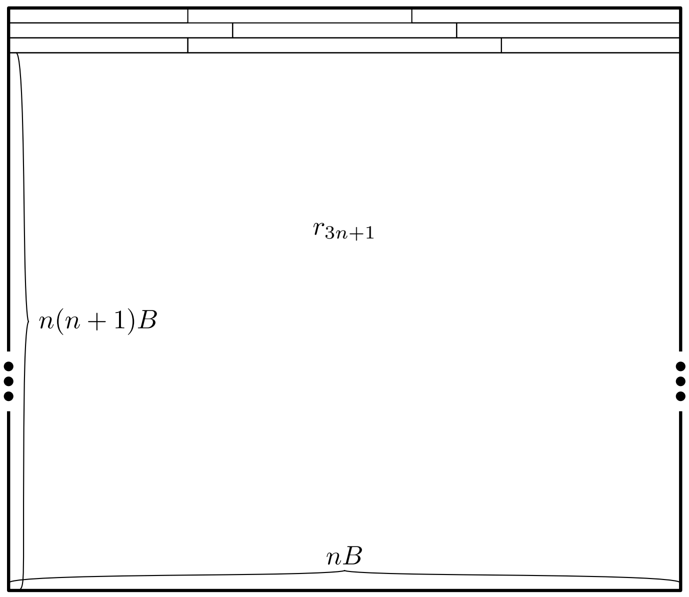
8.3 Conclusion

Summarizing, we showed the following theorem.

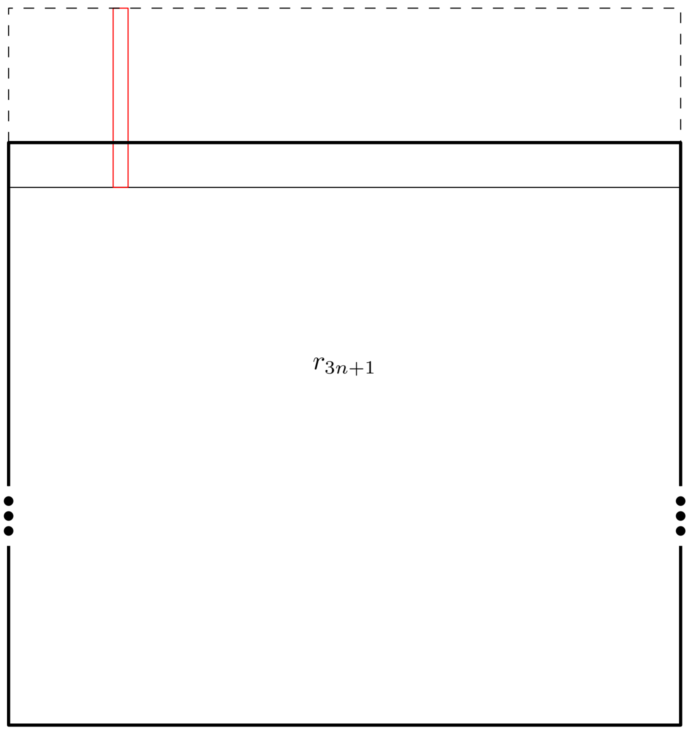
Theorem 8.7. *Under the assumption* $\mathsf{P} \neq \mathsf{NP}$ *there cannot be an FPTAS for minimizing*

- *the area of a rectangular container*
  - *for packing a given set of axis-parallel rectangles under translations, or*
  - *for packing a given set of rectangles under rigid motions,*
- *nor the area of a convex container*
  - *for packing a given set of axis-parallel rectangles under translations, or*
  - *for packing a given set of rectangles under rigid motions.*

The theorem directly generalizes to packing general polygons instead of rectangles. It would be interesting to see if there can be a PTAS for this kind of problem or if there cannot be a PTAS like for Strip-Packing and Bin-Packing.



(i) The Optimal packing for $\mathcal{A} \in$ 3-PARTITION looks similar as without rotations



(ii) In a packing for $\mathcal{A} \notin$ 3-PARTITION, rotation does not help.

Figure 8.2: Packing examples with $n = 3$ for packing under translation and rotation by 90°.

Chapter 9

Line Segment Packing is NP-hard

In this chapter we discuss the complexity of line segment packing, i.e., given a set of line segments, can they be packed into a given container under translation. Note that this problem becomes trivial for polygonal containers when allowing rotations: one only needs to check if the longest line segment can be packed by computing the diameter of the container. The other segments can then be packed parallel and arbitrarily close to the longest segment.

Related Work The most famous problem related to packing line segments might be the Kakeya Problem introduced in 1917 [37]: What is the smallest area convex set such that a unit line segment can be rotated in it continuously by 360°? Pál showed that the solution is an equilateral triangle with area $1/\sqrt{3}$ [43]. Ahn et al. studied a generalization of this problem, i.e., given a set of line segments, what is the smallest area convex set that contains a translate of each input line segment? They show that there is always an optimal set that is a triangle and give an $\mathcal{O}(n \log n)$ time algorithm to compute such a triangle [4]. We can reformulate this result in the following way: Given a set of line segments, a smallest area container to pack the line segments under translation is a triangle that can be computed in $\mathcal{O}(n \log n)$ when we allow the line segments to intersect in their interior. When intersections are not allowed, the problem of packing line segments becomes hard: Dobbins and Kim showed that it is NP-hard to find the maximum number of line segments from a given set of input line segments that can be packed into a convex polytope in three dimensions[20]. Kim and Miltzow showed that the same holds for two dimensions and simple polygonal containers [40]. In the following, we will show that this also holds for square containers under certain assumptions.

In detail, we will show that the following problem is NP-hard.

Definition 9.1 (line segment packing)**.** *Given a multiset of open line segments and a closed* *square container, can the line segments be packed nonoverlappingly into the container under translation?*

Observe that, since the line segments are open, they are allowed to touch at their endpoints and an endpoint can lie on the interior of another line segment. Since the container is closed, line segments are allowed to lie on the boundary of the container. In particular, we do not allow to pack parallel line segments such that they overlap. Our proof will heavily rely on these properties. This is in contrast to the constructions by Dobbins and Kim, and Kim and Miltzow where two line segments only touch at their endpoints and no two line segments have the same slope such that line segments with same slope could be identified with each other by definition. Nevertheless, our version of the problem is in accordance to packing higher dimensional objects where the interior of the objects is not allowed to overlap.

In Section 9.1, we will prove the following theorem by a reduction from the well known problem PARTITION i.e., given a set of positive integers, can they be partitioned into two subsets such that the sum of both subsets is equal.

Theorem 9.2. Line segment packing *is* NP-*hard.*

9.1 Packing Line Segments into a Square is NP-hard

First, we describe a reduction from PARTITION to line segment packing as defined in Definition 9.1. Afterwards, we use it to prove Theorem 9.2.

9.1.1 Reduction from PARTITION

Given a multiset S of positive integers, we construct a multiset L of line segments and a side length a such that L can be packed into a square with side length a if and only if S can be partitioned into two subsets such that the sum of both subsets is equal. The idea is to construct a vertical line segment for each integer in S and force them with diagonal line segments to be packed on the vertical boundaries of the container square (see Fig. 9.1 for an example). Hence, every vertical line segment should be at least as long as the difference between the endpoints of diagonal line segments in Fig. 9.1ii and there is only a polynomial number of diagonal line segments. One way to achieve this is by ensuring that the factor between the lengths of the shortest and longest vertical segment is bounded by a constant. We implement this by creating for each $s_i \in \mathcal{S}$ a line segment of length $s_i + s_\Sigma$ where $s_\Sigma = \sum_{1 \leq i \leq n} s_i$. To ensure that a possible partition of $\mathcal{S}$ is reflected by a packing of the line segments on the two different vertical boundaries of the container square, we introduce n extra vertical line segments of length s_Σ such that on both vertical boundaries n line segments need to be packed in order to pack all the line segments. The reduction in more detail is described in Algorithm 9.1

Algorithm 9.1:

```
Input: Multiset of $n$ integers $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$
Output: Multiset of open line segments $\mathcal{L}$ given by their endpoints and an
        integer $a$
1  $s_\Sigma \leftarrow \sum_{1 \le i \le n} s_i$;
2  $a \leftarrow \frac{1}{2} s_\Sigma + n s_\Sigma$;
3  for $i \leftarrow 1$ to $n$ do
4      $l_i \leftarrow (0,0), (0, s_i + s_\Sigma)$;
5  end
6  for $i \leftarrow n+1$ to $2n$ do
7      $l_i \leftarrow (0,0), (0, s_\Sigma)$;
8  end
9  $k \leftarrow \left\lfloor \frac{a}{s_\Sigma} \right\rfloor$;
10 for $i \leftarrow 1$ to $k$ do
11     $l_{2n+i} \leftarrow (0,0), (a, i \cdot s_\Sigma)$;
12     $l_{2n+k+i} \leftarrow (0,0), (a, i \cdot s_\Sigma)$;
13 end
14 $l_{2n+2k+1} \leftarrow (0,0), (a,a)$;
15 $\mathcal{L} \leftarrow \{l_1, l_2, \ldots, l_{2n+2k+1}\}$;
16 return $\mathcal{L}, a$
```

9.1.2 Proof of Theorem 9.2

We will show that the multiset of line segments $\mathcal{L}$ created by Algorithm 9.1 from a set $\mathcal{S}$ of positive integers can be packed into a square with side length a under translation if and only if $\mathcal{S}$ can be partitioned into two subsets $\mathcal{S}_1, \mathcal{S}_2$ with $\sum_{s' \in \mathcal{S}_1} s' = \sum_{s'' \in \mathcal{S}_2} s''$. For ease of notation, we call the line segments $\mathcal{L}_o = \{l_1, \ldots l_n\}$ object line segments, $\mathcal{L}_v = \{l_1, \ldots, l_{2n}\}$ vertical line segments, and $\mathcal{L}_d = \mathcal{L} \setminus \mathcal{L}_v$ diagonal line segments.

First, we prove that if $\mathcal{S}$ can be partitioned into two subsets $\mathcal{S}_1, \mathcal{S}_2$ with $\sum_{s' \in \mathcal{S}_1} s' = \sum_{s'' \in \mathcal{S}_2} s''$, then $\mathcal{L}$ can be packed into a square of side length a. We start by placing the diagonal line segments like in Fig. 9.1ii, i.e., $l_{2n+2k+1}$ on the diagonal from bottom left to top right of the container square, and from each of the remaining diagonal line segment pairs l_{2n+i}, l_{2n+k+i} one with one endpoint in the lower left corner, the other with one endpoint at the upper right corner . It remains to show that $\mathcal{L}_v$ can be packed into the container square afterwards. Since $\mathcal{S}_1, \mathcal{S}_2$ is a valid partition, we have

$$\sum_{s' \in \mathcal{S}_1} s' = \sum_{s'' \in \mathcal{S}_2} s'' \qquad = \frac{1}{2} \sum_{s \in \mathcal{S}} s.$$

So,

$$\sum_{s' \in \mathcal{S}_1} s' + n s_\Sigma = \sum_{s'' \in \mathcal{S}_2} s'' + n s_\Sigma \qquad = \frac{1}{2} \sum_{s \in \mathcal{S}} s + n s_\Sigma,$$

and rearranging gives

$$\sum_{s' \in \mathcal{S}_1} (s' + s_\Sigma) + (n - |\mathcal{S}_1|)s_\Sigma = \sum_{s'' \in \mathcal{S}_2} (s'' + s_\Sigma) + (n - |\mathcal{S}_2|)s_\Sigma \qquad = a,$$

by the definition of a. Let $\mathcal{L}_1 \subseteq \mathcal{L}_o$ be the set of line segments generated from integers in $\mathcal{S}_1$. Since $\mathcal{S}_1, \mathcal{S}_2$ is a partition, we get

$$\sum_{l' \in \mathcal{L}_1} |l'| + (n - |\mathcal{S}_1|)s_\Sigma = \sum_{l'' \in \mathcal{L}_o \setminus \mathcal{L}_1} |l''| + (n - |\mathcal{S}_2|)s_\Sigma \qquad = a,$$

where $|l|$ denotes the length of the line segment l. Observe that $n - |\mathcal{S}_2| = |\mathcal{S}_1|$ since $\mathcal{S}_1, \mathcal{S}_2$ is a partition. Hence,

$$\sum_{l' \in \mathcal{L}_1} |l'| + \sum_{i=1}^{n-|\mathcal{S}_1|} |l_{n+i}| = \sum_{l'' \in \mathcal{L}_o \setminus \mathcal{L}_1} |l''| + \sum_{i=n-|\mathcal{S}_1|+1}^{n} |l_{n+i}| \qquad = a,$$

since the length of a line segment in $\{l_{n+1}, \ldots l_{2n}\}$ is precisely s_Σ.

That means that we can place the line segments in $\mathcal{L}_1$ and $n - |\mathcal{S}_1|$ line segments in $\mathcal{L}_v \setminus \mathcal{L}_o$ on top of each other on the left vertical boundary of the container square and the remaining line segments in $\mathcal{L}_v$, i.e., the line segments in $\mathcal{L}_o \setminus \mathcal{L}_1$ and the remaining $|\mathcal{S}_1|$ line segments in $\mathcal{L}_v \setminus \mathcal{L}_o$, on top of each other on the right boundary of the container square as in Fig. 9.1ii. So, all line segments are packed into the container square.

Now, we turn to showing that if all line segments in $\mathcal{L}$ can be packed into the container square, then $\mathcal{S}$ can be partitioned into two subsets $\mathcal{S}_1, \mathcal{S}_2$ with $\sum_{s' \in \mathcal{S}_1} s' = \sum_{s'' \in \mathcal{S}_2} s''$. Observe that $l_{2n+2k+1}$ can only be packed on the diagonal of the container square from bottom left to top right. Afterwards, the remaining diagonal line segments have to be placed with one endpoint either in the top right or the bottom left corner. Since the diagonal line segments l_{2n+i} and l_{2n+k+i} are identical for $1 \leq i \leq k$ and we do not allow the segments to intersect in their interior, one of them is placed with one endpoint in the lower left corner and the other with one endpoint in the top right corner. Observe that the endpoints not placed at a corner lie either on the right or left vertical boundary of the container square. Now, consider the endpoints of the diagonal line segments in the partial packing just described on the left vertical boundary of the container square in increasing order by their y-coordinates. The distance between the bottom left corner and the first endpoint, the distance between two consecutive endpoints, and the distance between the last endpoint and the top left corner of the container square is at most s_Σ by construction. The same holds for the right vertical boundary of the container square. Since the vertical line segments have length at least s_Σ, this implies that they have to be packed onto the vertical boundaries of the container square, i.e., the packing has to look like in Fig. 9.1ii. Hence, $\mathcal{L}_v$ can be partitioned into two subsets $\mathcal{L}_1, \mathcal{L}_2$ with $\sum_{l' \in \mathcal{L}_1} |l'| \leq a$ and $\sum_{l'' \in \mathcal{L}_2} |l''| \leq a$. Since $\sum_{l \in \mathcal{L}_v} |l| = 2a$, the following holds

$$\sum_{l' \in \mathcal{L}_1} |l'| = a = \sum_{l'' \in \mathcal{L}_2} |l''|.$$

Since all line segments in $\mathcal{L}_v \setminus \mathcal{L}_o$ have length s_Σ,

$$\sum_{l' \in \mathcal{L}_1 \cap \mathcal{L}_o} |l'| + |\mathcal{L}_1 \setminus \mathcal{L}_o| \cdot s_\Sigma = a = \sum_{l'' \in \mathcal{L}_2 \cap \mathcal{L}_o} |l'| + |\mathcal{L}_2 \setminus \mathcal{L}_o| \cdot s_\Sigma.$$

Let $\mathcal{S}_1$ be the set of integers the line segments in $\mathcal{L}_1 \cap \mathcal{L}_o$ have been created from and alogoulsy let $\mathcal{S}_2$ be the set of integers the line segments in $\mathcal{L}_2 \cap \mathcal{L}_o$ have been created from. Observe that, since $\mathcal{L}_1 \cap \mathcal{L}_o, \mathcal{L}_2 \cap \mathcal{L}_o$ is a partition of $\mathcal{L}_o$, $\mathcal{S}_1, \mathcal{S}_2$ is a partition of $\mathcal{S}$. We get

$$\begin{aligned} &\left(\sum_{s' \in \mathcal{S}_1} s' \right) + |\mathcal{L}_1 \cap \mathcal{L}_o| \cdot s_\Sigma + |\mathcal{L}_1 \setminus \mathcal{L}_o| \cdot s_\Sigma = a \\ &\qquad\qquad = \left(\sum_{s'' \in \mathcal{S}_2} s'' \right) + |\mathcal{L}_2 \cap \mathcal{L}_o| \cdot s_\Sigma + |\mathcal{L}_2 \setminus \mathcal{L}_o| \cdot s_\Sigma, \end{aligned}$$

which is equivalent to

$$\left(\sum_{s' \in \mathcal{S}_1} s' \right) + |\mathcal{L}_1| \cdot s_\Sigma = a = \left(\sum_{s'' \in \mathcal{S}_2} s'' \right) + |\mathcal{L}_2| \cdot s_\Sigma.$$

By the definition of a, we have

$$\left(\sum_{s' \in \mathcal{S}_1} s' \right) + |\mathcal{L}_1| \cdot s_\Sigma = \frac{1}{2} s_\Sigma + n s_\Sigma = \left(\sum_{s'' \in \mathcal{S}_2} s'' \right) + |\mathcal{L}_2| \cdot s_\Sigma.$$

Recall that $\mathcal{S}_1, \mathcal{S}_2$ is a partition of $\mathcal{S}$. Hence, $\sum_{s'' \in \mathcal{S}_2} s'' = s_\Sigma - \sum_{s' \in \mathcal{S}_1} s'$. Therefore, $\sum_{s'' \in \mathcal{S}_2} s'' < s_\Sigma$ or $\sum_{s' \in \mathcal{S}_1} s' < s_\Sigma$ (or both). Furthermore, we know that $|\mathcal{L}_1| + |\mathcal{L}_2| = 2n$ since $\mathcal{L}_1, \mathcal{L}_2$ is a partition of $\mathcal{L}_v$. Together with the equation above, this gives $|\mathcal{L}_1| = n = |\mathcal{L}_2|$. Hence,

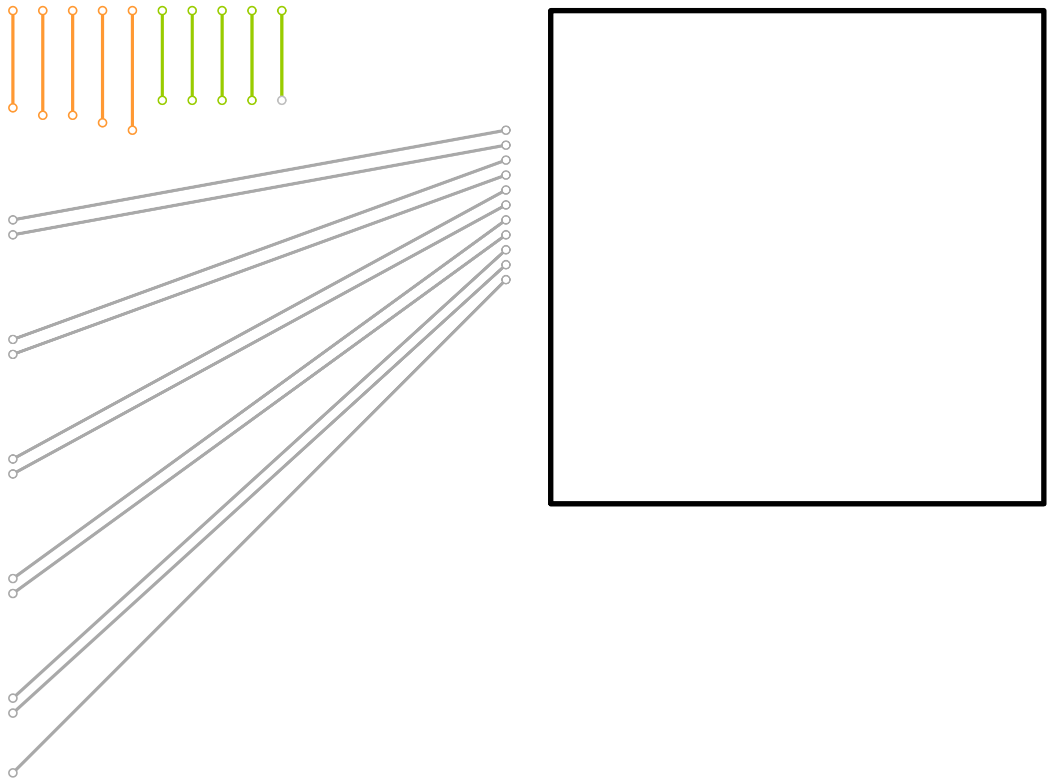
$$\sum_{s' \in \mathcal{S}_1} s' = \sum_{s'' \in \mathcal{S}_2} s''.$$

Summarizing, if the line segments in $\mathcal{L}$ can be packed into the container, there is a partition $\mathcal{S}_1, \mathcal{S}_2$ of $\mathcal{S}$ with $\sum_{s' \in \mathcal{S}_1} s' = \sum_{s'' \in \mathcal{S}_2} s''$.

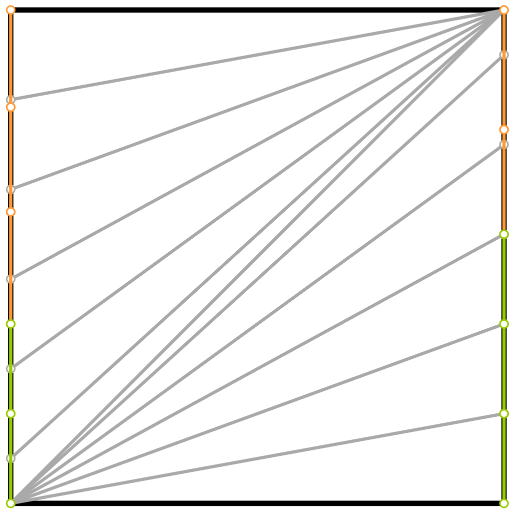
It remains to show that the reduction runs in polynomial time. Line 1 to Line 8 obviously run in polynomial time. Observe, that $k \in \mathcal{O}(n)$ and hence, also the computation of the diagonal line segments runs in polynomial time. Therefore, Algorithm 9.1 runs in polynomial time. This concludes the proof of Theorem 9.2.

9.2 Conclusion

We showed that packing line segments as defined in Definition 9.1 is NP-hard, i.e. deciding whether a given set of open line segments can be packed into a closed square container such that their interiors do not overlap is NP-hard. We explicitly do not allow parallel line segments to intersect, which is in contrast to other definitions of the problem. The result directly translates to other containers such as convex or polygonal containers. It is still open if this result also holds for, e.g., open convex containers or when there are no parallel line segments allowed.



(i) Multiset of line segments and container square with side length a generated from multiset of integers by Algorithm 9.1. The object line segments in $\mathcal{L}_o$ are depicted in orange, $\mathcal{L}_v$ is formed by the orange and green line segments, the diagonal line segments are grey.



(ii) The only way to pack the diagonal line segments is as depicted. Then, the vertical line segments have to be packed on the vertical boundaries of the container square.

Figure 9.1: Example for generated line segments and packing into container for YES-instance $\{1, 2, 2, 3, 4\}$ of Partition.

Bibliography

[1] M. Abrahamsen, T. Miltzow, and N. Seiferth. “Framework for ER-Completeness of Two-Dimensional Packing Problems”. In: *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS).* 2020, pp. 1014–1021.

[2] Anna Adamaszek, Tomasz Kociumaka, Marcin Pilipczuk, and Michał Pilipczuk. “Hardness of Approximation for Strip Packing”. In: *ACM Transactions on Computation Theory* 9.3 (2017), 14:1–14:7.

[3] Anna Adamaszek and Andreas Wiese. “A quasi-PTAS for the Two-dimensional Geometric Knapsack Problem”. In: *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 2015, pp. 1491–1505.

[4] Hee-Kap Ahn, Sang Won Bae, Otfried Cheong, Joachim Gudmundsson, Takeshi Tokuyama, and Antoine Vigneron. “A Generalization of the Convex Kakeya Problem”. In: *Algorithmica* 70.2 (2014), pp. 152–170.

[5] Helmut Alt. “Computational Aspects of Packing Problems”. In: *Bulletin of EATCS* 1.118 (2016). Number: 118.

[6] Helmut Alt, Mark de Berg, and Christian Knauer. “Approximating Minimum-Area Rectangular and Convex Containers for Packing Convex Polygons”. In: *Journal of Computational Geometry* 8.1 (2017). Number: 1, pp. 1–10.

[7] Helmut Alt, Kevin Buchin, Steven Chaplick, Otfried Cheong, Philipp Kindermann, Christian Knauer, and Fabian Stehn. “Placing Your Coins on a Shelf”. In: *Journal of Computational Geometry* 9.1 (2018), pp. 312–327.

[8] Helmut Alt, Otfried Cheong, Ji-won Park, and Nadja Scharf. “Packing 2D Disks into a 3D Container”. In: *Proceedings of the 13th International Conference and Workshops on Algorithms and Computation (WALCOM).* 2019, pp. 369–380.

[9] Helmut Alt and Nadja Scharf. “Approximating Smallest Containers for Packing Three-Dimensional Convex Objects”. In: *International Journal of Computational Geometry & Applications* 28.02 (2018), pp. 111–128.

[10] Helmut Alt and Nadja Scharf. “Polynomial Time Approximation Schemes for Circle Packing Problems”. In: *Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG).* 2017.

[11] Helmut Alt and Nadja Seiferth. “Approximating the Packing of Unit Disks into Simple Containers”. In: *Proceedings of the 36th European Workshop on Computational Geometry (EuroCG).* 2020, 84:1–84:6.

[12] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. "On the Combinatorial and Algebraic Complexity of Quantifier Elimination". In: *Journal of the ACM* 43.6 (1996), pp. 1002–1045.

[13] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. 3rd ed. Springer-Verlag TELOS, 2008. ISBN: 978-3-540-77973-5.

[14] Richard P. Brent and Paul Zimmermann. *Modern Computer Arithmetic*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2010.

[15] John Canny. "Some Algebraic and Geometric Computations in PSPACE". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. 1988, pp. 460–467.

[16] Joonsoo Choi, Jürgen Sellen, and Chee-Keng Yap. "Approximate Euclidean shortest path in 3-space". In: *Proceedings of the 10th annual symposium on Computational geometry (SoCG)*. 1994, pp. 41–48.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. Third Edition. The MIT Press, 2009. ISBN: 978-0-262-03384-8.

[18] Erik D. Demaine, Sàndor P. Fekete, and Robert J. Lang. "Circle Packing for Origami Design Is Hard". In: *Origami 5: Fifth International Meeting of Origami Science, Mathematics, and Education*. 2011.

[19] Florian Diedrich, Rolf Harren, Klaus Jansen, Ralf Thöle, and Henning Thomas. "Approximation Algorithms for 3D Orthogonal Knapsack". In: *Journal of Computer Science and Technology* 23.5 (2008), pp. 749–762.

[20] Michael Gene Dobbins and Heuna Kim. "Packing Segments in a Convex 3-Polytope is NP-hard". In: *Proceedings of the 30th European Workshop on Computational Geometry (EuroCG)*. 2014.

[21] Jack Edmonds. "Systems of Distinct Representatives and Linear Algebra". In: *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics* 71B.4 (1967), p. 241.

[22] *Erich's Packing Center*. URL: https://erich-friedman.github.io/packing/index.html (visited on 02/21/2021).

[23] L. Fejes Tóth. *Lagerungen in der Ebene auf der Kugel und im Raum*. 2. Springer Berlin Heidelberg, Mar. 1972. ISBN: 3-540-05477-4.

[24] Sándor P. Fekete, Sven von Höveling, and Christian Scheffer. "Online Circle Packing". In: *Proceedings of the 16th Algorithms and Data Structures Symposium (WADS)*. 2019, pp. 366–379.

[25] Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. "Packing Disks into Disks with Optimal Worst-Case Density". In: *Proceedings of the 35th International Symposium on Computational Geometry (SoCG)*. 2019, 35:1–35:19.

[26] Sándor P. Fekete, Sebastian Morr, and Christian Scheffer. “Split Packing: Algorithms for Packing Circles with Optimal Worst-Case Density”. In: *Discrete Comput. Geom.* 61.3 (2019), pp. 562–594.

[27] Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. “Optimal Packing and Covering in the Plane are NP-complete”. In: *Information Processing Letters* 12.3 (1981), pp. 133–137.

[28] M. R. Garey and D. S. Johnson. ““ Strong ” NP-Completeness Results: Motivation, Examples, and Implications”. In: *Journal of the ACM* 25.3 (1978), pp. 499–508.

[29] Thomas Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs.* Cambridge University Press, 2012. ISBN: 978-0-521-61770-3.

[30] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean Mclaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. “A Formal Proof of the Kepler Conjecture”. In: *Forum of Mathematics, Pi* 5 (2017).

[31] Juris Hartmanis and Janos Simon. “On the Power of Multiplication in Random Access Machines”. In: *Proceedings of the 15th Annual Symposium on Switching and Automata Theory (SWAT).* 1974, pp. 13–23.

[32] Kun He, Hui Ye, Zhengli Wang, and Jingfa Liu. “An Efficient Quasi-physical Quasi-human Algorithm for Packing Equal Circles in a Circular Container”. In: *Computers & Operations Research* 92 (2018), pp. 26–36.

[33] Mhand Hifi and Rym M’Hallah. “A Literature Review on Circle and Sphere Packing Problems: Models and Methodologies”. In: *Adv. Operations Research* 2009 (2009), pp. 150624–150624.

[34] Dorit S. Hochbaum and Wolfgang Maass. “Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI”. In: *Journal of the ACM* 32.1 (1985), pp. 130–136.

[35] Wenqi Huang and Tao Ye. “Global Optimization Method for Finding Dense Packings of Equal Circles in a Circle”. In: *European Journal of Operational Research* 210.3 (2011), pp. 474–481.

[36] Klaus Jansen and Lars Prädel. “A New Asymptotic Approximation Algorithm for 3-Dimensional Strip Packing”. In: *Proceedings of the 40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM).* 2014, pp. 327–338.

[37] Sōichi Kakeya. “Some Problems on Maxima and Minima Regarding Ovals”. In: *Tohoku Science Reports* 6 (1917), pp. 71–88.

[38] Josef Kallrath and Markus M. Frey. “Packing Circles into Perimeter-minimizing Convex Hulls”. In: *Journal of Global Optimization* 73.4 (2019), pp. 723–759.

[39] Dania El-Khechen, Muriel Dulieu, John Iacono, and Nikolaj van Omme. “Packing 2×2 Unit Squares into Grid Polygons is NP-complete”. In: *Proceedings of the 21st Annual Canadian Conference on Computational Geometry (CCCG)*. 2009, pp. 33–36.

[40] Heuna Kim and Tillmann Miltzow. “Packing Segments in a Simple Polygon is APX-hard”. In: *Proceedings of the 31st European Workshop on Computational Geometry (EuroCG)*. 2015, pp. 24–27.

[41] Flávio K. Miyazawa, Lehilton L. C. Pedrosa, Rafael C. S. Schouery, Maxim Sviridenko, and Yoshiko Wakabayashi. “Polynomial-Time Approximation Schemes for Circle and Other Packing Problems”. In: *Algorithmica* 76 (2016), pp. 536–568.

[42] *Packomania*. URL: http://www.packomania.com/ (visited on 11/18/2019).

[43] Julius Pal. “Ueber ein Elementares Variationsproblem”. In: *Det Kgl. Danske Videnskabernes Selskab., Mathematisk-fysiske Meddelelser* III.2. (1920), p. 35.

[44] Christos H. Papadimitriou. “An Algorithm for Shortest-Path Motion in Three Dimensions”. In: *Information Processing Letters* 20.5 (1985), pp. 259–263.

[45] *Problem 55: Pallet Loading*. from The Open Problems Project edited by Erik D. Demaine, Joseph S.B. Mitchell, Joseph O’Rourke. URL: https://cs.smith.edu/~jorourke/TOPP/P55.html#Problem.55 (visited on 08/06/2019).

[46] Nadja Scharf. *On a Detail in Hales’s "Dense Sphere Packings: A Blueprint for Formal Proofs"*. arXiv:1712.03568 [math]. 2017. URL: http://arxiv.org/abs/1712.03568.

[47] Guntram Scheithauer. *Zuschnitt- und Packungsoptimierung: Problemstellungen, Modellierungstechniken, Lösungsmethoden*. Vieweg+Teubner Verlag, 2008. ISBN: 978-3-8351-0215-6.

[48] Arnold Schönhage. “On the Power of Random Access Machines”. In: *Proceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP)*. 1979, pp. 520–529.

[49] Chee-Keng Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, 2000. ISBN: 978-0-19-512516-0.

Zusammenfassung

Die Arbeit ist in zwei Teile unterteilt, wobei im ersten Teil Packungsprobleme untersucht werden, in denen gleiche Objekte gepackt werden:

Der Container ist ein fettes Parallelogramm oder Dreieck. Wir zeigen, dass es für folgendes Problem ein PTAS gibt: Gegeben ein Containerparallelogram, dessen Innenwinkel nach unten durch eine Konstante beschränkt sind (es ist *fett*), wie viel Kopien eines Objekts, das entweder Teil der Problembeschreibung und nicht Teil der Eingabe für den Algorithmus ist oder dessen Fläche dividiert durch seinen quadrierten Umfang nach unten beschränkt ist durch eine Konstante, können unter Translation oder euklidischen Transformationen in den Container gepackt werden? Gleiches gilt für ein Containerdreieck.
Der Container ist ein beliebiges Dreieck. Wir beschreiben einen Algorithmus mit konstantem Approximationsfaktor, der gegeben ein beliebiges Dreieck in Polynomialzeit eine Näherung für die maximale Anzahl an Einheitskreisen, die in das Dreieck gepackt werden können, berechnet.
Packen von Einheitskreisen und Einheitskugeln. Wir geben für jedes der folgende Probleme einen PTAS an: Gegeben einen konvexen Container, dessen Fläche dividiert durch seinen quadrierten Umfang nach unten durch eine Konstante beschränkt ist, was ist die maximale Anzahl an Einheitskreisen, die hineingepackt werden können. Wie viel Einheitskreise können in ein festes skaliertes Polygon gepackt werden? Wieviel Einheitskugeln können in eine gegebene Kugel gepackt werden in drei Dimensionen? Um das letzte Ergebnis zu erhalten, verstärken wir ein Theorem von Hales et al. [29, 30], indem wir die entsprechenden Beweise modifizieren.
Packen von Einheitskreisscheiben in 3D unter Translation. Gegeben eine Menge von Kreisscheiben in 3D, was ist der achsenparallele Quader kleinsten Volumens, sodass alle Kreisscheiben hineingepackt werden können? Für dieses Problem beschreiben wir einen Algorithmus mit konstantem Approximationsfaktor und polynomieller Laufzeit. Zusätzlich zeigen wir, dass es keinen konvexen Container endlichen Volumens geben kann, in den wir alle möglichen verschiedenen Einheitskreisscheiben zusammen packen können.

Im zweiten Teil der Arbeit untersuchen wir die Komplexität von zwei grundlegenden Packungsproblemen. Wir zeigen, dass es für folgendes Problem keinen FPTAS geben kann: Was ist der kleinste konvexe Container, in den eine gegebene Multimenge von Polygonen gepackt werden kann unter Translation. Selbiges gilt, wenn wir zusätzlich Rotationen um 90° erlauben. Als zweites zeigen wir, dass es NP-schwer ist zu entscheiden, ob eine Multimenge von offenen Strecken in ein gegebenes geschlossenes Quadrat gepackt werden kann. Die Strecken dürfen sich dabei nicht in ihrem Inneren schneiden.