Universidade Estadual de Campinas
Instituto de Computação

Elisa Dell'Arriva

In-line Packing of Circles

Empacotamento de Círculos em Linha

CAMPINAS
2021

Elisa Dell'Arriva

In-line Packing of Circles

Empacotamento de Círculos em Linha

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Flávio Keidi Miyazawa

Este exemplar corresponde à versão final da Dissertação defendida por Elisa Dell'Arriva e orientada pelo Prof. Dr. Flávio Keidi Miyazawa.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Dell'Arriva, Elisa, 1993-
D38i In-line packing of circles / Elisa Dell'Arriva. – Campinas, SP : [s.n.], 2021.

Orientador: Flávio Keidi Miyazawa.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Empacotamento de círculos. 2. Algoritmos de aproximação. 3. Problemas de empacotamento. I. Miyazawa, Flávio Keidi, 1970-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Empacotamento de círculos em linha
Palavras-chave em inglês:
Circle packing
Approximation algorithms
Packing problems
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
Flávio Keidi Miyazawa [Orientador]
Cristina Gomes Fernandes
Eduardo Candido Xavier
Data de defesa: 17-06-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-7505-5386
- Currículo Lattes do autor: http://lattes.cnpq.br/3025112976267292

Universidade Estadual de Campinas
Instituto de Computação

Elisa Dell'Arriva

In-line Packing of Circles

Empacotamento de Círculos em Linha

Comissão Examinadora:

- Prof. Dr. Flávio Keidi Miyazawa
  Universidade Estadual de Campinas
- Prof. Dr. Eduardo Candido Xavier
  Universidade Estadual de Campinas
- Profa. Dra. Cristina Gomes Fernandes
  Universidade de São Paulo

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 17 de junho de 2021

To José Luís, Josiane, Renata and Guilherme.

Acknowledgements

I shall start thanking my family; without them I would not be where I am today. I will be forever grateful for having them in my life and for the immense support they have always given me. Growing up, my patents used to tell me that working with what makes you happy is something valuable. I am grateful for having had such opportunity. I thank my father for instigating me to pursue something that makes me happy. He certainly loves his profession and has always set an example for me. I thank my mother for putting so much effort into ensuring that my siblings and I could go to university. I also thank her for taking me to classes when I could not have done by myself. I thank my sister, for always being there for me, showing support and encouragement so that I would keep going and face the difficult times. Finally, I thank my brother, for all the shared moments (and for the not so little amount of coffee). José Luís, Josiane, Renata and Guilherme, I truly love you.

Maria Esther, thank you for helping me grow. I am positive that without your influence in my life, I would not have overcome several of the obstacles in my way.

I thank some special friends who have always been supportive of me during so many outbursts. Milena, I thank you for caring so much and being there. Mayara, I thank you for every and each moment of encouragement and for keeping reminding me that I am capable of achieving my goals. Felipe, I thank you for the constant support. I thank Laura, for helping me through the very first semester. I thank some friends from undergraduate. I thank Mariane, who kept instigating me to pursue the so called academic career despite my fears and doubts. I thank Lucas, for offering me support in a period where I found myself quite ill. I thank Wilson and Priscilla, for so many times helping me with the bureaucracies.

I thank every teacher that, one way or another, contributed with my education. I want to specially thank some of them who played crucial roles in my academic path so far. Chris, I will never forget when, in the very first semester of my undergraduate, you one day made time to talk to me about my insecurities. You offered me words of encouragement to keep going. Several semesters ahead, you again showed me support and agreed to be advisor in undergraduate. Lehilton, I thank you for also being my advisor in undergraduate. You kept faith in my potential despite my difficulties and you have since then been available to help me. I thank my advisor, Flávio, for granting me the opportunity of being one of your students. I remember that, when I contacted you about your projects, I was behind the expected schedule, but you still agreed to be my advisor. I was quite worried about changing my research field this late. I worried that I would not be able to follow through with the research, that I had not a strong enough background and things of sort. Despite that, you was always patient and attentive. I can easily state that without such supportive orientation, I would not have concluded my Masters with success. I thank Eduardo and Cristina, for accepting to compose the evaluation board.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo

Problemas de empacotamento têm muitas aplicações práticas. Eles podem ser usados para modelar problemas como corte de materiais, armazenamento e transporte de mercadorias, escalonamento de processos e alocação de anúncios e propagandas, entre outros. Nesta dissertação, estamos interessados em problemas de empacotamento em que os objetos a serem empacotados possuem formas geométricas bidimensionais. Em particular, abordamos um problema que aqui chamamos de problema do Empacotamento de Círculos em Linha (*In-line Circle Packing (ICP)*). Neste problema, temos como entrada uma lista de círculos e queremos empacotá-los sobre uma linha horizontal de forma que cada círculo toque a linha por cima em exatamente um ponto, sem causar sobreposição dos círculos. O objetivo é minimizar o *span* do empacotamento, isto é, minimizar a distância entre o ponto mais à esquerda e o ponto mais à direita do empacotamento.

Como acontece com a maioria dos problemas de empacotamento, o ICP é NP-difícil. Isso significa que não existe algoritmo capaz de produzir uma solução ótima em tempo polinomial, a menos que P = NP. Sendo assim, uma forma natural de investigar esses problemas é obter de forma eficiente soluções cujos valores estejam próximos do valor ótimo. Uma abordagem nessa direção é a dos Algoritmos de Aproximação. Em termos gerais, algoritmos de aproximação são algoritmos eficientes que produzem uma solução com garantia de qualidade em relação à solução ótima, para todas as instâncias do problema. Neste trabalho, estamos particularmente interessados em algoritmos de aproximação que produzem soluções cujos valores estão dentro de um fator do valor de uma solução ótima. Apresentamos alguns resultados encontrados na literatura para o ICP, como uma prova de sua NP-dificuldade e um algoritmo de aproximação de fator constante. Mostramos um PTAS simples para uma versão restrita do problema ICP, em que todos os círculos têm raio cujo valor é pelo menos uma constante. Além disso, investigamos uma versão do ICP em *bins*, que chamamos de problema do Empacotamento de Círculos em Linha em Bins (*In-line Circle Bin Packing (ICB)*). Nesta versão, devemos empacotar os círculos dentro de *bins* e cada círculo deve tocar o fundo da *bin* em exatamente um ponto. Mostramos um APTAS para esta variante restrito a grandes círculos em *bins* aumentadas.

Abstract

Packing problems have many practical applications. They can be used to model problems such as cutting stock, storage and transportation of goods, processes scheduling, and allocation of advertisements, among others. In this dissertation, we are interested in packing problems where the objects to be packed have two-dimensional forms. In particular, we address a problem we call here In-line Circle Packing (ICP) problem. In this problem we have as input a list of circles, and we want to pack the circles on a horizontal line in a way that each circle touches the line from above at exactly one point, without causing overlap of the circles. The goal is to minimize the *span* of the packing, i.e., minimize the distance between the leftmost and the rightmost points of the packing.

As it happens with the majority of the packing problems, the ICP is NP-hard. This means that there is no algorithm capable of producing an optimal solution in polynomial time, unless P = NP. That being the case, one natural way to investigate these problems is to efficiently obtain solutions whose values are close to the optimal value. One approach in this direction is that of Approximation Algorithms. In general terms, approximation algorithms are efficient algorithms that produce a solution with a guarantee of quality in relation to the optimal solution, for every instance of the problem. In this work, we are particularly interested in approximation algorithms that produce solutions whose values are within a factor of the value of an optimal solution. We present some results found in the literature for the ICP problem, such as a proof of its NP-hardness and a constant factor approximation algorithm. We show a simple PTAS for a restricted version of the ICP problem, where all circles have radius whose value is at least a constant. In addition, we investigate a version of the ICP in bins, which we call the In-line Circle Bin Packing (ICB) problem. In this version, we must pack the circles inside bins and every circle must touch the bottom of the bin at exactly one point. We show an APTAS for this variant restricted to large circles in augmented bins.

Contents

Chapter 1

Introduction

The problems exploited in this dissertation are optimization problems of combinatorial nature. We say an *optimization problem* is a problem where one aims to minimize or maximize a function, which we call *objective function*. In this text, we call *solution* of an optimization problem a set of attributes that satisfies all the restrictions of the problem, except maybe by the objective function. We then say a solution is *optimal* if it satisfies all the restrictions of the problem, including the objective function.

We are particularly interested in a class of problems referred to as packing problems. Packing problems are quite useful, since they appear in several contexts in the commonly said real-world applications. They can model problems from logistics and loading trucks and shelves, to stock-cutting where large lengths of materials need to be sliced into smaller defined shapes, to scheduling processes in shared machines or distributing time in commercial breaks in television or spaces in websites, to resources allocation such as memory allocation, and so on. It serves well to handle partition problems in general. However, packing problems are, in majority, problems for which it is not known whether or not there exist efficient algorithms capable of computing optimal solutions. It does not mean, though, that one should not give such problems much attention. On the contrary, besides being convenient in applied fields, they are also interesting problems from the theoretical perspective, in particular to design and analysis of algorithms.

There are several versions of packing problems appearing in the literature. Here we cite the bin packing, the strip packing and the knapsack problems. In the bin packing problem, we have an unrestricted number of recipients, and the goal is to use the minimum number of recipients to pack all items. In contrast to the bin packing, the strip-packing problem has one recipient with one dimension not fixed and the goal is to pack every item as to minimize the length used of the free dimension. Finally, in the knapsack problem, we have one recipient with fixed dimensions and a function of values on the items, and the goal is to pack a subset of items inside the recipient maximizing the total sum of the values of the packed items.

The most classic packing problem is the one dimensional version of bin packing. In this version the items are one-dimensional bars whose heights are in the interval $(0, 1]$ and the recipients are unit one-dimensional bars. The decision version of this problem is well known to be NP-complete, while the optimization version is NP-hard. In fact, the optimization

versions of all variants of packing problems mentioned above are NP-hard. The reductions are usually from the partition problem, the subset sum problem or the numerical matching problem. Therefore, unless P = NP, it is not likely to exist an efficient exact algorithm to solve such problems. Since it is widely believed that P $\neq$ NP, there are investigations on other ways to approach NP-hard problems. They are basically four: *i*) we can focus on designing exact algorithms that work well for small instances, despite not being polynomial, *ii*) we can design algorithms that are exact and polynomial only for particular cases of the problem, as in the field of parameterized algorithms, *iii*) we can invest in heuristics that seems to work well in practice although without any guarantees, or *iv*) we can sacrifice optimality by designing efficient algorithms capable of producing not an optimal solution, but a solution whose value is close to the value of an optimal solution, for any instance of the problem. In this dissertation, we exploit the last approach, more precisely, the field called Approximation Algorithms.

Approximation Algorithms refer to algorithms that produce solutions which can be formally compared to optimal solutions. In this dissertation, we work specifically with approximation algorithms that compute solutions whose value is within a factor of the value of an optimal solution. We refer to such factor as *approximation factor*. The beginnings of Approximation Algorithms date to the 1960 decade, through the work of Graham [12] on multiprocessors scheduling of tasks and factors that might trigger anomalies in the system. Another work that is related to this theme is one of Erdös [8] on bipartite subgraphs. Latter, Garey et al. [10] and Johnson [16] offered formal definitions for these ideas. Since then the field of Approximation Algorithms has been widely studied in combinatorial optimization. In 1981, Coffman et al. [4] published a survey specifically on approximation algorithms for the one-dimensional bin packing problem. In 1984, the same authors [5] updated the survey with many new results. Ideally, we aim to obtain solutions whose value is as close as possible to the optimal value. That is where the concept of approximation scheme emerges. Given a positive constant ε, we say an *approximation scheme* is a family of approximation algorithms with approximation factor equals to $1 + \varepsilon$. Traditionally, when speaking of approximation algorithms, we speak of efficient algorithms, i.e., algorithms that run in polynomial time. However, we sometimes may obtain algorithms that compute solutions whose value is close to the optimal value in time greater than polynomial but still less than exponential. One example is *quasi*-polinomial-time algorithms. We say PTAS for a polynomial time approximation scheme and QPTAS for a *quasi*-polinomial-time approximation scheme. One famous and important result on approximation algorithms for packing problems is the asymptotic PTAS proposed by de la Vega and Lueker [9], in 1981, for the one-dimensional bin packing problem. This work introduced a very useful technique called *linear grouping*, which is considered a mark in the history of approximation algorithms for packing problems.

In this dissertation, we consider packing problems where the items are two-dimensional geometric forms. In 2012, Allen and Iacono [19] showed that the problem of packing copies of a smaller polygon, the items, inside a larger polygon, the recipient, is NP-hard. That is an interesting result for its scope is general geometric forms. More specifically, the packing problems regarded here are somewhat related to the versions of bins and strip. Classically, the items, the bins and the strips are rectangles. In the two-dimensional bin packing problem,

we have a list of small rectangles, the items, and an unrestricted number of unit squares, that we call *bins*. The objective is to find a packing of all items into the minimum number of bins. A packing here is a positioning of the rectangles in a way that every rectangle is entirely contained inside the bins and there is no overlap among items. In the strip packing problem, we have a list of rectangle as input as well as *strip* with fixed height and the objective is to pack all circles minimizing the height of the strip. In 2014, Lodi et al. [17] gathered some results for the two-dimensional bin and strip packings, though their concern is not only approximation algorithms but heuristics and exact algorithms as well. In a survey of 2017, Christensen et al. [3] consider approximation results for several versions and generalizations of the bin packing problem. Some of these generalizations are the two-dimensional bin packing, the strip packing and the knapsack problems. For the strip packing problem, there is one classical result from 1997, which is a 2-approximation due to Steinberg [20]. In 2009, Harren and van Stee [15] improved this factor to 1.9396. The best known result so far for the two-dimensional strip packing is a $(5/3 + \varepsilon)$-approximation due to Harren et al. [14] in 2014. In this dissertation, we are particularly interested in the case where the items are circles. In 2016, Miyazawa et al. [18] showed an asymptotic PTAS for circle packing into augmented bins. The authors observed that such PTAS can be extended to circle packing into a strip.

The strip packing problem admits a variant where every circle must touch the bottom of the recipient. Such variant is called level strip packing. Here the strip has fixed width and free height. The objective then is to minimize the total height used to pack all items. A level can be interpreted as a horizontal strip inside the strip but vertically bounded from above and from below. The height of each level is given by the height of the largest item packed in that level. The height of a level defines its upper bound. The lower bound is defined by the upper bound of the previous level, that is, the level immediately below. The lower bound of the first level is defined by the bottom of the strip. This variant can be applied, for instance, in organization of objects in a shelf.

In this dissertation, we exploit one particular packing problem, which we called in-line packing problem, that resembles the level strip packing problem. The in-line packing problem is quite simple to formulate. From a very abstract view, we have a list of items and aim to organize them over some horizontal surface, avoiding overlaps. We particularly considered circles as the items and a horizontal line as the surface. Then, we wish to obtain a positioning of every circle such that each one touches the line from above at exactly one point and there is no overlaps among circles. The objective is to minimize the length or span of the packing, i.e., the distance from the leftmost to the rightmost points of the packing. Note that, although the items are two-dimensional, this problem resembles a one dimensional packing, since the position of every circle can be given by one coordinate. Figure 1.1 illustrates one example of an in-line packing of circles on a horizontal line. Our work is based mainly on two other works found in the literature. Alt et al. [1] studied the problem they called packing coins on a shelf, which is basically the in-line packing problem for circles. They show the problem is NP-hard and propose a greedy algorithm that is an approximation algorithm with factor 4/3. Dürr et al. [7] considered the problem of scheduling processes with different criticality in a singular machine. As the authors themselves observe, this problem can be interpreted as an in-line packing of triangles. They even refer to it as the triangle scheduling problem. In

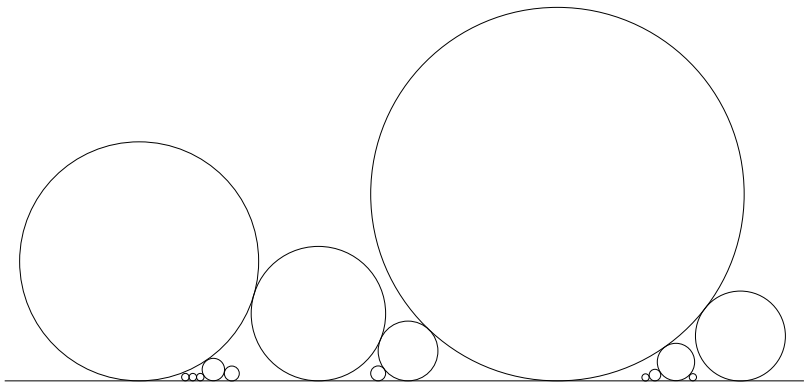


Figure 1.1: Illustration of an in-line packing of Circles.

their case, the triangles are right and isosceles. They show the problem is NP-hard and offer a 3/2 factor approximation algorithm. Furthermore, they show a QPTAS for the problem.

The remaining of this text is organized as follows. In Chapter 2 we introduce some concepts and notation used further on. In Chapter 3 we present the NP-hardness proof of the In-line Circle Packing problem, as showed by Alt et al. [1] in their work of 2018. Then, in Chapter 4, we present some results found in the literature as well as the work of our project. In the first section, we show the 4/3-approximation proposed by Alt et al. [1] for the in-line circle packing. In the next section of this chapter, we show the QPTAS for the in-line circle packing mentioned previously. This QPTAS is based on rounding and dynamic programming and it is highly based on the QPTAS showed by Dürr et al. [7] for in-line packing of triangles, also from 2018. In the last section of the chapter, we present an APTAS in augmented bins for the bin packing version of the in-line circle packing, as well as a simple PTAS for the in-line packing of circles when the size of the circles is lower bounded. Finally, in Chapter 5, we offer some final remarks.

Chapter 2

Preliminaries

In this chapter, we introduce formal definitions of the problems addressed later on this text, as well as some general results regarding circles. Moreover, we present some basic concepts of approximation algorithms.

In the context of the problems here discussed, we will use the xy-euclidean plane as reference. Given two points (x_1, y_1) and (x_2, y_2) in the plane we denote by $dist((x_1, y_1), (x_2, y_2))$ the euclidean distance between the points (x_1, y_1) and (x_2, y_2), i.e., $dist((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Furthermore, given a circle i, we denote its radius by r_i and its diameter by d_i. In many contexts along this text, we need to use ordered sets. In such cases, we represent a set as a list, and so we apply the set notation and operators equally for lists and sets.

2.1 Problems

If we observe the structure of a packing for the In-line Circle Packing problem, we notice that the y-coordinate of a circle is given by its radius. Therefore, to represent an in-line packing it is sufficient to know the x-coordinate of the center of each circle. Then, we offer a standard notation for such point.

Definition 1. *Given a list L of circles and an in-line packing $\mathcal{P}$ of L, the* base position *of a circle i in $\mathcal{P}$, denoted by $\mathcal{P}(i)$, is the point where i touches the x-axis.*

In the following, we present a formal definition of an in-line packing. Without loss of generality, if not mentioned otherwise, the input list will have circles with radius at most 1.

Definition 2. *Given a list L of circles, each circle $i \in L$ with radius r_i, and the x-axis of the euclidean plane, an* in-line packing *of L is a function $\mathcal{P} : L \to \mathbb{Q}$ respecting the following restrictions:*

- $\mathcal{P}(i) - r_i \geq 0$;
- $dist((\mathcal{P}(i), r_i), (\mathcal{P}(j), r_j)) \geq r_i + r_j$, *for* $1 \leq i \neq j \leq n$.

Without loss of generality, we consider that the leftmost point of a packing $\mathcal{P}$ has its x-coordinate equals to 0, i.e., $\min\{\mathcal{P}(i) - r_i : i \in L\} = 0$. *Then, we define the* size *of the packing $\mathcal{P}$ as* $\max\{\mathcal{P}(i) + r_i : i \in \mathcal{P}\}$, *and we denote it by $|\mathcal{P}|$.*

The first constraint guarantees that every circle lies entirely on the non-negative x-axis, while the second constraint guarantees that there is no overlap between every two circles. Now we can formally state the optimization version of the In-line Circle Packing problem.

Problem 1 (In-line Circle Packing (ICP))**.** *Given a tuple (L, r) where L is a list of circles and r a function of their radii, find an in-line packing of L of minimum size.*

Although the input list of circles is given together with the radii of the circles, throughout the text we switch between radius and diameter as it becomes more convenient. In some occasions, we have two or more lists of circles and we want to unify them. Given two lists L_1 and L_2 of circles, the *concatenation* of L_1 and L_2, denoted by $L_1 \| L_2$, is the list L containing all circles of L_1 and all circles of L_2. Similarly, in some contexts we might have two or more packings and we wish to join them together, obtaining then one final packing. One way to do that is to sequentially add one packing to the end of the other.

Definition 3. *Given two in-line packings $\mathcal{P}_1$ and $\mathcal{P}_2$, the* concatenation *of $\mathcal{P}_1$ and $\mathcal{P}_2$ is a packing $\mathcal{P}$ obtained by adding $\mathcal{P}_2$ at the end of $\mathcal{P}_1$ as much to the left as possible without causing overlaps. We denote $\mathcal{P} = \mathcal{P}_1 \| \mathcal{P}_2$.*

Based on the idea of an in-line packing, we propose a slightly different version of the two-dimensional bin-packing problem, which we call In-line Circle Bin Packing. In the classic version of the problem, we have a list of items, in this case we will consider circles, and we want to pack all the items into bins minimizing the number of bins used in the packing. For the ICB problem we have the same input and objective as in the classical bin packing, but there is an additional constraint: every item must touch the bottom of the bin where it is packed at exactly one point. In the following, we present a formal definition of what characterizes a packing in this context.

Definition 4. *Given a tuple (L, r) where L is a list of n circles and $r : L \to (0, 1]$ is a function of the radii, an* in-line bin packing *is a partition of L into sublists $L_1, \ldots, L_m$, and in-line packings $\mathcal{P}_i$ for each L_i such that $|\mathcal{P}_i| \leq 2$, for $i = 1, \ldots, m$. We define the* size *of the in-line bin packing $\mathcal{P}$, denoted by $|\mathcal{P}|$, as the number of parts m.*

Previously, we discussed the In-line Circle Packing problem where the objective is to obtain an in-line packing of minimum size. In the In-line Circle Bin Packing problem, we want in-line packings, each one inside one bin. Consequently, each in-line packing must have size at most two, and instead of touching the x-axis, here the circles must touch the bottom of the bin. A formal definition of the problem is given below.

Problem 2 (In-line Circle Bin Packing (ICB))**.** *Consider a list L of n circles. Given a tuple (n, r) where $r : [n] \to (0, 1]$ is a function of the radii of the circles, and bins $\mathcal{B}$ of capacity 2, find an in-line bin packing $\mathcal{P}$ of L into bins $\mathcal{B}$ that minimizes the size of $\mathcal{P}$.*

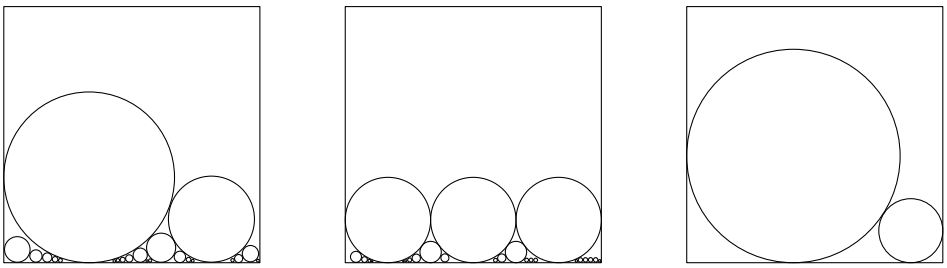


Figure 2.1: Illustration of the In-line Circle Bin Packing problem.

Figure 2.1 shows an example of the In-line Bin Packing problem. In some contexts it might be convenient to use bins of certain size different than 2. In such cases, we add a parameter W to the problem, which is the width of the bins. Then the instance becomes a tuple (L, r, W), where W is the width of the bins.

2.2 Notation and General Results

In this section, we introduce some basic notation that is used throughout this text. We remind the reader that, although the input list of circles is given together with the radii of the circles, we switch between radius and diameter as it becomes more convenient. Also, when convenient, we represent an in-line packing by a list of the circles in the packing sorted in increasing order of base positions, from left to right. We alert the reader that this is for the sole purpose of a simpler notation, since we need not only the radii of the circles but also their base positions in order to reconstruct a packing. A more detailed discussion on this topic is offered in Chapter 3. Besides the base positions, we might sometimes need to refer to the two extremes of the packing as well.

Definition 5. *Given an in-line packing $\mathcal{P}$, the* left (right) boundary *of $\mathcal{P}$ is the x-coordinate of the vertical line that crosses the leftmost (rightmost) point of $\mathcal{P}$. We denote the leftmost point of $\mathcal{P}$ by $\vdash_{\mathcal{P}}$ and the rightmost point of $\mathcal{P}$ by $\dashv_{\mathcal{P}}$.*

In Figure 2.2, we illustrate these basic notations. Now suppose we have two consecutive circles in the packing. Then, there is a space between them where some other circles could be packed in. We refer to such space as a *gap*.

Definition 6. *Given a list L of circles and an in-line packing $\mathcal{P}$ of L, let i and j be two consecutive circles in $\mathcal{P}$. Then a* gap *between i and j, denoted by $gap(i, j)$, is a tuple*

$$(r_i, \mathcal{P}(i), r_j, \mathcal{P}(j)).$$

The value *of a gap between i and j is the distance between the base positions of i and j, i.e.,*

$$gap(i, j) = dist(\mathcal{P}(i), \mathcal{P}(j)).$$

Given two circles i and j, we say a circle k *fits* in the gap $gap(i, j)$ if k can be packed in the space between i and j without causing any overlaps nor any increase in the size of the

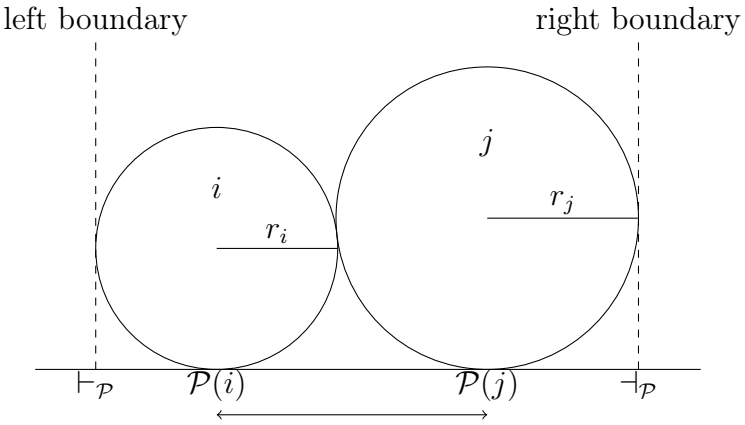


Figure 2.2: Illustration of some basic notations.

packing. Now before we go further, we need some preliminary lemmas regarding geometric properties of circles. A sketch of the calculations from the lemmas presented ahead is shown in Figure 2.3.

Lemma 2.1. *If two circles i and j touch each other in an in-line packing, then the distance between their base positions is $2\sqrt{r_i}\sqrt{r_j}$.*

Proof. If $r_i = r_j$ then the distance between i and j is $2r_i = 2\sqrt{r_i}^2 = 2\sqrt{r_i}\sqrt{r_i}$, and the result follows. Suppose, without loss of generality, that $r_i > r_j$ and that i is to the left of j. Consider the right triangle with hypotenuse $r_i + r_j$. By the Pythagorean Theorem we have that the distance between i and j is $\sqrt{(r_i + r_j)^2 - (r_i - r_j)^2} = \sqrt{4r_i r_j} = 2\sqrt{r_i}\sqrt{r_j}$. □

The following lemma gives us a relation between a gap and the size of the largest circle that fits in that gap.

Lemma 2.2. *Let k be a largest circle that fits between two touching circles i and j. Then* $\dfrac{1}{\sqrt{r_k}} = \dfrac{1}{\sqrt{r_i}} + \dfrac{1}{\sqrt{r_j}}$.

Proof. We know k is the largest circle that fits in the gap between i and j so k must touch both i and j. Without loss of generality, suppose i is to the left of j. Then the distance between i and j is equal to the distance between i and k plus the distance between k and j. By Lemma 2.1 we have $2\sqrt{r_i}\sqrt{r_j} = 2\sqrt{r_i}\sqrt{r_k} + 2\sqrt{r_k}\sqrt{r_j}$. From that we obtain the relation $\dfrac{1}{\sqrt{r_k}} = \dfrac{1}{\sqrt{r_i}} + \dfrac{1}{\sqrt{r_j}}$. □

Finally, we consider the case where a circle is touching one of the boundaries of the packing. The following lemma contemplates the case of the left wall; the case of the right wall is analogous.

Lemma 2.3. *Let k be a largest circle that fits between the left wall and a circle i touching that wall. Then $\sqrt{r_k} = (\sqrt{2} - 1)\sqrt{r_i}$.*

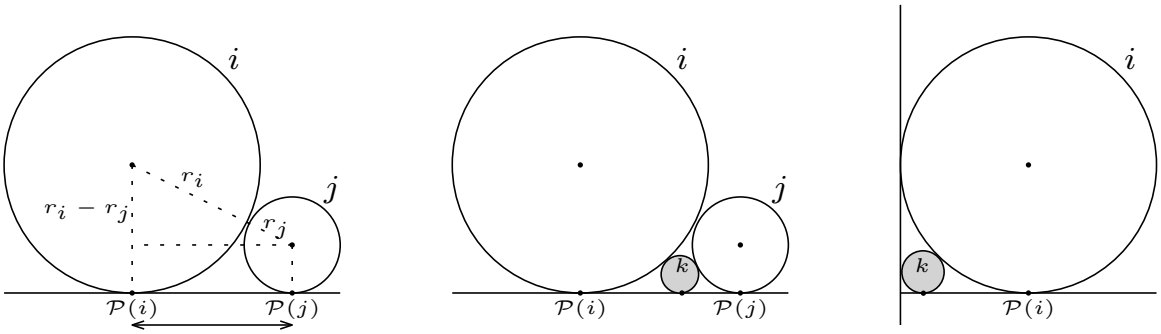


Figure 2.3: Illustration for Lemmas 2.1, 2.2 and 2.3, respectively from left to right.

Proof. We know k must be touching both i and the left wall. Then the distance between the left wall and the base position of i is its radius. Moreover, the distance between i and k plus the radius of k is equals to the radius of i. By Lemma 2.1 we have $r_i = 2\sqrt{r_i}\sqrt{r_k} + r_k$. From that we obtain the relation $\sqrt{r_k} = (\sqrt{2} - 1)\sqrt{r_i}$. □

2.3 Approximation Algorithms

In this section, we offer some basic concepts of approximation algorithms. They are necessary to a better understanding of the remaining of the text. In this dissertation we regard approximation algorithms whose solutions are within a factor of the value of an optimal solution. Moreover, the problems considered here are minimization problems, so we present the definitions accordingly. The definitions for maximization problems are analogous.

Definition 7. *Let $\mathcal{A}$ be an algorithm, and let I be an instance of $\mathcal{A}$. We denote by $\mathcal{A}(I)$ the value of the solution produced by $\mathcal{A}$ for I, and by* $\mathrm{OPT}(I)$ *the value of an optimal solution for the instance I. Then $\mathcal{A}$ is an* approximation algorithm *with approximation factor $\alpha \geq 1$ if for any instance I we have $\mathcal{A}(I) \leq \alpha\mathrm{OPT}(I)$.*

Ideally, we want to be as close to the optimum as possible. There is another type of approximation algorithms where the approximation factor is not absolute, but instead it depends on another constant, which we commonly refer to as ε. In this case, the algorithm has as input an instance of the problem together with a positive constant ε, and the solution produced has a value within a factor of $1 + \varepsilon$ of the optimal value. Note that, in a sense, we have not only one algorithm, but a family of approximation algorithms. This concept, of a family of $(1 + \varepsilon)$-approximation, is known as *approximation scheme*.

Definition 8. *A* polynomial-time approximation scheme (PTAS) *is a family of polynomial-time algorithms $\{\mathcal{A}_\varepsilon\}$, in which there is an algorithm A_ε for every $\varepsilon > 0$ such that, for every instance I, it holds that $A_\varepsilon(I) \leq (1 + \varepsilon)\mathrm{OPT}(I)$.*

Although the definition of a PTAS implies that we have a $(1 + \varepsilon)$-approximation, in some cases, we might actually prove a $(1 + O(\varepsilon))$-approximation. Despite the fact that approximation algorithms are defined as polynomial time algorithms, we have found in the

literature some "approximation schemes" that do not achieve such time complexity. One example is *quasi*-polinomial time, i.e., $2^{O(\log^c n)}$, for some constant c, algorithms. In this text, we say approximation algorithms for algorithms that produce a solution whose value is within a factor of the optimal values, regardless of its time complexity. The same idea holds for approximation schemes. The motivation behind this abuse of formality is the fact that logarithmic order is close to polynomial order. It is closer to polynomial order than it is to exponential order, for instance. In the following, we bring a formal definition of an approximation scheme in *quasi*-polinomial time.

Definition 9. *A* quasi-polynomial-time approximation scheme (QPTAS) *is a family of quasi-polinomial-time algorithms* $\{\mathcal{A}_\varepsilon\}$*, in which, for every* $\varepsilon > 0$ *and for every instance* I*, it holds that* $A_\varepsilon(I) \leq (1+\varepsilon)\mathrm{OPT}(I)$*.*

In the context of approximation algorithms, we frequently have algorithms with asymptotic approximation factors. We say an approximation algorithm has *asymptotic approximation factor* α if, for every instance I, it holds that $\lim_{\mathrm{OPT}(I)\to\infty} \mathcal{A}(I)/\mathrm{OPT}(I) \leq \alpha$. We then say that $\mathcal{A}$ is an *asymptotic approximation algorithm.* A simpler way to describe an asymptotic approximation factor is stating that $\mathcal{A}(I) \leq \alpha\mathrm{OPT}(I) + \lambda$ for instance I, where λ is a positive constant. We can also have asymptotic approximation schemes, i.e., approximations of the form $(1+\varepsilon)\mathrm{OPT}(I) \pm \lambda$, where λ is a positive constant.

Definition 10. *An* asymptotic polynomial-time approximation scheme (APTAS) *is a family of polynomial time algorithms* $\{\mathcal{A}_\varepsilon\}$*, in which there is an algorithm* A_ε *for every* $\varepsilon > 0$ *such that, for every instance* I*, it holds that* $A_\varepsilon(I) \leq (1+\varepsilon)\mathrm{OPT}(I) + \lambda$*, where* λ *is a positive constant.*

Both schemes we presented so far are good results, however, the running time depends arbitrarily on ε. Then, as the value of ε decreases, the constant behind the running time of the algorithm may increase too much. So it is desirable that the running time of the algorithm could be polynomial not only on the size of the instance, but also on $1/\varepsilon$.

Definition 11. *A* fully polynomial-time approximation scheme (FPTAS) *is family of algorithms* $\{\mathcal{A}_\varepsilon\}$*, in which there is an algorithm* A_ε*, polynomial on the size of the instance and on* $1/\varepsilon$*, for every* $\varepsilon > 0$*, and such that, for every instance* I*, it holds that* $A_\varepsilon(I) \leq (1+\varepsilon)\mathrm{OPT}(I)$*.*

Chapter 3

NP-hardness

In the first section of this chapter we present a proof of the NP-hardness of the In-line Circle Packing (ICP) as well as a brief comment about some hints on the difficulty of proving whether or not it is in NP. In the second section we present a proof of the NP-hardness of the In-line Circle Bin Packing (ICB), which is heavily based on the proof shown for the ICP.

3.1 NP-hardness of the In-line Circle Packing problem

Here we discuss the complexity of the In-line Circle Packing (ICP) problem. Recall that an in-line packing is given by a function whose domain is the input list of circles and the image is a set of rational numbers. Note that the ICP problem may not admit optimal solutions whose base positions are all rational. For example, if we have only two circles: circle i of radius 2 and circle j of radius 1. Then there are two possible optimal packing: a packing $\mathcal{P}_1$ where i is the first circle and j is touching i, and a packing $\mathcal{P}_2$ where j is the first circle and i is touching j. In the first packing we have $\mathcal{P}_1(j) = 2 + 2\sqrt{2}$, while in the second packing we have $\mathcal{P}(i) = 1 + 2\sqrt{2}$. This implies that the base positions of optimal solutions may be irrational, which in turn gives some insight into the difficulty that is to compare the sizes of two packings, as we still do not know how to perform basic operations and comparisons with irrational numbers. For instance, suppose we have n circles and a sequence $p_1, \ldots, p_n$ of the order in which the circles were packed, where p_i is the base position of the i-th circle in the packing, from left to right. Without loss of generality, we refer to the circle at position p_i as i, and therefore it has radius r_i. Each base position p_i can be written in the form $a+b\sqrt{c}$, where a, b, and c are rational numbers. By Lemma 2.1, the distance between the base positions of two touching circles i and j is given by $2\sqrt{r_i r_j}$. Thus, to obtain the size of a packing, we could go through the circles of the packing that touch each other, starting from the circle that defines its left boundary, and repeatedly sum the distance between the previous and the current circles. We repeat this until we reach the circle that defines the right boundary of the packing. However, this process descends to a case of the problem of sum of square roots, which is still an open problem. Although this does not prove that both problems have the same complexity, it does give us some hints that calculating the size of a packing in the way we described may be as hard as the sum of square roots. We did not find in the literature

any works proposing a valid certificate for the problem, therefore we do not know if the ICP problem is in NP. Nonetheless, Alt et al. [1] proved that the ICP problem is NP-hard. In the following, we present the reduction they showed.

Problem 3 (CirclesInLine)**.** *Given a list C of n circles $1, \ldots, n$ with rational radii and a rational constant $\delta > 0$, is there an in-line packing of C of span at most δ?*

The polynomial reduction is from the *3-Partition Problem* [11], described below, to the ICP problem.

Problem 4 (3-Partition)**.** *Given a multiset $\mathcal{S}$ of $3m$ integers $s_1, \ldots, s_{3m}$ and an integer T such that $\sum_{i=1}^{3m} s_i = mT$ and $\frac{T}{4} < s_i < \frac{T}{2}$, is there a partition of $\mathcal{S}$ into m triplets $S_1, \ldots, S_m$ such that $\sum_{s \in S_j} s = T$, for $j = 1, \ldots, m$?*

Given an instance $(\mathcal{S}, T)$ of 3-Partition, we construct an instance $(\mathcal{C}, \delta)$ of CirclesInLine. We create $\mathcal{C}$ consisting of some circles whose radii are fixed and other circles whose radii depends on the values of the elements in $\mathcal{S}$. The idea is to strategically define values for δ and the radii in a way that, except for symmetry, there is only one packing of span equal to δ, when considering only the fixed radii circles. Such unique packing presents a certain pattern that is repeated m times. In each repetition of the pattern it is possible to pack three circles corresponding to three elements of $\mathcal{S}$. Then each repetition of the pattern represents a triplet. For the purpose of simplifying some calculations, we define the *size* of a circle i as the square root of its radius. Based on $(\mathcal{S}, T)$, we create $9m + 11$ circles of fixed size, and a corresponding circle for each $s_i \in \mathcal{S}$, giving us a total of $12m + 11$ circles. The size of the packing must be $\delta = 2(m + 1)$, and the radii are determined as follows, in terms of their size.

Definition 12. *Given an instance $(\mathcal{S}, T)$ of the* 3-Partition *problem, consider an instance of the* CirclesInLine *problem as follows. Set δ equals to $2(m + 1)$, and create $12m + 11$ circles distributed in six different types, as described below.*

i) Outer frame circles*: $m + 1$ circles of size 1;*

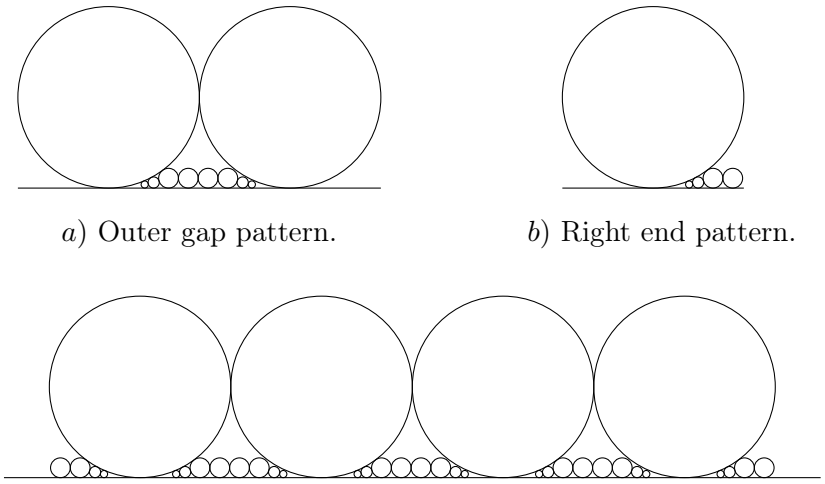
ii) inner frame circles*: $4(m + 1)$ circles of size $z_0 = \dfrac{33}{100} = 0.33$;*

iii) large filler circles*: $2(m + 1)$ circles of size $z_1 = \dfrac{z_0}{1 + z_0} = \dfrac{33}{133} \approx 0.24812$;*

iv) small filler circles*: $2(m + 1)$ circles of size $z_2 = \dfrac{z_1}{1 + z_1} = \dfrac{33}{166} \approx 0.198795$;*

v) end circles*: 2 circles of size $z_3 = \dfrac{1 - z_0^2 - 2z_0}{4z_0} = \dfrac{2311}{13200} \approx 0.175076$;*

vi) partition circles*: $3m$ circles $1 \ldots 3m$ with sizes $t_i = \dfrac{17}{99}\left(\dfrac{3s_i}{100T} + \dfrac{99}{100}\right)$.*



a) Outer gap pattern. *b*) Right end pattern.

c) Example of a packing with $m = 3$.

Figure 3.1: Illustration of the unique pattern of a packing with span $2(m + 1)$.

The purpose of the outer frame circles is to guarantee that we have a packing of size at least $2(m + 1)$. The remaining circles are defined so that all of them can be packed in the spaces between outer frame circles. We also need to assure that between two consecutive outer frame circles there are exactly three partition circles, and to achieve that we must use the inner frame, small filler and large filler circles to fill in the spaces between outer frame circles. Figure 3.1 illustrates this idea. We observe that an instance of CirclesInLine created as described in Definition 12 is a valid instance and can be obtained in polynomial time. The construction is linear on m, and all circles in $\mathcal{C}$ have rational sizes and δ is also a rational number. Thus, $(\mathcal{C}, \delta)$ meets the criteria for an instance of CirclesInLine.

Once we have created an instance of CirclesInLine, it remains to show that $(\mathcal{S}, T)$ is a YES instance for 3-Partition if and only if $(\mathcal{C}, \delta)$ is a YES instance for CirclesInLine. Prior to this, however, we need to present a sequence of lemmas. Let $\mathcal{P}$ be an optimal in-line packing of $\mathcal{C}$. As mentioned previously, an optimal packing follows a very specific structure (that is proved further on the text). The largest circles in $\mathcal{C}$ are the outer frame circles and they have size 1. Since there are $m + 1$ of them, the size of $\mathcal{P}$ is at least $2(m + 1)$. Then a packing of $\mathcal{C}$ of size equal to $2(m+1)$ requires that i) every two consecutive outer frame circles touch each other, and ii) that all the remaining circles fit into the free spaces between pairs of consecutive outer frame circles. We call *outer gap* the space between the base positions of two consecutive outer frame circles, *left end* the space to the left of the base position of the leftmost circle and *right end* the space to the right of the base position of the rightmost circle. For the sole purpose of simplicity, whenever regarding the ends we use the right end as reference. All arguments are analogous for the left end, except if otherwise expressed. We start by packing only the outer and filler circles. The partition circles will be added to the packing once the outer and filler circles were already packed. From now on we might sometimes need to refer to a sequence of circles where the importance lays on their sizes, not on the circles themselves. For that we define a *sequence of sizes* as the representation

of a sequence of circles by their sizes. For example, suppose we have a list C of circles c_1, c_2, c_3, c_4, c_5 with sizes 1, 1, 2, 3 and 2, respectively; if the sequence $P = (c_1, c_4, c_2, c_3, c_5)$ represents a packing of C, then $(1, 3, 1, 2, 2)$ is a sequence of sizes of P. We say P has *span* α if the size of the packing represented by P is α.

Lemma 3.1. *There is only one possible packing of the frame and filler circles of size* $2(m+1)$*, except for symmetries.*

Such unique packing consists of all outer frame circles packed side by side so that every two consecutive circles touch. Then in each outer gap, there is a sequence of sizes $(z_2, z_1, z_0, z_0, z_0, z_0, z_1, z_2)$. In the left end, there is a sequence of sizes (z_0, z_0, z_1, z_2), and in the right end, there is a sequence of sizes (z_2, z_1, z_0, z_0). Figure 3.1 illustrates such patterns. In order to prove that such structure is the same for every packing of span $2(m+1)$, we show, through a sequence of claims, that all other possibilities yield a packing of larger span. As argued previously, the outer frame circles must be packed side by side and such that every two consecutive circles touch each other. So we suppose the outer frame circles are packed in this way, and analyze ways to add the remaining (inner frame and filler) circles to a packing of only the outer frame circles.

Claim. *Each outer gap contains four inner frame circles and each end contains two inner frame circles.*

Proof. We show that for an optimal packing to have size $2(m+1)$, we cannot have more than four inner frame circles (the ones of size z_0) in an outer gap. Since the size of an outer frame circle is 1, the span of each outer gap is 2. Suppose there is one outer gap containing five inner frame circles. The span of a sequence of sizes $(1, z_0, z_0, z_0, z_0, z_0, 1)$ is $4z_0+8z_0^2 = 2.1912$. That implies the two outer frame circles forming the outer gap do not touch, a contradiction. However, the span of the sequence $(1, z_0, z_0, z_0, z_0, 1)$ is $4z_0 + 6z_0^2 = 1.9734$, implying that we can pack up to four inner frame circles in an outer gap. Now we consider the left and right ends. The total span of the inner frame circles packed in each of the ends has to be at most 1, otherwise the total span of the packing would be greater than $2(m + 1)$. The spans of the sequences of sizes $(1, z_0, z_0, z_0)$ and $(1, z_0, z_0)$ are $2z_0 + 5z_0^2 = 1.2045$ and $2z_0 + 3z_0^2 = 0.9867$, respectively. This in turn implies that we can pack up to two inner frame circles in the right end. Analogously, the same holds for the left end.

Finally, because there is a total of $4(m + 1)$ inner frame circles to be distributed among m outer gaps and two ends, it can be inferred that each outer gap contains exactly four inner frame circles and each end contains exactly two inner frame circles. □

So far, we have packed the outer and inner frame circles. Now, we add the filler circles to the current packing. We start with the large filler circles. By definition, a large filler circle has the radius of a largest circle that fits between an outer frame circle and an inner frame circle that touch each other. This follows directly from Lemma 2.2. We show that, in an outer gap, there is only two spaces where packing a large filler circle does not increase the total span. Moreover, we show that each outer gap must contain exactly two large filler

Table 3.1: Impossible packings of a large filler circle.

Place	Sequence of sizes	Span
Outer gap	$(1, z_0, z_1, z_0, z_0, z_0, 1)$	2.08312
	$(1, z_1, z_1, z_0, z_0, z_0, z_0, 1)$	2.09653
Right end	$(1, z_0, z_1, z_0)$	1.09642
	$(1, z_1, z_1, z_0, z_0)$	1.10983
	$(1, z_0, z_0, z_1)$	1.00845

circles. Similarly, in each end, there is only one way to pack a large filler circle without causing any increase in the total span, and each end contains exactly one large filler circle.

Claim. *Each outer gap contains exactly two large filler circles and each end contains exactly one large filler circle.*

Proof. We want to show that the only space where a large filler circle can be packed without increasing the total span is the space between an outer frame circle and an inner frame circle. Let us consider the other possibilities. First, suppose a large filler circle is packed between two inner frame circles. Without loss of generality suppose the large filler circle was packed between the first two inner frame circles. Then we have a sequence of sizes $(1, z_0, z_1, z_0, z_0, z_0, 1)$ which has span $4z_0 + 4z_0z_1 + 4z_0^2 = 2.08312$, implying a total span larger than $2(m+1)$. Now suppose we pack two large filler circles consecutively. We already know they cannot be between two inner frame circles, so they must be between an outer frame and an inner frame. Without loss of generality, we would have a sequence of sizes $(1, z_1, z_1, z_0, z_0, z_0, z_0, 1)$. Such sequence has span $2z_1 + 2z_1^2 + 2z_1z_0 + 6z_0^2 + 2z_0 = 2.09653$, implying a total span larger than $2(m + 1)$. Lastly, suppose one large filler circle is packed between an outer frame circle and an inner frame circle. Since a large filler circle fits in this space, the total span remains the same. Hence this is the only space where we can pack a large filler circle, entailing that each outer gap contains at most two circles of this type.

The analysis is analogous with the ends. The only space where a large filler circle can be packed in the ends is between an outer frame circle and an inner frame circle, entailing that each end can contain at most one large filler circle. Since there are $2(m + 1)$ large filler circles, each outer gap must contain exactly two large filler circles, and each end must contain exactly one large filler circle. □

Table 3.1 summarizes the impossible packings of a large filler circle inside an outer gap and in the right end. Finally, we now analyze the packing of the small filler circles. By definition of its size, a small filler circle is a largest circle that fits between an outer frame circle and a large filler circle. We show that this is the only space where a small filler circle can be packed without increasing the total span. In addition, we ascertain that each outer gap contains exactly two small filler circles and each end contains exactly one small filler circle. The proof follows the same reasoning used for determining the packing of the large filler circles.

Table 3.2: Impossible packings of a small filler circle.

Place	Sequence of sizes	Span
Outer gap	$(1, z_0, z_2, z_0, z_0, z_0, 1)$	2.01801
	$(1, z_1, z_2, z_0, z_0, z_0, z_0, 1)$	2.03949
	$(1, z_2, z_2, z_1, z_0, z_0, z_0, z_0, 1)$	2.05244
Right end	$(1, z_0, z_2, z_0)$	1.03131
	$(1, z_1, z_2, z_0, z_0)$	1.05279
	$(1, z_0, z_0, z_2)$	1.04852
	$(1, z_2, z_2, z_1, z_0, z_0)$	1.06574

Claim. *Each outer gap contains exactly two small filler circles and each end contains exactly one small filler circle.*

Proof. Firstly, we regard the cases in an outer gap. Suppose a small filler circle is packed between an outer frame circle and a large filler circle. Since it fits in this space, the total span does not change. Now we consider the remaining possibilities of how to pack a small filler circle. Packing a small filler circle between two inner frame circles gives us a sequence of sizes $(1, z_0, z_2, z_0, z_0, z_0, 1)$, which has span $4z_0 + 4z_0z_2 + 4z_0^2 = 2.01801$, implying a total span larger than $2(m + 1)$. Now if we pack a small filler circle between a large filler circle and an inner frame circle, we obtain a sequence of sizes $(1, z_1, z_2, z_0, z_0, z_0, z_0, 1)$. The span is $2z_1 + 2z_1z_2 + 2z_2z_0 + 6z_0^2 + 2z_0 = 2.03949$, also implying a total span larger than $2(m+1)$. Lastly, if we pack two small filler circles consecutively, then we have a sequence of sizes $(1, z_2, z_2, z_1, z_0, z_0, z_0, z_0, 1)$, which has span $2z_2 + 2z_2^2 + 2z_2z_1 + 2z_1z_0 + 6z_0^2 + 2z_0 = 2.05244$, again implying a total span larger than $2(m+1)$. These restrictions impose that each outer gap must contain at most two small filler circles. For the cases of both the right and the left ends the proof is analogous, entailing that each end must contain at most one small filler circle. To conclude, because we have $2(m + 1)$ small filler circles, m outer gaps and two ends, we can infer that each outer gap contains exactly two small filler circles and each end contains exactly one small filler circle. □

Table 3.2 summarizes the ways we cannot pack a small filler circle without increasing the total span. The sequence of three claims presented previously proves Lemma 3.1.

Up to this point we have a packing of the outer and filler circles, of size $2(m+1)$. It remains to pack the end and partition circles in the spaces remaining in the packing constructed so far. In order to know whether or not an end or partition circle fits in a certain gap, we must analyze their size. The end circles have fixed size equals to 0.175076. The sizes of the partition circles, however, vary according to the values of the multiset $\mathcal{S}$ from the input of the 3-Partition problem. By definition of the problem, we have $\frac{1}{4} < \frac{s_i}{T} < \frac{1}{2}$ for every element $s_i \in \mathcal{S}$. This restriction gives us bounds on the sizes, given in Definition 12, item (iv), of the partition circles. Thus, for every $i = 1, \ldots, 3m$, we have

$$0.17129 \approx \frac{17}{99}\left(\frac{3}{400} + \frac{99}{100}\right) < t_i < \frac{17}{99}\left(\frac{3}{200} + \frac{99}{100}\right) \approx 0.17257.$$

The idea is to pack three partition circles in each outer gap, each one of these three between two inner frame circles, and to pack one end circle at each end, also between two consecutive inner frame circles. However, by Lemma 2.2, the largest circle that fits between two inner frame circles has size $z_0/2 = 0.165$. Hence, neither the end circles nor the partition circles fit between two inner frame circles. Nonetheless, we may still be able to pack the circles as described previously by allowing the inner frame circles to be shifted to the right by a small amount when a partition circle is added. This is possible because the sequence of sizes $(1, z_0, z_0, z_0, z_0, 1)$ has span $1.9734 < 2$. Thus, the inner frame circles are not "tight" in the space between two outer frame circles. The following lemma gives us a bound on the sum of the size of three partition circles so that they can be packed in an outer gap, each between two consecutive inner frame circles, without increasing the total size of the packing.

Lemma 3.2. *Let i, j and k be three partition circles. Packing i, j and k in a common outer gap, each between two consecutive inner frame circles does not increase the total size of the packing if and only if $t_i + t_j + t_k \leq \frac{17}{33}$.*

Proof. Due to its size, a partition circle packed in a gap between two inner frame circles must touch both circles forming that gap. Then, we have a packing represented by the sequence of sizes $(1, z_0, t_i, z_0, t_j, z_0, t_k, z_0, 1)$ whose span is $4z_0 + 4z_0t_i + 4z_0t_j + 4z_0t_k$. The suggested packing does not increase the total span if $4z_0(t_i + t_j + t_k + 1) \leq 2$, which implies $t_i + t_j + t_k \leq \frac{17}{33}$, proving the lemma. □

As for the left and the right end, we wish to pack an end circle, also between two consecutive inner frame circles, in each end. The following lemma gives us a bound on the size of a circle so that it can be packed in one of the ends without increasing the total size of the packing.

Lemma 3.3. *Packing a circle i of size z_i between two consecutive inner frame circles in the right or the left end increases the total span if and only if $z_i > \frac{2311}{13200}$.*

Proof. If $z_i \leq z_0/2$, then i fits between two consecutive touching inner frame circles, hence the total span does not change. Suppose now that $z_i > z_0/2$. Then we have a sequence of sizes $(1, z_0, z_i, z_0)$ where every two consecutive circles touch each other. For the total span to increase, we must have $2z_0 + 4z_0z_i + z_0^2 > 1$, which implies $z_i > \frac{2311}{13200}$, proving the lemma. □

Having knowledge of these bounds, we shall see that if we pack an end circle between two inner frame circles in the end, then the total size of the packing does not increase. Similarly, if we pack three partition circles in an outer gap, each between two consecutive inner frame circles, then the total size of the packing also does not increase. More specifically, we show that each end can contain up to one end circle and that each outer gap can contain up to three partition circles. The idea is to analyze all different ways how to pack an end or partition circle and discard the ones that yield a sequence with span greater than two in the outer gaps, or greater than one in the ends. Table 3.3 summarizes the impossible packings of an end circle while Table 3.4 summarizes the impossible packings of a partition circles.

Claim. *In each end it is possible to pack at most one end or partition circle without increasing the total span.*

Table 3.3: Impossible packings of an end circle.

Place	Sequence of sizes	Span
Outer gap	$(1, z_3, z_3, z_2, z_1, z_0, z_0, z_0, z_0, 1)$	2.05687
	$(1, z_2, z_3, z_1, z_0, z_0, z_0, z_0, 1)$	2.03124
	$(1, z_1, z_3, z_0, z_0, z_0, z_0, 1)$	2.01207
	$(1, z_0, z_3, z_3, z_0, z_0, z_0, 1)$	2.04800
	$(1, z_3, z_2, z_1, z_0, z_3, z_0, z_0, z_0, 1)$	2.00887
	$(1, z_3, z_2, z_1, z_0, z_0, z_0, z_0, z_1, z_2, z_3, 1)$	2.01774
Right end	$(1, z_3, z_2, z_1, z_0, z_0)$	1.00887
	$(1, z_2, z_3, z_1, z_0, z_0)$	1.04454
	$(1, z_1, z_3, z_0, z_0)$	1.02537
	$(1, z_0, z_3, z_3, z_0)$	1.06130
	$(1, z_3, z_2, z_1, z_0, z_3, z_0)$	1.01458
	$(1, z_0, z_0, z_3)$	1.02400

Proof. We analyze every possibility. Let i and j be two partition circles of size t_i and t_j, respectively. Then the size of i and j is bounded by $0.17129 < t_i, t_j < 0.17257$. Consider packing the partition circle i between an outer frame circle and a small filler circle. By Lemma 2.2, the largest circle that fits in such space has size $33/199 \approx 0.16583$. Hence i does not fit in this space, so in the packing represented by a sequence of sizes $(1, t_i, z_2, z_3, z_0, z_0)$, every two consecutive circles are touching each other. The span of this sequence is at least 0.99977 and at most 1.00286, implying that there might be a partition circle which can be packed in this space. However, we cannot have two consecutive partition circles, say i and j, in this space, since the span of the sequence of sizes $(1, t_i, t_j, z_2, z_3, z_0, z_0)$ is at least 1.05844. Likewise, we cannot have a partition circle followed by an end circle, because the sequence of sizes $(1, t_i, z_3, z_2, z_1, z_0, z_0)$ has span at least 1.06125.

Now consider packing the partition circle i between two consecutive inner frame circles. Since i does not fit in this space, then i touches both circles, and so we have a sequence of sizes $(1, z_0, t_i, z_0)$ whose span is at most 0.99669. Therefore, we can pack i in such space. Again, we cannot have two consecutive partition circles in this space, since a sequence of sizes $(1, z_0, t_i, t_j, z_0)$ is at least 1.05366. There is still the possibility of packing one partition circle in each of the two places where it fits, i.e., a sequence of sizes $(1, t_i, z_2, z_1, z_0, t_j, z_0)$. Its span is at least 1.00806, causing increase in the total size of the packing. See the bottom of Table 3.4 for the impossible placements of a partition circle in the right end.

For the end circles, the proof follows the same steps, with exception of the fact that a sequence of sizes $(1, z_3, z_2, z_1, z_0, z_0)$ yields a span 1.00887, so an end circle cannot be packed between an outer frame circle and a small filler circle. As a consequence, in both ends, an end circle can only be packed between two inner frame circles, and we cannot have more than one end circle. We conclude, then, that in each end it can be packed up to one partition circle or one end circle, and the result follows. □

Claim. *In an outer gap it is possible to pack at most three partition or at most two end circles without increasing the total span.*

Proof. We start the analysis with the partition circles. Let i, j and k be three partition circles. There are four spaces where we can place i, j and k. We analyze every possibility dividing in four cases.

Case 1: packing i between an outer frame circle and a small filler circle. Since i does not fit in this space, it follows that i touches both the outer frame and the small filler circles. Then the sequence of sizes $(1, t_i, z_2, z_1, z_0, z_0, z_0, z_0, 1)$ has span $2t_i+2t_iz_2+2z_2z_1+2z_1z_0+6z_0^2+2z_0$, which is greater than 1.98647 and lesser than 1.98956. So placing one partition circle in this space does not cause the total span to increase. However, there cannot be two consecutive partition circles, say i and j, in this space, because the sequence of sizes $(1, t_i, t_j, z_2, z_1, z_0, z_0, z_0, z_0, 1)$ has span greater than 2.04514.

Case 2: packing i between a small filler circle and a large filler circle. The sequence of sizes $(1, z_2, t_i, z_1, z_0, z_0, z_0, z_0, 1)$ has span greater than 2.02784, therefore no partition circle can be placed in this space.

Case 3: packing i between a large filler circle and an inner frame circle. The sequence of sizes $(1, z_1, t_i, z_0, z_0, z_0, z_0, 1)$ has span greater than 2.00768, therefore no partition circle can be placed in this space.

Case 4: packing circle i between two consecutive inner frame circles. In this case, the sequence $(1, z_0, t_i, z_0, z_0, z_0, 1)$ has span $4z_0 + 4z_0t_i + 4z_0^2$, which is greater than 1.98169 and lesser than 1.98339. As a consequence, we can pack one partition circle in this space without causing increase in the total size of the packing. Again, we cannot place two consecutive partition circles, say i and j, in this space, because the sequence $(1, z_0, t_i, t_j, z_0, z_0, z_0, 1)$ has span at least 2.04036. This implies that between two inner frame circles at most one partition circle can be packed.

Now that we know where one partition circle can be packed we analyze how to pack three of them. Consider packing one partition circle between an outer frame circle and a small filler circle, and two partition circles each between two consecutive inner frame circles. This give us a sequence of sizes $(1, t_i, z_2, z_1, z_0, t_j, z_0, t_k, z_0, z_0, 1)$, which has span greater than 2.00305. Now consider packing two partition circles between an outer frame circle and a small filler circle, and one partition circle between two consecutive inner frame circles. The span of the sequence $(1, t_i, z_2, z_1, z_0, z_0, t_j, z_0, z_0, z_1, z_2, t_k, 1)$ is greater than 2.00783. Finally, consider three partition circles, each packed between two consecutive inner frame circles. The span of the sequence of sizes $(1, z_0, t_i, z_0, t_j, z_0, t_k, z_0, 1)$ is at least 1.99823, meaning that there might exist three partition circles that can be placed in this space without increasing the total size of the packing. Such sequence has span at most 2.00338, but this does not exclude the possibility of having three partition circles that can be packed in this space without increasing the total span. By Lemma 3.2, if $t_i + t_j + t_k < 17/33$, then the span of the sequence of sizes $(1, z_0, t_i, z_0, t_j, z_0, t_kz_0, 1)$ is at most 2. We then conclude that it is possible to pack at most three partition circles in an outer gap, and moreover, if we place exactly three, then each must be between two consecutive inner frame circles.

Finally, we now analyze the end circles. Akin to the case of partition circles discussed

Table 3.4: Impossible packings of partition circles.

Place	Sequence of sizes	Span	
		At least	At most
Outer gap	$(1, t_i, t_j, z_2, z_1, z_0, z_0, z_0, z_0, 1)$	2.04514	2.04912
	$(1, z_2, t_i, z_1, z_0, z_0, z_0, z_0, 1)$	2.02784	2.02899
	$(1, z_1, t_i, z_0, z_0, z_0, z_0, 1)$	2.00768	2.00917
	$(1, z_0, t_i, t_j, z_0, z_0, z_0, 1)$	2.04036	2.04295
	$(1, t_i, z_2, z_1, z_0, t_j, z_0, t_k, z_0, z_0, 1)$	2.00305	2.00955
	$(1, t_i, z_2, z_1, z_0, z_0, t_j, z_0, z_0, z_1, z_2, t_k, 1)$	2.00783	2.01571
Right end	$(1, t_i, t_j, z_2, z_1, z_0, z_0)$	1.05844	1.06242
	$(1, z_2, t_i, z_1, z_0, z_0)$	1.04114	1.04230
	$(1, z_1, t_i, z_0, z_0)$	1.02098	1.02247
	$(1, z_0, t_i, t_j, z_0)$	1.05366	1.05625
	$(1, z_0, z_0, t_i)$	1.02018	1.02148
	$(1, t_i, z_2, z_1, z_0, t_j, z_0)$	1.00806	1.01027

previously, one end circle can be packed without increasing the total size of the packing either between an outer frame circle and a small filler circle, or between two consecutive inner frame circles. We also cannot have consecutive end circles in either spaces. There are three different ways to place two end circles, and the only one that does not cause increase in the total size of the packing is given by sequence of sizes $(1, z_0, z_3, z_0, z_3, z_0, z_0, 1)$. See Table 3.3 for the impossible placements of an end circle. □

Theorem 3.4. *There is a partition of $\mathcal{S}$ into m triplets $S_1, \ldots, S_m$ such that $\sum_{s \in S_j} s = T$ for all $j = 1, \ldots, m$ if and only if there is an in-line packing of $\mathcal{C}$ of span at most $2(m+1)$.*

Proof. ($\Rightarrow$) Suppose $\mathcal{S}$ can be partitioned into m triplets $S_1, \ldots, S_m$ such that $\sum_{s \in S_j} s = T$ for all $j = 1, \ldots, m$. Let $S_l = (s_i, s_j, s_k)$ be one arbitrary triplet, and let i, j and k be the partition circles associated with elements s_i, s_j and s_k, respectively.

Let $\mathcal{P}$ be a packing of $\mathcal{C}$. By Lemma 3.1, the frame and filler circles are packed respecting the pattern shown in Figure 3.1. Moreover, because of the outer frame circles, the size of $\mathcal{P}$ is $2(m+1)$. It remains to consider the partition and end circles. By Lemma 3.3, packing an end circle between two inner frame circles in an end does not increase the total size of $\mathcal{P}$. Since there are two end circles, we can place one in each end. For the partition circles, consider the sum of their sizes.

$$\begin{aligned} t_i + t_j + t_k &= \frac{17}{99}\left(\frac{3(s_i + s_j + s_k)}{100T} + 3\frac{99}{100}\right) \\ &= \frac{17}{99}\left(\frac{3T}{100T} + 3\frac{99}{100}\right) \\ &= \frac{17}{33}. \end{aligned}$$

Then, by Lemma 3.2, we can pack i, j and k in a common outer gap, each between two consecutive inner frame circles. Since there are m outer gaps and m triplets, we can pack the circles corresponding to each triplet in an outer gap without causing the total span of $\mathcal{P}$ to increase. Hence the packing $\mathcal{P}$ imposes that $\mathcal{C}$ is a YES instance of CirclesInLine.

($\Leftarrow$) Suppose now that there exists a packing $\mathcal{P}$ of $\mathcal{C}$ of size at most $2(m+1)$. Observe that, in this case, the packing has size exactly $2(m+1)$, imposed by the outer gap circles. We show that each outer gap contains exactly three partition circles, and furthermore, that the three circles in an outer gap correspond to one triplet of a 3-partition of $\mathcal{S}$. Each end contain at most one end or partition circle and each outer gap contains at most three end or partition circles. Then, because there are $3m+2$ end and partition circles in total, we have that each end contains exactly one end or partition circle, and each outer gap contains exactly three end or partition circles.

By Lemma 3.3, the end or partition circle packed in an end must have size at most $\frac{2311}{13200}$, which is the exact size of an end circle. Thus, we can assume, without loss of generality, that each end circle is packed in one of the two ends.

Now consider an arbitrary outer gap. Let i, j and k be the three partition circles packed there. Let s_i, s_j and s_k be the three values of the multiset $\mathcal{S}$ corresponding to the circles i, j and k. Since each of the partition circles must be packed between two consecutive inner frame circles, we can apply Lemma 3.2 and obtain the following inequality.

$$\begin{aligned} t_i + t_j + t_k &= \frac{17}{99}\left(\frac{3(s_i+s_j+s_k)}{100T} + 3\frac{99}{100}\right) \\ &\leq \frac{17}{33}. \end{aligned}$$

From that we conclude that $s_i + s_j + s_k \leq T$. Since the same holds for every gap, it follows that we can partition $\mathcal{S}$ into m triplets $S_1, \ldots, S_m$ such that $\sum_{s \in S_j} s \leq T$, for $j = 1, \ldots, m$. In addition, the fact that $\sum_{s \in \mathcal{S}} s = mT$ implies that $\sum_{s \in S_j} s = T$, for $j = 1, \ldots, m$. Hence, $(\mathcal{S}, T)$ is a YES instance of 3-Partition. □

3.2 NP-hardness of the In-line Circle Bin Packing problem

We now prove that the ICB problem is NP-hard.

Theorem 3.5. *The problem* ICB *is NP-hard.*

Proof. Given an instance for the decision version of the ICP problem, $I_{\text{ICP}} = (L, r, \delta)$, we can construct an instance $I_{\text{ICB}} = (L, r, \delta)$ for the ICB problem. Note that the parameter δ in I_{ICB} refers to the width of the bins, while, in I_{ICP}, δ refers to the size of the in-line packing. It is straightforward to see that I_{ICB} has a solution using only one bin if and only if I_{ICP} has an in-line packing of size at most δ (i.e., I_{ICP} is a YES instance for the ICP problem). □

This theorem tells us that deciding if a list of circles admits an in-line bin packing using only one bin is NP-hard. This leads to the following corollary.

Corollary 3.6. *There is no polynomial-time approximation algorithm for the* ICB *problem with approximation factor strictly less than* 2*, unless* $P = NP$.

Proof. The proof is by contradiction. Suppose there exists a polynomial time algorithm $\mathcal{A}$ with approximation factor $\alpha < 2$ for the ICB problem, i.e., $\mathcal{A} \leq \alpha \mathrm{OPT}(I)$, for any instance I. If the instance (L, r, δ) can be packed in one bin, i.e., $\mathrm{OPT}(L, r, \delta) = 1$, then the algorithm $\mathcal{A}$ will use at most α bins to pack L. Since the number of bins is an integer, and $\alpha < 2$, then $\mathcal{A}$ will use exactly one bin to pack L. Otherwise, $\mathrm{OPT}(L, r, \delta) \geq 2$, and $\mathcal{A}$ will use at least 2 bins. Therefore, (L, r, δ) can be packed in one bin if and only if $\mathcal{A}(L, r, \delta) = 1$, and so we could decide an NP-hard problem in polynomial time. □

Chapter 4

In-line Packing of Circles

4.1 A 4/3-approximation

In this section, we present a 4/3-approximation for the In-line Circle Packing problem, shown by Helmut et al. [1].

In the following, we give an overview of how the algorithm works, as well as some insight on the proof of the approximation factor. Let L be a list of circles and a packing of L. The algorithm first sorts L in non-increasing order of radius. Then it packs each circle $i \in L$, respecting the sorting order and according to the following rules. If i fits into a gap between two consecutive circles in the current packing, then it packs i in this gap, touching the smaller circle out of the two circles forming the gap. Otherwise, the algorithm packs the circle i either in the beginning or in the end of the current packing. If the first circle has radius larger than the radius of the last circle, then the algorithm packs i touching the first circle from the left; otherwise, the algorithm packs i touching the last circle from the right. Note that, because of the sorting, packing the circle i in one of the extremes of the current packing does not cause overlaps. That happens because the current circle being packed is never larger than any of the circles that are already packed. So if the previous circle did not caused overlaps, nor does the current one. For the proof of the constant approximation factor, the authors introduce a notion of intervals, which they call support intervals, associated to the circles. Then they consider the length of such intervals to obtain a lower bound on the optimum. Finally, they show that, in every situation, the sum of the length of the support intervals is at least 3/4 of the total span, leading to a 4/3-approximation.

As for the algorithm, the authors propose the use of a priority queue that stores the largest circle that fits in the gap between each two consecutive circles in the current packing. We present a possible implementation, making use of a max heap, which we call Q, to maintain the gaps in the current packing. In this way, each element of the heap is a tuple (r_i, x_i, r_j, x_j), where x_i and x_j are the base positions of the circles forming the gap, and $x_i < x_j$, that is, circle i appears to the left of circle j. To define the priority of each gap, the heap uses a function that gives the radius of a largest circle that fits in a given gap. This function, described in function 1, is named max_circle. Such function receives a gap as its only parameter, and calculates the radius of a largest circle that fits in the gap as shown in

Lemma 2.2.

Function 1: max_circle

Input: A gap (r_l, x_l, r_r, x_r).
Output: The radius of a maximum circle that fits in the gap.

1 **return** $\sqrt{\frac{\sqrt{r_r r_l}}{\sqrt{r_r}+\sqrt{r_l}}}$

We note that, when referring to the operations related to heaps, we are employing the patterns adopted by Cormen et al. [6] for maximum priority queues. More specifically, the functions we refer to are named MAX-HEAP-INSERT, HEAP-MAXIMUM and HEAP-EXTRACT-MAX. Aiming to maintain this text as self-contained as we can, we bring a brief description of these functions.

MAX-HEAP-INSERT(S, e): adds the element e to the priority queue S.

HEAP-MAXIMUM(S): returns an element with maximum value in S.

HEAP-EXTRACT-MAX(S): removes and returns an element of maximum value in S.

In Algorithm 2, we described the implementation we are proposing. Since we want to prove that the algorithm Greedy is a constant factor approximation for the ICP problem, we shall first assure that the algorithm is polynomial. In the first step, the list L is sorted in non-increasing order of radius. For this purpose it can be used an algorithm with time complexity $O(n \log n)$, where n is the number of circles. Next, steps 2 to 5 all take constant time. In step 6, it is called the subroutine MAX-HEAP-INSERT, which in turn has time complexity $O(\log n)$. Lastly, the loop in step 7 iterates on $n-2$ circles, and in each iteration, it calls the subroutine MAX-HEAP-INSERT, and possibly it also calls the subroutine HEAP-EXTRACT-MAX. The first, as we said previously, takes time $O(\log n)$ while the latter takes constant time. Thus, the loop takes time $O(n \log n)$. From the analysis of the steps, we can conclude that the algorithm Greedy has time complexity equals to $O(n \log n)$.

Now before we move to proving the approximation factor, we need to introduce some intermediate results. Without loss of generality, we suppose that the last circle increased the span of the packing, and we refer to such circle as the n-th circle in the list L. Also, we consider the circles in L are scaled so that the smallest circle has radius equal to one, i.e., $r_n = 1$.

Definition 13. *Consider a list L and a packing $\mathcal{P}$ of L, and let i be a circle in $\mathcal{P}$. The* support interval *of i in $\mathcal{P}$, denoted by $\mathcal{P}^{supp}(i)$, is defined as the interval*

$$[\mathcal{P}(i) - 2\sqrt{r_i} + 1; \mathcal{P}(i) + 2\sqrt{r_i} - 1].$$

We denote by $|\mathcal{P}^{supp}(i)|$ the length of the support interval of the circle i. The open support interval *of i is the open version of the support interval, i.e., excluding the extremity points of the interval.*

Algorithm 2: Greedy: a 4/3-approximation for the ICP problem.

Input: A tuple (L, r) where L is a list of circles and r is a function of the radii.
Output: A packing $\mathcal{P}$ of L.

Sort $L = (1, \ldots, n)$ in non-increasing order of radius, i.e., $r_1 \geq \ldots \geq r_n$.
Let Q be a max-heap of gaps, using the function max_circle as its priority criteria.
Let $\mathcal{P}$ be the empty packing.
Pack circle 1 in $\mathcal{P}$, with base position $\mathcal{P}(1) = r_1$.
if $|L| \geq 2$ **then**
    Pack circle 2 touching circle 1 from the right.
    MAX-HEAP-INSERT$(Q, (r_1, \mathcal{P}(1), r_2, \mathcal{P}(2)))$.
    for $k = 3, \ldots, n$ **do**
        if $r_k >$ max_circle(HEAP-MAXIMUM(Q)) **then**
            Let i and j be the leftmost and rightmost circles in $\mathcal{P}$.
            if $r_i > r_j$ **then**
                Pack k touching i from the left.
                MAX-HEAP-INSERT$(Q, (r_k, \mathcal{P}(k), r_i, \mathcal{P}(i)))$.
            else
                Pack k touching j from the right.
                MAX-HEAP-INSERT$(Q, (r_j, \mathcal{P}(j), r_k, \mathcal{P}(k)))$.
        else
            $(r_i, \mathcal{P}(i), r_j, \mathcal{P}(j)) \leftarrow$ HEAP-EXTRACT-MAX(Q).
            Pack k between i and j, touching the smaller one, or the leftmost one in case of ties.
            MAX-HEAP-INSERT$(Q, (r_i, \mathcal{P}(i), r_k, \mathcal{P}(k)))$.
            MAX-HEAP-INSERT$(Q, (r_k, \mathcal{P}(k), r_j, \mathcal{P}(j)))$.
return $\mathcal{P}$

The next lemma addresses some inequalities that will be necessary further on. We stress that, due to the nature of the content regarded in this lemma, the proof we are presenting here is quite similar to the proof presented in the original article.

Lemma 4.1. *The following inequalities hold.*

$$x + y - xy \leq 1, \quad for \quad 0 < x, y \leq 1 \tag{4.1a}$$

$$x + y - xy \geq \frac{3}{4}, \quad for \quad 0 < x, y \leq 1 \quad and \quad x + y \geq 1 \tag{4.1b}$$

$$x + y + xy \geq \frac{7}{9}, \quad for \quad \frac{1}{3} \leq x, y \leq 1 \tag{4.1c}$$

$$\frac{1}{x} + \frac{1}{y} + 2\frac{z-1}{xy} \geq \frac{7}{9}, \quad for \quad 1 \leq x, y, z \leq 3 \quad and \quad (x-z)y \leq x + z \tag{4.1d}$$

$$\frac{3x + y - 1}{2x + xy + 1} \geq \frac{3}{4}, \quad for \quad 1 \leq x \leq \frac{3}{2} \quad and \quad 1 \leq y \leq 4 \tag{4.1e}$$

Proof. We prove each inequality separately.

(4.1a). Consider the function $f(x, y) = x + y - xy$. Its partial derivatives are $\frac{\partial f}{\partial x} = 1 - y$ and $\frac{\partial f}{\partial y} = 1 - x$, which are positive when $x < 1$ and $y < 1$. This implies that the function f is increasing in both x and y-axis in the interval $[0; 1]$. Therefore the point $f(1, 1)$ is a maximum point when considering $0 < x, y \leq 1$, and we have

$$f(x, y) \leq f(1, 1) = 1.$$

(4.1b). Again, consider the function $f(x, y) = x+y-xy$, in the interval $(0; 1]$. The constraint $x + y \geq 1$ implies $y \geq 1 - x$. Then,

$$\begin{aligned} f(x, y) &\geq f(x, 1 - x) \\ &= x + (1 - x) - x(1 - x) \\ &= x^2 - x + 1 \\ &= \left(x - \frac{1}{2}\right)^2 + \frac{3}{4} \\ &\geq \frac{3}{4}, \end{aligned}$$

where the first inequality holds because $0 < x \leq 1$ and the function f is increasing in the interval $(0; 1]$.

(4.1c). Consider the function $g(x, y) = x + y + xy$. Its partial derivatives are $\frac{\partial f}{\partial x} = 1 + y$ and $\frac{\partial f}{\partial y} = 1 + x$, which are positive when $x, y > -1$. Since $x, y \geq \frac{1}{3}$, the function g is increasing in both x and y. Therefore, the minimum point of the function restricted to the interval $\left[\frac{1}{3}; 1\right]$

is $g\left(\frac{1}{3}, \frac{1}{3}\right)$, and so we have

$$\begin{aligned} g(x,y) &\geq g\left(\frac{1}{3}, \frac{1}{3}\right) \\ &= \frac{7}{9} \\ &> \frac{3}{4}. \end{aligned}$$

(4.1d). Consider the function $h(x,y) = \dfrac{1}{x} + \dfrac{1}{y} + 2\dfrac{z-1}{xy}$. The partial derivatives of h are $\frac{\partial h}{\partial x} = -\frac{y-2z-2}{x^2y}$, $\frac{\partial h}{\partial y} = -\frac{x+2z-2}{xy^2}$ and $\frac{\partial h}{\partial z} = \frac{2}{xy}$. Therefore, the partial derivatives for x and y are negative for $x, y, z \geq 1$ and the partial derivative for z is positive for $x, y > 0$. We consider two cases. First, suppose $y \leq 9/4$. When restricted to the intervals $1 \leq x, z \leq 3$ and $1 \leq y \leq 9/4$, the minimum point of h is at $h\left(3, \frac{9}{4}, 1\right)$, and we have

$$\begin{aligned} h(x,y,z) &\geq h\left(3, \frac{9}{4}, 1\right) \\ &= \frac{7}{9}. \end{aligned}$$

Now suppose $y > 9/4$. In this case, we must use the constraint $(x-z)y \leq x+z$, which gives us $z \geq \left(1 - \dfrac{2}{y+1}\right) x$. Then we write a function $h'(x,y)$ as follows

$$\begin{aligned} h'(x,y) &= h\left(x, y, \left(1 - \frac{2}{y+1}\right) x\right) \\ &= \frac{1}{x} + \frac{1}{y} - \frac{2}{xy} + \frac{2(y-1)}{y(y+1)}. \end{aligned}$$

Since the partial derivative of h for z is positive for $x, y > 0$, and because we have $x, y \geq 1$, the minimum point of h has the coordinate z equals to $\left(1 - \frac{2}{y+1}\right) x$. Therefore, $h(x,y,z) \geq h'(x,y)$. Now we analyze h'. Its partial derivative for x is $\frac{\partial h'}{\partial x} = \dfrac{2-y}{x^2y}$, which is negative for $y > 9/4$. Then the x coordinate of the minimum point of h' restricted to the interval $1 \leq x \leq 3$ is equals to 3. Thus,

$$\begin{aligned} h'(x,y) &\geq h'(3,y) \\ &= \frac{1}{3} + \frac{1}{y} - \frac{2}{3y} - \frac{4}{y(y+1)} \\ &= \frac{1}{3} + \frac{1}{3y} + \frac{6(y-1)}{y+1}. \end{aligned}$$

Finally, consider the partial derivatives of $h'(3,y)$ for y. It is given by $\frac{\partial h'(3,y)}{\partial y} = -\dfrac{7y^2 - 10y - 5}{3y^2(y+1)}$,

which is negative for $y > 9/4$, and so the minimum point is at $h'(3,3)$. Substituting $x = y = 3$ in the constraint $(x - z)y \leq x + z$, we obtain $z \geq 3/2$. Thus,

$$\begin{aligned} h(x,y,z) &\geq h'(x,y) \\ &\geq h'(3,y) \\ &\geq h'(3,3) \\ &= h\left(3,3,\frac{3}{2}\right) \\ &= \frac{7}{9}. \end{aligned}$$

(4.1e). The inequality we ought to prove is $\frac{3x+y-1}{2x+xy+1} \geq \frac{3}{4}$. For that, consider the function $f(x,y) = 3x + 2y - \frac{3xy}{2}$ restricted to the intervals $1 \leq x \leq \frac{3}{2}$ and $1 \leq y \leq 4$. If we fix y, then the function $f(x,y)$ is linear in x, and so the x coordinate of its minimum point is either 1 or $\frac{3}{2}$. We have $f(1,y) = 3 + \frac{y}{2} \geq \frac{7}{2}$ and $f\left(\frac{3}{2},y\right) = \frac{9}{2} - \frac{y}{4} \geq \frac{7}{2}$, entailing $f(x,y) \geq \frac{7}{2}$. Then

$$\begin{aligned} \frac{f(x,y)}{2} &\geq \frac{7}{4} \\ \frac{3x}{2} + y - \frac{3xy}{4} &\geq \frac{7}{4}, \end{aligned}$$

implying $\dfrac{3x}{2} + y \geq \dfrac{3xy}{4} + \dfrac{7}{4}$. From this, we can write

$$\begin{aligned} 3x + y - 1 &= \frac{3}{2}x + \left(\frac{3}{2}x + y\right) - 1 \\ &\geq \frac{3}{2}x + \frac{3}{4}xy + \frac{7}{4} - 1 \\ &\geq \frac{3}{2}x + \frac{3}{4}xy + \frac{3}{4} \\ &= \frac{3}{4}(2x + xy + 1), \end{aligned}$$

implying $\frac{3x+y-1}{2x+xy+1} \geq \frac{3}{4}$. □

The next lemma shows a lower bound on the size of an in-line packing of a list of circles. This lower bound is based on the lengths of the support intervals of the circles. More precisely, it is the sum of the length of each support interval. We remind the reader that, for the analysis of the approximation factor, every circle has radius at least 1, since we scaled the radii such that $r_n = 1$.

Lemma 4.2. *Given a list of circles L, the total sum of the lengths of the support intervals of all circles is a lower bound on the size of an optimal in-line packing of L.*

Proof. We show that, in any feasible packing, the open support intervals of all the circles are disjoint. Let $\mathcal{P}$ be a feasible packing of L, and let i and j be two consecutive circles in

$\mathcal{P}$. Without loss of generality, we assume that i is to the left of j. The interval between their base positions is given by $B = [\mathcal{P}(i); \mathcal{P}(j)]$. We want to analyze the sub-intervals of the open support intervals of i and j that are contained in the interval B. So, the sub-intervals we are considering are

$$\left[(\mathcal{P}(i); \mathcal{P}(i) + 2\sqrt{r_i} - 1)\right] \text{ and } \left[(\mathcal{P}(j) - 2\sqrt{r_j} + 1; \mathcal{P}(j))\right].$$

To prove that the open support intervals are disjoint, we compare the sum of the length of these two sub-intervals and the length of the interval B. Note that each sub-interval corresponds to half of the support interval of its respective circle. Then, the ratio we shall analyze is given by

$$\begin{aligned} \frac{\frac{|\mathcal{P}^{supp}(i)|}{2} + \frac{|\mathcal{P}^{supp}(j)|}{2}}{|B|} &\leq \frac{2(\sqrt{r_i} + \sqrt{r_j} - 1)}{dist(\mathcal{P}(i), \mathcal{P}(j))} \\ &\leq \frac{2(\sqrt{r_i} + \sqrt{r_j} - 1)}{2\sqrt{r_i r_j}} \\ &\leq \frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_j}} - \frac{1}{\sqrt{r_i r_j}} \\ &\leq 1, \end{aligned} \tag{4.2}$$

where the first inequality is valid since $|B| \geq dist(\mathcal{P}(i), \mathcal{P}(j))$ (reaching equality when i and j touch each other), the second inequality follows from Lemma 2.1, and the last inequality comes from Lemma 4.1, inequality 4.1a. □

From now on we classify a circle as *large* if its radius is greater than or equal to four, and as *small* otherwise.

Lemma 4.3. *Given a list of circles L and a packing $\mathcal{P}$ of L produced by the algorithm* Greedy *(Algorithm 2), if two small circles are consecutive in the packing, they touch each other.*

Proof. The proof is by induction on the number m of small circles. If $m = 0$ or $m = 1$, the result follows trivially. So suppose that for any packing produced by the algorithm Greedy with $m - 1$ small circles, for $m \geq 2$, no two small circles are consecutive and non-touching. Let $\mathcal{P}'$ be a packing with $m - 1$ small circles such that there do not exist two consecutive non-touching small circles in $\mathcal{P}'$. We want to show that packing the m-th small circle in $\mathcal{P}'$ does not violate the hypothesis. Let k be such m-th small circle. If there exist circles i and j in $\mathcal{P}'$ such that k fits within their gap, then the algorithm packs k in this gap touching the smaller circle. If both i and j are small, by Lemma 2.2, the radius of a largest circle that fits between them must be smaller than one, implying that the gap between i and j cannot accommodate any circle. Therefore, at least one of the circles i and j is large. Let us say i is large. Then we have two cases. If j is also large, then k will not be consecutive to any other small circle and the result follows. Otherwise, if j is small, then j is certainly smaller than i. Thus, the algorithm packs k touching j and the result follows. Lastly, suppose k does not fit in any gap of $\mathcal{P}'$. By the nature of the algorithm, circle k will be packed touching either the first or the last circle, and the result follows. □

Before we move to further results, we need one more definition concerning the support intervals.

Definition 14. *Given a list L of circles and a packing $\mathcal{P}$, let i and j be two circles in $\mathcal{P}$. We call* support interval density *of i and j in $\mathcal{P}$, denoted by $\mathcal{SI}^{\rho}(\mathcal{P}, i, j)$, the total sum of the lengths of the support intervals and sub-intervals entirely or partially contained in the interval $[\mathcal{P}(i); \mathcal{P}(j)]$, divided by the length of the interval $[\mathcal{P}(i); \mathcal{P}(j)]$, i.e.,*

$$\mathcal{SI}^{\rho}(\mathcal{P}, i, j) = \frac{\sum_{k \in L} \left| \mathcal{P}^{supp}(k) \cap [\mathcal{P}(i); \mathcal{P}(j)] \right|}{\left| [\mathcal{P}(i); \mathcal{P}(j)] \right|}.$$

In respect to the boundaries, the support interval density is defined as follows

$$\mathcal{SI}^{\rho}(\mathcal{P}, \vdash_{\mathcal{P}}, i) = \frac{\sum_{k \in L} \left| \mathcal{P}^{supp}(k) \cap [\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(i)] \right|}{\left| [\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(i)] \right|},$$

for the left boundary, and as

$$\mathcal{SI}^{\rho}(\mathcal{P}, i, \dashv_{\mathcal{P}}) = \frac{\sum_{k \in L} \left| \mathcal{P}^{supp}(k) \cap [\mathcal{P}(i); \mathcal{P}(\dashv_{\mathcal{P}})] \right|}{\left| [\mathcal{P}(i); \mathcal{P}(\dashv_{\mathcal{P}})] \right|},$$

for the right boundary.

Finally, we have formally defined all the concepts we need to prove the following lemma, which says that, given two consecutive circles in an in-line packing, their support interval density is at least 3/4.

Lemma 4.4. *Given a list L with $|L| \geq 2$, and a packing $\mathcal{P}$ of L produced by the algorithm* Greedy*, let i and j be two consecutive circles. Then $\mathcal{SI}^{\rho}(\mathcal{P}, i, j) \geq 3/4$ or there exists a circle k touching i such that $\mathcal{SI}^{\rho}(\mathcal{P}, i, k) \geq 3/4$ with j packed between i and k.*

Proof. The proof is divided in two cases.

Case 1. Circles i and j touch each other. Then, by definition of the support interval density, we have

$$\begin{aligned} \mathcal{SI}^{\rho}(\mathcal{P}, i, j) &= \frac{2\sqrt{r_i} - 1 + 2\sqrt{r_j} - 1}{dist(\mathcal{P}(i), \mathcal{P}(j))} \\ &= \frac{2(\sqrt{r_i} + \sqrt{r_j} - 1)}{2\sqrt{r_i r_j}} \\ &= \frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_j}} - \frac{1}{\sqrt{r_i r_j}}. \end{aligned} \tag{4.3}$$

In order to reach the desired result, we will apply one of the inequalities of Lemma 4.1, so we must show that some constraints are respected. First, we assure that $0 < \frac{1}{\sqrt{r_i}}, \frac{1}{\sqrt{r_j}} \leq 1$, which is true since every circle has radius at least one. Now, consider the radius of a largest

circle g that would fit between i and j. By Lemma 2.2, it follows that $\frac{1}{\sqrt{r_g}} = \frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_j}}$. Since packing the last circle, which has radius one, caused some increase in the span, we know it was not packed in a gap. As a consequence, $r_g < 1$, implying $\frac{1}{\sqrt{r_g}} > 1$, which in turn implies that $\frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_j}} > 1$. That being the case, we can then apply inequality (4.1b), from Lemma 4.1, obtaining

$$\frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_j}} - \frac{1}{\sqrt{r_i r_j}} \geq \frac{3}{4} \tag{4.4}$$

and so, together with equation (4.3), the result follows.

Case 2. Circles i and j do not touch. Then, by Lemma 4.3, at least one of them, say i, is large; otherwise, i and j would touch each other. To assist the calculation of the length of the interval $\big[\mathcal{P}(i); \mathcal{P}(j)\big]$, consider again a largest circle g that fits between i and j. Then g touches both i and j, hence Lemma 2.1 gives us the following.

$$\begin{aligned} dist(\mathcal{P}(i), \mathcal{P}(j)) &= dist(\mathcal{P}(i), \mathcal{P}(g)) + dist(\mathcal{P}(g), \mathcal{P}(j)) \\ &= 2\sqrt{r_i}\sqrt{r_g} + 2\sqrt{r_g}\sqrt{r_j} \\ &= 2\sqrt{r_g}(\sqrt{r_i} + \sqrt{r_j}) \\ &\leq 2(\sqrt{r_i} + \sqrt{r_j}), \end{aligned} \tag{4.5}$$

where the last inequality comes from the fact that the last circle was not packed between i and j, therefore $r_g < 1$, entailing $\sqrt{r_g} < 1$. Now we analyze the support interval density between i and j.

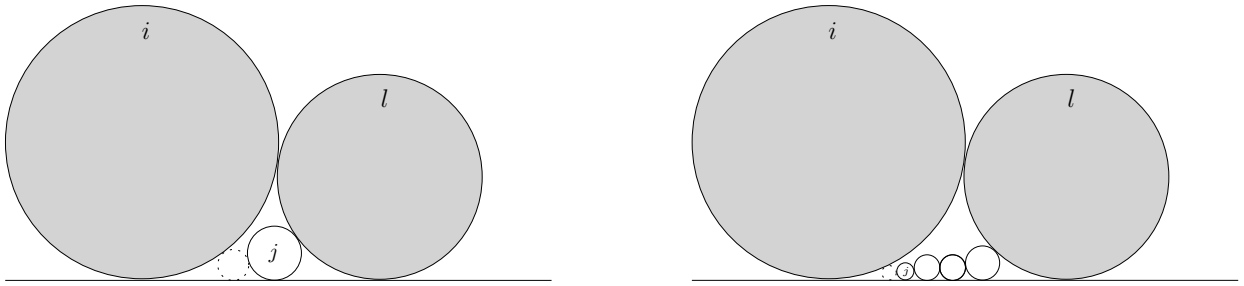
$$\begin{aligned} \mathcal{SI}^{\rho}(\mathcal{P}, i, j) &= \frac{2\sqrt{r_i} - 1 + 2\sqrt{r_j} - 1}{dist(\mathcal{P}(i), \mathcal{P}(j))} \\ &\geq \frac{2(\sqrt{r_i} + \sqrt{r_j} - 1)}{2(\sqrt{r_i} + \sqrt{r_j})} \\ &= 1 - \frac{1}{\sqrt{r_i} + \sqrt{r_j}}. \end{aligned} \tag{4.6}$$

Since i is large, we know that $r_i \geq 4$. Thus, if $r_i \geq 9$ or $r_j \geq 4$, the inequality

$$1 - \frac{1}{\sqrt{r_i} + \sqrt{r_j}} \geq \frac{3}{4}$$

holds and the result follows.

It remains to analyze the case where $4 \leq r_i < 9$ and $1 \leq r_j < 4$. By hypothesis, the circles i and j do not touch. Then, there must exist a third circle to the right of i, touching i from the right. We call such circle l. By the nature of the algorithm, the circle j was packed between the circles i and l. Since j does not touch i, it must be touching another circle that is to its right. If j is the only circle in the gap between i and l, then j necessarily touches l. If, on the other hand, there are two or more circles in the gap between i and l, then j is the



(a) case when j is the only circle between i and l.

(b) case when there are two or more circles between i and l.

Figure 4.1: Sketch of the possible scenarios when the circles i and j do not touch. In both cases, the circle in dotted line is a largest circle that fits between i and j and does not exist in the packing.

leftmost of them (because i and j are consecutive). See Figure 4.1 for an illustration.

We will now analyze the support interval density of the interval $\big[\mathcal{P}(i);\mathcal{P}(l)\big]$. Since we have two different possibilities, depending on the number of circles packed between i and l, we divide the remaining of the proof in two sub-cases.

Case 2.1. Circle j is the only circle packed in the gap between i and l, as illustrated in Figure 4.1a. Since i and j do not touch, j must be touching l. Therefore, $r_l \leq r_i$. Then we shall consider half of the support interval of i, half of the support interval of l and the entire support interval of j, obtaining $2\sqrt{r_i} + 4\sqrt{r_j} + 2\sqrt{r_l} - 4$. As for the distance, we have $dist(\mathcal{P}(i), \mathcal{P}(l)) = 2\sqrt{r_i}\sqrt{r_l}$. Again, we want to use one of the inequalities from Lemma 4.1, more precisely the inequality 4.1d, and for that we shall show that its two necessary constraints are respected. The first constraint imposes that $1 \leq \sqrt{r_i}, \sqrt{r_j}, \sqrt{r_l} \leq 3$, and it follows directly from the fact that $r_j \leq r_l \leq r_i < 9$. The second constraint imposes the following:

$$\sqrt{r_l}(\sqrt{r_i} - \sqrt{r_j}) < \sqrt{r_i} + \sqrt{r_j}. \tag{4.7}$$

To verify this, consider a largest circle g that fits between i and l. According to Lemma 2.1, it holds that

$$2\sqrt{r_i}\sqrt{r_l} = 2\sqrt{r_i}\sqrt{r_g} + 2\sqrt{r_g}\sqrt{r_j} + 2\sqrt{r_j}\sqrt{r_l},$$

which gives us

$$\sqrt{r_g} = \frac{\sqrt{r_l}(\sqrt{r_i} - \sqrt{r_j})}{\sqrt{r_i} + \sqrt{r_j}}.$$

Since the last circle was not placed between i and j, we know that $r_g < 1$, entailing $\sqrt{r_g} < 1$. Therefore, we infer the inequality 4.7 and so we can apply inequality (4.1d). At last, we

calculate the support interval density.

$$\begin{aligned}\mathcal{SI}^{\rho}(\mathcal{P}, i, l) &= \frac{2\sqrt{r_i} + 4\sqrt{r_j} + 2\sqrt{r_l} - 4}{2\sqrt{r_i}\sqrt{r_l}} \\ &= \frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_l}} + \frac{2(\sqrt{r_j} - 1)}{\sqrt{r_i}\sqrt{r_l}} \\ &\geq \frac{7}{9} \qquad (4.8) \\ &> \frac{3}{4},\end{aligned}$$

where inequality (4.8) comes from Lemma 4.1, inequality (4.1d).

Case 2.2. There are two or more circles between i and l, as illustrated in Figure 4.1b. Here we have $m \geq 2$ circles between i and l, and as a consequence of i and j being consecutive, we know that j is the leftmost circle among the m circles packed between i and l. In addition, because each circle has radius at least one, the total sum of the lengths of the support intervals that must be considered is at least

$$2\sqrt{r_i} - 1 + 2\sqrt{r_l} - 1 + m(4\sqrt{1} - 2) \geq 2\sqrt{r_i} + 2\sqrt{r_l} + 2. \qquad (4.9)$$

Again, the distance between $\mathcal{P}(i)$ and $\mathcal{P}(l)$ is given by $2\sqrt{r_i}\sqrt{r_l}$. Prior to calculating the support interval density, we must assure that some constraints about r_i and r_l are respected in order to apply Lemma 4.1, inequality 4.1c. More precisely, we need to assure that $\frac{1}{3} \leq \frac{1}{\sqrt{r_i}}, \frac{1}{\sqrt{r_l}} \leq 1$. This follows directly from the fact that $1 \leq r_l \leq r_i < 9$. Finally, the support interval density is given by

$$\begin{aligned}\mathcal{SI}^{\rho}(\mathcal{P}, i, l) &\geq \frac{2\sqrt{r_i} + 2\sqrt{r_l} + 2}{2\sqrt{r_i}\sqrt{r_l}} \\ &= \frac{1}{\sqrt{r_i}} + \frac{1}{\sqrt{r_l}} + \frac{1}{\sqrt{r_i}\sqrt{r_l}} \\ &\geq \frac{7}{9} \qquad (4.10) \\ &> \frac{3}{4},\end{aligned}$$

where inequality (4.10) follows from Lemma 4.1, inequality (4.1c).

Finally, the cases showed cover all possible cases and the result follows. □

The result we have just presented handles only the case of consecutive circles. This means we still need to regard the case of the boundaries. The next lemma shows that, for each of the two boundaries, the support interval density of the circle touching the boundary and the respective boundary is at least 3/4.

Lemma 4.5. *Given a list L with $|L| \geq 2$, and a packing $\mathcal{P}$ of L produced by the algorithm*

Figure 4.2: Sketch of the configuration of the circles near the left and right boundaries.

Greedy, *there exist circles s and t such that the following inequality holds.*

$$\mathcal{SI}^{\rho}(\mathcal{P}, \vdash_{\mathcal{P}}, s) + \mathcal{SI}^{\rho}(\mathcal{P}, t, \dashv_{\mathcal{P}}) \geq \frac{3}{4}.$$

Proof. Without loss of generality, we assume that the last circle packed, which has radius one, increased the span, hence it must be touching either the left or the right boundary of $\mathcal{P}$. Suppose it touches the right boundary.

Let i and j be the two leftmost circles in $\mathcal{P}$, i.e., the first and second circles, in order of base positions from left to right, and let y and z be the two rightmost circles in $\mathcal{P}$, with $\mathcal{P}(y) < \mathcal{P}(z)$, i.e., circle y appears before circle z, considering the order of base positions. Figure 4.2 shows a sketch of such configuration. From the assumption that packing the last circle increased the span and that it touches the right boundary, we infer that circle z is the last circle and so $r_z = 1$. In addition, by the nature of the algorithm, we have $r_i \leq r_y$.

Now we need to represent the positions of the two boundaries as a function of the circles mentioned so far. It is straightforward that the position of the right boundary is given by the base position of z plus the radius of z, i.e., $\mathcal{P}(\dashv_{\mathcal{P}}) = \mathcal{P}(y) + r_y$. Similarly, the position of the left boundary is given by $\mathcal{P}(\vdash_{\mathcal{P}}) = \mathcal{P}(i) - r_i$. Now, let g be a largest circle that would fit between the circle i and the left boundary. Then the position of the left boundary can also be written as $\mathcal{P}(\vdash_{\mathcal{P}}) = \mathcal{P}(g) - r_g$. This implies that $|\left[\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(i)\right]| = |\left[\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(g)\right]| + |\left[\mathcal{P}(g); \mathcal{P}(i)\right]|$. From this, we infer that $r_i = r_g + \sqrt{r_g r_i}$. Because the last circle was not packed in this space, we know that $r_g < 1$, implying that $r_i < 1 + \sqrt{r_i}$. The proof is divided in two cases, depending on the radius of circle i.

Case 1. $r_i \geq 9/4$. In this case, we do not need to use circle j to help bound the density, and so we shall proceed with the analysis of the support interval density of the two following intervals together:

$$\left[\mathcal{P}(i) - r_i; \mathcal{P}(i)\right] \text{ and } \left[\mathcal{P}(y); \mathcal{P}(z) + 1\right].$$

Using Lemma 2.1 to calculate the distance between the base positions of two consecutive and touching circles, we obtain that the total length of the intervals being analyzed is given by

$$\begin{aligned} |\left[\mathcal{P}(i) - r_i; \mathcal{P}(i)\right]| + |\left[\mathcal{P}(y); \mathcal{P}(z) + 1\right]| &= r_i + 2\sqrt{r_y} + 1 \\ &< 2\sqrt{r_i} + 2\sqrt{r_y} + 2. \end{aligned}$$

In order to calculate the support interval density of the intervals being analyzed, we still need to know the total sum of the support intervals falling within them. Recall that g is an imaginary circle, so we must consider only the support intervals of i, y and z. Then, the sum we are looking for is

$$2\sqrt{r_i} - 1 + 2\sqrt{r_y} - 1 + 4\sqrt{r_z} - 2 = 2\sqrt{r_i} + 2\sqrt{r_y}.$$

Finally, we can calculate the support interval density of the two intervals, which is shown below.

$$\begin{aligned}
\mathcal{SI}^{\rho}(\mathcal{P}, \vdash_{\mathcal{P}}, i) + \mathcal{SI}^{\rho}(\mathcal{P}, y, \dashv_{\mathcal{P}}) &> \frac{2\sqrt{r_i} + 2\sqrt{r_y}}{2\sqrt{r_i} + 2\sqrt{r_y} + 2} \\
&= 1 - \frac{1}{\sqrt{r_i} + \sqrt{r_y} + 1} \\
&\geq 1 - \frac{1}{2\sqrt{\frac{9}{4}} + 1} \qquad (4.11) \\
&= 1 - \frac{1}{4} \\
&= \frac{3}{4},
\end{aligned}$$

where inequality (4.11) holds because, by hypothesis, $r_i \geq \frac{9}{4}$, and $r_y \geq r_i$.

Case 2. $r_i < 9/4$. Thus, i is small. Here we will need circle j, and so we must know how it stands in relation to i. If $r_j < r_i$, then j is also small, and by Lemma 4.3 it follows that i and j touch each other. If, however, $r_j \geq r_i$, then j was packed before i. Since i is the leftmost circle, at the moment it was being packed, it did not fit in any gap, so it was packed in one of the extremities (in this case in the left one). Therefore, by the nature of the algorithm, i touches j. From that we can conclude that i and j certainly touch each other.

In this case, since we need circle j to bound the density, we shall analyze the following intervals:

$$\big[\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(j)\big] \text{ and } \big[\mathcal{P}(y); \mathcal{P}(\dashv_{\mathcal{P}})\big].$$

While the length of the second interval was already calculated, for the first we must consider a largest circle g that fits between i and the left boundary. Using Lemma 2.1 to calculate the distance between the base positions of two consecutive touching circles, we obtain

$$r_g + 2\sqrt{r_g}\sqrt{r_i} + 2\sqrt{r_i}\sqrt{r_j} + 2\sqrt{r_y}\sqrt{r_z} + r_z \quad < \quad 2\sqrt{r_i} + 2\sqrt{r_y} + 2\sqrt{r_i}\sqrt{r_j} + 2, \qquad (4.12)$$

since $r_g < 1$. Before we proceed to the calculation of the support intervals, we shall show how the square-root of the radius of circle j is bounded, so that we can apply the inequality (4.1e), from Lemma 4.1. For that, consider the length of the interval between the left boundary

and the base position of j. Since $r_g < 1$ and $r_i < 9/4$, we obtain

$$\begin{aligned}\left|\left[\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(j)\right]\right| &= r_g + 2\sqrt{r_g}\sqrt{r_i} + 2\sqrt{r_i}\sqrt{r_j} \\ &< 2\sqrt{r_i} + 2\sqrt{r_i}\sqrt{r_j} + 1 \\ &< 2\sqrt{\frac{9}{4}} + 2\sqrt{\frac{9}{4}}\sqrt{r_j} + 1 \\ &= 3\sqrt{r_j} + 4. \end{aligned} \tag{4.13}$$

We already know that the left boundary is at position $\mathcal{P}(g) - r_g$. Obviously, the leftmost point of j is to the right of the left boundary, so $r_j \leq |\left[\mathcal{P}(\vdash_{\mathcal{P}}); \mathcal{P}(j)\right]|$, which in turn gives us $r_j < 3\sqrt{r_j} + 4$. Solving this inequality, we have

$$1 \leq \sqrt{r_j} \leq 4. \tag{4.14}$$

Now we shall look at the support intervals. In this case, we consider the circles i, j, y and z, obtaining

$$4\sqrt{r_i} - 2 + 2\sqrt{r_j} - 1 + 2\sqrt{r_y} - 1 + 4\sqrt{r_z} - 2 \quad = \quad 4\sqrt{r_i} + 2\sqrt{r_j} + 2\sqrt{r_y} - 2.$$

Finally, the support interval density of the intervals being analyzed is given by

$$\begin{aligned}\mathcal{SI}^{\rho}(\mathcal{P}, \vdash_{\mathcal{P}}, j) + \mathcal{SI}^{\rho}(\mathcal{P}, y, \dashv_{\mathcal{P}}) &= \frac{4\sqrt{r_i} + 2\sqrt{r_j} + 2\sqrt{r_y} - 2}{2\sqrt{r_i} + 2\sqrt{r_y} + 2\sqrt{r_i}\sqrt{r_j} + 2} \\ &= \frac{2\sqrt{r_i} + \sqrt{r_j} + \sqrt{r_y} - 1}{\sqrt{r_i} + \sqrt{r_y} + \sqrt{r_i}\sqrt{r_j} + 1} \\ &\geq \frac{3\sqrt{r_i} + \sqrt{r_j} - 1}{2\sqrt{r_i} + \sqrt{r_i}\sqrt{r_j} + 1} \qquad (4.15) \\ &\geq \frac{3}{4}, \qquad (4.16)\end{aligned}$$

where the inequality (4.15) holds because $r_y \geq r_i$ and the inequality (4.16) comes from Lemma 4.1, inequality (4.1e), which in turn can be applied because we have the bounds in (4.14).

Finally, it is important to observe that, to prove the second case, we used the circle j, adding half of the support interval of j to the total sum of lengths of the support intervals considered. If, however, this support interval was also needed to help bound the density between two consecutive circles forming an interval of the form $\left[\mathcal{P}(i); \mathcal{P}(l)\right]$ where l is larger than j, l is touching i, and j was packed in the gap between i and l, then half of the support interval of j would be double counted. According to Lemma 4.4 this happens only when $r_l < 9$. Though, if $r_i < 9/4$ and $r_l < 9$, then no circle of size at least one would fit between i and l. This guarantees that the problematic situation actually never happens. □

In the following, we show the main result of this section, which is the approximation factor of algorithm Greedy.

Theorem 4.6. *Algorithm Greedy is a* 4/3*-approximation for the ICP problem.*

Proof. Let L be a list of circles and $\mathcal{P}^\star$ be an optimal packing of L. Let $\mathcal{SI}(L)$ be the sum of the lengths of the support intervals of all circles of L. By Lemma 4.2, we have

$$\mathcal{SI}(L) \leq |\mathcal{P}^\star| = \mathrm{OPT}(L).$$

Let $\mathcal{P}$ be a packing of L produced by algorithm Greedy, and let $P = (i_1, \ldots, i_n)$ be the sequence of the circles in $\mathcal{P}$ in order of base positions, from left to right. We want to show that $\mathcal{SI}(L)$ is at least 3/4 of the size of $\mathcal{P}$, i.e.,

$$\mathcal{SI}(L) \geq \frac{3}{4}|\mathcal{P}|,$$

which in turn implies

$$\begin{aligned} |\mathcal{P}| &\leq \frac{4}{3}\mathcal{SI}(L) \\ &\leq \frac{4}{3}|\mathcal{P}^\star| = \frac{4}{3}\mathrm{OPT}(L). \end{aligned}$$

If $|L|$ equals to one, there is only one packing $\mathcal{P}$ of L, its size is given by the diameter of the only circle forming L, and the result follows trivially. So suppose $|L| \geq 2$. From lemmas 4.4 and 4.5, we have

$$\mathrm{OPT}(L) \geq SI(L) \geq \frac{3}{4}|P|,$$

implying

$$|P| \leq \frac{4}{3}\mathrm{OPT}(L).$$

□

4.2 A QPTAS

In their work, Dürr et al. [7] presented a QPTAS for the in-line packing of isosceles right triangles. As Helmut et al. [1] suggested, such QPTAS can be adapted to a QPTAS for the case when the items are circles. In this section, we present an implementation of the suggested adaption. To do so we must first introduce, in the subsection 4.2.1, a restricted version of the ICP problem, as well as an algorithm which computes an optimal solution for this restricted version. Next, in the subsection 4.2.2, we present the QPTAS for the general version of the ICP problem. For the remaining of this section, recall that we denote the diameter of a circle i as d_i.

4.2.1 A Restricted Version of the ICP Problem

Consider a restricted version of the ICP problem where the circles have a logarithmic (on the number of circles in the input list) number of different diameters and the base positions are limited to a finite set of points, whose size is polynomial in the number of circles in the input list. We can take advantage of these two restrictions to design a dynamic programming algorithm that computes an optimal solution for this restricted version of the problem, in *quasi*-polinomial-time. As it happens in the general version of the ICP problem, an instance of this restricted version consists of a list of circles.

Let L be a list of circles. Then, we enumerate the different diameters of the circles in L by a sequence $(f_1, \ldots, f_k)$, and we say a circle is *indexed* by j if it has diameter f_j. We say the *demand* of each diameter f_j is the number of circles indexed by j. We call *configuration* a tuple $Q = (i_1, \ldots, i_k, p_1, \ldots, p_k)$ where i_t is the demand of diameter f_t and p_t is the base position of the last packed circle of diameter f_t. When we have $i_t = 0$, we set $p_t = 0$, by convention. We say a configuration $A = (a_1, \ldots, a_k, q_1, \ldots, q_k)$ is *smaller* than a configuration $B = (b_1, \ldots, b_k, p_1, \ldots, p_k)$ if $a_i \leq b_i$ for every $1 \leq i \leq k$ and there exists a j such that $a_j < b_j$. Moreover, we say a configuration is *feasible* (*unfeasible*) if it is possible (not possible) to build a packing with i_j circles of diameter f_j such that the rightmost one is packed at position p_j, for $1 \leq j \leq k$. The size of a feasible configuration is denoted $size(Q) = \max_{j \in \{1,\ldots,k\}}(p_j + f_j/2)$. If Q is not feasible, by convention we say $size(Q) = 0$. We say Q is *empty* if all demands are zero. Given a demand i_j, if $i_j = 0$ then, by convention, $p_j = 0$.

As mentioned previously, there is an algorithm which computes an optimal solution for the restricted version of ICP. Such algorithm is a dynamic programming and it builds a table T indexed by configurations. Given a feasible configuration $Q = (i_1, \ldots, i_j, \ldots, i_k, p_1, \ldots, p_j, \ldots, p_k)$, $T(Q)$ stores one of the three following values:

i) $\varnothing$, if Q is an empty configuration,

ii) a base position p'_j, such that the last circle of a packing $\mathcal{P}$ with demands $(i_1, \ldots, i_k)$ is indexed by j, and p'_j is the base position of the rightmost circle, if any, indexed by j that is packed before p_j (i.e., $p'_j < p_j$) in $\mathcal{P}$,

iii) $\otimes$, if there is no such p'_j or Q is unfeasible.

The following definition associates a configuration with the packing represented by such configuration.

Definition 15. *Consider table T and a configuration $Q = (i_1, \ldots, i_k, p_1, \ldots, p_k)$. Let $p'_j = T(Q)$. We denote by $pck(Q)$ the packing built as follows. Find the greatest base position in Q, say it is p_j. Then pack a circle indexed by j at base position p_j and repeat the process for the configuration $(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)$, until $(i_1, \ldots, i_k) = (0, \ldots, 0)$.*

In some contexts, we might want to add a circle indexed by j at base position p_j in a packing $\mathcal{P}$.

Definition 16. *Given a packing feasible $\mathcal{P}$ and a pair (j, p_j), we denote by $\mathcal{P}|(j, p_j)$ the packing obtained by adding to $\mathcal{P}$ a circle indexed by j at position p_j, if such resulting packing is feasible.*

Once the table T is built, the algorithm finds a configuration Q of minimum size and then builds the packing $pck(Q)$. Next, we present a recurrence formula for the construction of such table. The value of $T(Q)$ is $\varnothing$ if all the demands are zero, $\otimes$ if the configuration is unfeasible and, otherwise, the value is the base position p'_j of the rightmost circle of diameter f_j when its demand is $i_j - 1$. The recurrence formula of the algorithm is as follows. See the algorithm RestrictedICP (Algorithm 5) for an implementation.

$$T(i_1, \ldots, i_k, p_1, \ldots, p_k) = \begin{cases} \varnothing & \text{if } (i_1, \ldots, i_k) = (0, \ldots, 0). \\ 0 & \text{if } (i_j - 1 = 0) \text{ and } T(i_1, \ldots, 0, i_k, p_1, \ldots, 0, p_k). \\ p'_j & \text{if } j = \arg\max_t \{p_t : t \in \{1, \ldots, k\}\}, \text{ and} \\ & \quad p'_j = \max_t \{p_t : p_t \in P \text{ and } 0 < p_t < p_j \text{ such that} \\ & \quad T(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k) \neq \otimes, \text{ and} \\ & \quad pck(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)|(j, p_j) \text{ is feasible}\}. \\ \otimes & \text{if there is no such } p'_j \text{ or } (i_1, \ldots, i_k, p_1, \ldots, p_k) \text{ is unfeasible.} \end{cases} \tag{4.17}$$

In order to check if two circles overlap, the algorithm RestrictedICP uses the subroutine called overlaps. In addition, given a configuration Q, the algorithm uses a subroutine called exist_overlap_with_last_circle to check if there exists overlap between the last circle of Q with of the rightmost circles of each distinct diameter. In the following, we present implementations for these procedures.

Function 3: overlaps

Input: A tuple (r_1, r_j, x_i, x_j) where r_i, r_j are the radii of two circles and x_i, x_j are their respective base positions.

Output: True if the circles overlap, False otherwise.

```
1 if r_i r_j < (x_i - x_j)^2 / 4 then
2     return True
3 return False
```

By Lemma 2.1, when two circles i and j touch each other, the distance between their base positions is $2\sqrt{r_i r_j}$. The condition in line 1 tests if the difference between the base positions of the circles is smaller than the distance between two circles that touch each other, and comes from squaring the inequality $2\sqrt{r_i r_j} < |x_i - x_j|$ to avoid calculating square-roots, since they are not rational numbers.

Function 4: exist_overlap_with_last_circle

Input: A configuration $(i_1, \ldots, i_k, p_1, \ldots, p_k)$.

Output: True if there exists overlap between the last circle and the rightmost circles of each distinct diameters in the candidate packing corresponding to the input configuration, False otherwise.

Let $j = \arg\max_t \{p_t : t \in \{1, \ldots, k\}\}$.
for $i = 1$ to k, $i \neq j$ **do**
    if overlaps(f_i, f_j, p_i, p_j) **then**
        return True
return False

In the following, we present the algorithm RestrictedICP.

Algorithm 5: RestrictedICP

Input: A tuple (L, N, P) where L is a list of k circles, N a list of the k demands of the circles in L, and P is a list of positions.

Output: An optimal packing of L restricted to P.

forall $(p_1, \ldots, p_k) \in P^k$ **do**
    $T(0, \ldots, 0, p_1, \ldots, p_k) \leftarrow \varnothing$.
for $i_1 = 0, \ldots, n_1$ **do**
        $\ddots$
    for $i_k = 0, \ldots, n_k$ **do**
        if $(i_1, \ldots, i_k) \neq (0, \ldots, 0)$ **and** $(p_1, \ldots, p_k) < (f_1/2, \ldots, f_k/2)$ **then**
            forall $(p_1, \ldots, p_k) \in P^k$ **do**
                if exist_overlap_with_last_circle$(i_1, \ldots, i_k, p_1, \ldots, p_k)$ **then**
                    $T(i_1, \ldots, i_k, p_1, \ldots, p_k) \leftarrow \otimes$
                else
                    $j = \arg\max_t \{p_t : t \in \{1, \ldots, k\}\}$.
                    if $(i_j - 1) = 0$ **and** $T(i_1, \ldots, 0, \ldots, i_k, p_k, \ldots, 0, \ldots, p_k) \neq \otimes$ **then**
                        $T(i_1, \ldots, i_k, p_1, \ldots, p_k) \leftarrow 0$
                    else
                        $S = \{q \in P : 0 < q < p_j$ and
                        $T(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, q, \ldots, p_k) \neq \otimes$ and
                        overlaps$(f_j, f_j, p_j, q) =$ False$\}$.
                        if $S \neq \emptyset$ **then**
                            $T(i_1, \ldots, i_k, p_i, \ldots, p_k) \leftarrow p'_j$ where $p'_j = \max\{S\}$.
                        else
                            $T(i_1, \ldots, i_k, p_i, \ldots, p_k) \leftarrow \otimes$.
return ConstructPacking(T)

The input of the algorithm RestrictedICP is a list of the different diameters together with their demands, and a finite set of base positions. In a general view, the algorithm checks all combinations of demands, $(i_1, \ldots, i_k)$, and base positions, $(p_1, \ldots, p_k)$, and determines whether or not a configuration leads to a packing. The sequence of loops of lines 3 to 5 guarantees that all combinations of demands are considered, and in increasing order. This means that once a configuration is processed, the result for all the smaller configurations is already computed. In the next steps, the table T is filled. After obtaining a sequence $(i_1, \ldots, i_k)$ of demands, from the loops in lines 3 to 5, the algorithm checks whether or not this sequence corresponds to an empty packing, i.e., if $(i_1, \ldots, i_k) = (0, \ldots, 0)$, then regardless of the base positions, this entry is already filled with the value that indicates an empty packing (from the initialization of the table, in lines 1 and 2). Otherwise, the algorithm proceeds to the loop of line 7, where all possible configurations with demands equal to $(i_1, \ldots, i_k)$ are analyzed.

Let $Q = (i_1, \ldots, i_k, p_1, \ldots, p_k)$ be the current configuration being processed. If there exists in Q overlap between any of the rightmost circles of each different diameter with the last circle, then the algorithm fills the table with $\otimes$, in line 9, indicating that the current configuration, Q, is not feasible, and proceeds to check the next configuration.

If, on the other hand, there are no overlaps among the rightmost circles of each diameter in Q, then the algorithm must check if there exists a feasible smaller configuration for the case where the last circle of the packing associated with Q has not yet been packed. For that, in line 11, the algorithm first obtains the index of the last circle in the packing corresponding to Q, say it is j. Then, the algorithm must check the smaller configurations of the form $Q' = (i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)$, i.e., the demand of circles of diameter f_j is decreased by one. There are two cases. First, if the last circle of Q happens to be the last circle of diameter f_j, then the demand of f_j in the smaller configurations of the form Q' is zero, i.e., $i_j - 1 = 0$. So, if the configuration $(i_1, \ldots, 0, \ldots, i_k, p_1, \ldots, 0, \ldots, p_k)$ is feasible, then Q can be obtained from Q' and, in line 13, the table is filled with the base position 0. Otherwise, if $i_j - 1 \geq 1$, then the algorithm searches for the set S of the feasible configurations of the form Q', such that p'_j is smaller than p_j, and also such that adding the pair (j, p_j) to the packing $pck(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)$ does not cause any overlaps. Finally, if S is not empty, it means that, for at least one p'_j, one can obtain a feasible packing of the current configuration, Q. The algorithm chooses the greatest p'_j, in line 17. Otherwise, if S is empty, then, in line 19, the algorithm fills the table with $\otimes$, since Q cannot be built from any p'_j.

Lemma 4.7. *Given a configuration $A = (a_1, \ldots, a_k, p_1, \ldots, p_k)$, if A is feasible, then there exists a pair (j, p'_j) such that*

i) $A' = (a_1, \ldots, a_j - 1, \ldots, a_k, p_1, \ldots, p'_j, \ldots, p_k)$ *is feasible,*

ii) $p_j = \max_t\{p_t : t \in 1, \ldots, k\}$, *and*

iii) $pck(A) = pck(A')|(j, p_j)$.

The sequence of nested for's guarantees that the configurations are processed in increasing order, so when the current configuration is being processed, all smaller ones were already processed.

Lemma 4.8. *Algorithm* RestrictedICP *(Algorithm 5) correctly builds the table T.*

Proof. For each configuration $Q = (i_1, \ldots, i_k, p_1, \ldots, p_k)$, the value $T(Q)$ must be one of the following:

i) $\varnothing$, if Q represents an empty packing,

ii) p'_j, if Q is feasible and can be obtained by adding a circle of diameter f_j at position p_j to a packing corresponding by the smaller configuration $Q' = (i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)$,

iii) $\otimes$, if there is no p'_j or Q is unfeasible.

Observe that the loops in lines 3 to 5 guarantee that when one configuration is considered all the smaller configurations are already processed. So, we prove the lemma by induction.

The base case is when all demands are 0. Then we have an empty packing, and the algorithm correctly assigns the symbol $\emptyset$ to $T(Q)$, in lines 1 and 2.

Now consider a configuration Q and suppose that all configurations smaller than Q are already correctly computed. We have two cases. If there is overlap between the last circle of Q and one of the rightmost circles of the other diameters, then Q is unfeasible. In this case, the algorithm correctly assigns $\otimes$ to $T(Q)$, in line 9.

Otherwise, if there is not overlap between the last circle of Q and one of the rightmost circles of the other diameters, then there might exist a smaller feasible configuration $Q' = (i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)$ from which a feasible packing corresponding to Q can be obtained. To verify that, the algorithm checks all possible smaller configurations. If the demand of f_j in Q'_j is zero, then it suffices to know if the configuration $(i_1, \ldots, 0, \ldots, i_k, p_1, \ldots, 0, \ldots, p_k)$ is feasible. If that is the case, then the p'_j is 0, and the algorithm correctly assigns 0 to $T(i_1, \ldots, 0, \ldots, i_k, p_1, \ldots, 0, \ldots, p_k)$, in line 13. Otherwise, if $i_j - 1 \geq 1$, then there must exist at least one feasible configuration of the form $(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, q, \ldots, p_k)$ where a circle of diameter f_j is packed at some base position q, $0 < q < p_j$. Let C be the set of such configurations. If, for any configuration $(i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, q, \ldots, p_k)$ in C, it is possible to pack a circle of diameter f_j at base position p_j without causing overlap, then one can obtain the current configuration, Q, from such configuration in C. In this case, the base position q is added to the set S, in line 15. If S is not empty, then the table is filled with the greatest base position in S, in line 17. Otherwise, the configuration Q cannot be obtained from a smaller configuration, and the algorithm correctly fills the table with $\otimes$, indicating that Q is unfeasible. □

Once the table is fully filled, it is possible to construct a feasible packing of minimum size. The algorithm ConstructPacking does so. First, the algorithm finds one configuration Q that leads to a feasible packing of minimum size, according to the table produced by the algorithm RestrictedICP. Then, the packing is constructed in a recursive way. The last

circle of the packing represented by the configuration Q is indexed by $j = \arg\max\{p_t : t \in \{1, \ldots, k\}\}$. Therefore, a circle of diameter f_j is packed at base position p_j. Let $T(Q) = p'_j$. So Q was build starting from the configuration $Q' = (i_1, \ldots, i_j - 1, \ldots, i_k, p_1, \ldots, p'_j, \ldots, p_k)$. The algorithm then repeats the process for Q' until it reaches a configuration of the form $(0, \ldots, 0, p_1, \ldots, p_k)$. In Algorithm 6 we present an implementation of this construction process.

Algorithm 6: ConstructPacking

Input: A list of circles L and a table T produced by algorithm RestrictedICP.

Output: An optimal packing $\mathcal{P}$ of L.

1 Let $(p_1, \ldots, p_k) = \arg\min_{(q_1,\ldots,q_k) \in P_k : T((n_1,\ldots,n_k,q_1,\ldots,q_k)) \neq 0} \left\{\max_{j \in \{1,\ldots,k\}}(q_j + f_j/2)\right\}$.
2 $\mathcal{P} \leftarrow \emptyset$.
3 $(i_1, \ldots, i_k) \leftarrow (n_1, \ldots, n_k)$.
4 **while** $(i_1, \ldots, i_k) \neq (0, \ldots, 0)$ **do**
5 $\quad j \leftarrow \arg\max_{t \in \{1,\ldots,k\}} \{p_t\}$.
6 $\quad$ Pack a circle of diameter f_j at base position p_j in $\mathcal{P}$.
7 $\quad p'_j \leftarrow T(i_1, \ldots, i_k, p_1, \ldots, p_k)$.
8 $\quad i_j \leftarrow i_j - 1$.
9 $\quad p_j \leftarrow p'_j$.
10 **return** $\mathcal{P}$

Lemma 4.9. *The algorithm* ConstructPacking *(Algorithm 6) builds an optimal packing.*

Proof. Let $(f_1, \ldots, f_k)$ be the tuple of the different diameters in L and let $(n_1, \ldots, n_k)$ be the corresponding demands. By Lemma 5, the table T is correctly built. Then the list $(p_1, \ldots, p_k)$, obtained in line 1, represents the base positions of the rightmost circles of diameters $(f_1, \ldots, f_k)$ such that $pck(n_1, \ldots, n_k, p_1, \ldots, p_k)$ has minimum size. Therefore, the packing $\mathcal{P}$, constructed in lines 4 to 9, is an optimal packing.

In the first iteration of the loop from lines 4 to 9, the algorithm finds the base position and the diameter of the last circle of the packing. After packing such circle, in line 6, the algorithm consider a smaller packing, where the current last circle is not included, and updates the configuration corresponding to such smaller packing.

The construction is made from right to left, i.e., from the rightmost to the leftmost circle. We show that, in the beginning of each iteration, the packing $\mathcal{P}$ has $(n_1 + \ldots + n_k) - (i_1 + \ldots + i_k)$ circles. Let m be the number of circles in $\mathcal{P}$, i.e., $m = (n_1 + \ldots + n_k) - (i_1 + \ldots, i_k)$. The loop starts with $m = 0$. After the execution of the first iteration, the packing $\mathcal{P}$ has only the last circle, and so $m = 1$. Then, the configuration to be processed next is updated so that it has one less circle. Thus, at the beginning of the next iteration, we have that $m = m + 1$. Lastly, at the end of the last iteration, we have $i_1 + \ldots + i_k = 0$, and therefore $m = (n_1 + \ldots + n_k) - (i_1 + \ldots + i_k) = n_1 + \ldots + n_k$. □

So far we have shown that both the construction of the table T and the rebuilding of the minimum packing are correct. It basically remains to analyze the running time.

Lemma 4.10. *There exists an algorithm that computes, in quasi-polinomial time, an optimum packing for the ICP problem restricted to the case where the number of different diameters is logarithmic on the number of circles and the base positions are limited to a finite set of points, whose size is polynomial on the number of circles.*

Proof. By lemmas 4.8 and 4.9, the algorithm RestrictedICP correctly produces an optimal packing. It remains to argue about the time complexity. Let n be the number of circles and let k be the number of different diameters. Each configuration is a vector in the space $(0,\ldots,n)^k \times P^k$. The loop from line 1 runs over all possible vectors in the space P^k, so it contributes with $|P|^k$. The loops from lines 3 to 5 contributes with $(n+1)^k$. The loop from line 7 contributes with $|P|^k$ and, inside it, lines 8 and 11 contribute each with k, while line 12 contributes with $|P|$.

In our context, it suffices to consider the values of $|P|$ and k given by the discretization proposed in Algorithm 7, presented ahead in the next subsection. Therefore, we have

$$|P|^k + (n+1)^k|P^k|(k+|P|) \leq \lceil n^2/\varepsilon \rceil^{\lceil \log(n/\varepsilon)\rceil} + \\ (n+1)^{\lceil \log(n/\varepsilon)\rceil}\lceil n^2/\varepsilon \rceil^{\lceil \log(n/\varepsilon)\rceil}(\lceil \log(n/\varepsilon)\rceil + \lceil n^2/\varepsilon \rceil),$$

implying *quasi*-polinomial-time complexity. □

4.2.2 The General Version of the ICP Problem

Recall that we are trying to build a QPTAS for the general case of the ICP problem. Let L be a list of circles. We show that Algorithm 7 computes, in *quasi*-polinomial-time, a packing $\mathcal{P}$ of L, whose size is $(1+\varepsilon)\mathrm{OPT}(L)$.

Algorithm 7: A_ε: a QPTAS for the ICP problem.

Input: A tuple (L, r) where L is a list of n circles and r is a function of the radii.
Output: An in-line packing of L.

1 Small $\leftarrow \{i \in L : d_i < \frac{\varepsilon d_{\max}}{n}\}$, where $d_{\max} = \max\{d_i : i \in L\}$.
2 Large $\leftarrow L \setminus$ Small.
3 Sort Large $= (1,\ldots,n')$ so that $d_i \geq d_{i+1}$ for $i = 1,\ldots,n'-1$.
4 Scale all diameters in L so that $d_{n'} = \min\{d_i : i \in \text{Large}\} = 1$.
5 Round the diameters of the circles in Large up to the nearest power of $1+\varepsilon$.
6 Let $P = (K, 2K, \ldots, \lceil \frac{n^2}{\varepsilon} \rceil K)$, for $K = \frac{\varepsilon d_{\max}}{n}$.
7 $\mathcal{P}_{\text{Large}} \leftarrow$ RestrictedICP(Large, P) (Algorithm 5).
8 $\mathcal{P}_{\text{Small}} \leftarrow$ Greedy(Small) (Algorithm 2).
9 **return** $\mathcal{P}_{\text{Large}} \| \mathcal{P}_{\text{Small}}$.

The algorithm partitions the list L into large, list Large, and small, list Small, circles. Then, in line 7, it obtains an optimal packing of the large circles, and, in line 8, it obtains an approximated packing of the small circles. Finally, in line 9, it obtains a packing of the input list L.

Because of the rounding of the diameters of the large circles, in line 5, and the discretization of the base positions, in line 6, the pair $(Large, P)$ characterizes an instance of the restricted version of the ICP problem introduced previously. Then, one can use the algorithms RestrictedICP and ConstructPacking to obtain an optimal packing of the list Large of large circles. We show that the size of an optimal packing for the restricted version of the ICP problem is at most a factor of $1 + \varepsilon$ of the size of an optimal packing for the ICP problem. The first aspect we discuss is the rounding of the diameters of the circles in Large.

Lemma 4.11. *Given a list L of circles and an optimal packing $\mathcal{P}$ of L, rounding the diameters of every circle up to the nearest power of $1 + \varepsilon$ increases the optimal size by at most a factor of $1 + \varepsilon$.*

Proof. Construct a packing $\mathcal{P}'$ from $\mathcal{P}$ by multiplying each diameter and each base position by $1 + \varepsilon$. That only changes the scale of the packing, so $\mathcal{P}'$ is also feasible. Thus, $|\mathcal{P}'| \leq (1 + \varepsilon)|\mathcal{P}|$. Now round each diameter down to the nearest power of $1 + \varepsilon$. Since this does not increase the diameter of the circles, the size of $\mathcal{P}'$ also does not increase, and the result follows. □

Now recall that $d_{n'}$ is the diameter of the smallest circle in the list Large. Then, by construction, $d_{n'} \geq \frac{\varepsilon d_{\max}}{n}$ and, therefore, we have $\frac{d_{\max}}{d_{n'}} \leq \frac{n}{\varepsilon}$. Since all the circles in L where scaled so that $d_{n'} = 1$, we have $d_{\max} \leq \frac{n}{\varepsilon}$. Thus, the maximum number of different diameters is at most $\lceil \log_{1+\varepsilon}(n/\varepsilon) \rceil$. After restricting the available base positions to the set P, we might cause some increase in the size of the packing. We show that such increase is at most a factor of $1 + \varepsilon$.

Lemma 4.12. *Given an optimum packing $\mathcal{P}$ of the large circles, restricting the base positions to the set $P = \{K, 2K, \ldots, \lceil \frac{n^2}{\varepsilon} \rceil K\}$, where $K = \dfrac{\varepsilon d_{\max}}{n}$, increases the optimal size by at most a factor of $1 + \varepsilon$.*

Proof. Without loss of generality, denote by $(1, \ldots, n)$ the sequence of the circles in $\mathcal{P}$, ordered from left to right, and in a way that two circles are consecutive in the packing if and only if they are consecutive in the sequence. Now, consider each circle, starting from the leftmost to the rightmost circle, and repeatedly shift the current circle to the right, pushing the subsequent circles, to the next point in P. Each shift increases the size of $\mathcal{P}$ by at most K. Thus the total increase is at most $nK \leq \varepsilon d_{\max} \leq \varepsilon |\mathcal{P}|$. □

So far, we analyzed the impact of the large circles in the final packing, so it remains to regard the small circles. Establishing the ratio between the smallest and the greatest diameters of the large circles imposes an upper bound on the diameter of the small circles. This is important in order to ensure that the small circles, when concatenated to the packing $\mathcal{P}_{\text{Large}}$, in line 9, do not cause a big increase in the size of the packing. By construction of the list Small, the small circles have diameter smaller than $\frac{\varepsilon d_{\max}}{n}$. As a consequence, even if all the circles in L are small (except the first), they would still not cause a large increase. In such case, we would have an increase of at most $n\frac{\varepsilon d_{\max}}{n}$, which is at most εOPT.

Lemma 4.13. *Given a list of circles L, an optimal packing $\mathcal{P}$ of L and a positive constant ε, let* $(\text{Large}, \text{Small})$ *be a partition of L such that* $\text{Large} = (i \in L : d_i > \frac{\varepsilon d_{max}}{n})$*, where* $d_{max} = \max\{d_1, \ldots, d_n\}$*, and* $\text{Small} = \mathcal{C} \setminus \text{Large}$*. Then it is possible to obtain a packing $\mathcal{P}_{\text{Large}}$ of* Large*, constructed from $\mathcal{P}$ by removing the circles of* Small*, and a packing $\mathcal{P}_{\text{Small}}$ of* Small *such that the packing $\mathcal{P}' = \mathcal{P}_{\text{Large}} \| \mathcal{P}_{\text{Small}}$ satisfies $|\mathcal{P}'| \leq (1+\varepsilon)|\mathcal{P}|$.*

Proof. Let $\mathcal{P}_{\text{Large}}$ be a packing obtained from removing the circles of Small from $\mathcal{P}$. Since $\mathcal{P}_{\text{Large}} \subseteq \mathcal{C}$, we have that $|\mathcal{P}_{\text{Large}}| \leq |\mathcal{P}|$. Let $\mathcal{P}_{\text{Small}}$ be a packing of Small where the circles are packed side by side and as close as possible one to another without causing overlaps. Because each circle in Small has diameter at most $\frac{\varepsilon d_{max}}{n}$ and $|\text{Small}| \leq n$, we have that $|\mathcal{P}_{\text{Small}}| \leq \varepsilon d_{max} \leq \varepsilon|\mathcal{P}|$. Then, when $\mathcal{P}_{\text{Large}}$ and $\mathcal{P}_{\text{Small}}$ are concatenated to obtain $\mathcal{P}'$, we have $|\mathcal{P}'| \leq |\mathcal{P}_{\text{Large}}| + |\mathcal{P}_{\text{Small}}| \leq |\mathcal{P}| + \varepsilon|\mathcal{P}| \leq (1+\varepsilon)|\mathcal{P}|$. □

The main result of this subsection is presented in the following theorem.

Theorem 4.14. *There exist a QPTAS for the In-line Circle Packing problem.*

Proof. We show that Algorithm 7 is a QPTAS for the ICP problem. Let L be a list of circles and $\mathcal{P}^\star$ be an optimal packing of L. Partition L in two lists as follows. List $\text{Small} = \{i \in L : d_i < \frac{\varepsilon d_{\max}}{n}\}$, where $d_{\max} = \max\{d_i : i \in L\}$, and list $\text{Large} = L \setminus \text{Small}$. Now let $\mathcal{P}^\star_{\text{Small}}$ and $\mathcal{P}^\star_{\text{Large}}$ be optimal packings of Small and Large, respectively. By lemmas 4.11 and 4.12, together with Lemma 4.10, we have that $|\mathcal{P}_{\text{Large}}| \leq (1+\varepsilon)|\mathcal{P}^\star_{\text{Large}}|$, and by Theorem 4.6, we have that $|\mathcal{P}_{\text{Small}}| \leq \frac{4}{3}|\mathcal{P}^\star_{\text{Small}}|$. Thus,

$$\begin{aligned}
|\mathcal{P}_{\text{Large}}| + |\mathcal{P}_{\text{Small}}| &\leq (1+\varepsilon)|\mathcal{P}^\star_{\text{Large}}| + \frac{4}{3}|\mathcal{P}^\star_{\text{Small}}| \\
&\leq (1+\varepsilon)|\mathcal{P}^\star| + \frac{4}{3}\varepsilon|\mathcal{P}^\star| \\
&\leq \left(1 + \frac{7}{3}\varepsilon\right)|\mathcal{P}^\star|.
\end{aligned}$$

The time complexity of the algorithm is determined by the time complexity of the algorithm RestrictedICP, which, by Lemma 4.10, implies *quasi*-polinomial-time. □

4.3 In-line Packing of Large Circles

In this section we consider restricted versions of the In-line Circle Packing (ICP) and the In-line Circle Bin Packing (ICB) problems, where the values of the diameters of the circles from the input list are bounded, i.e., the diameters are at least a constant value ε. We refer to the restricted versions of the problems as ICP_ε and ICB_ε, respectively. We start by showing an APTAS for the ICB_ε, and subsequently we show a PTAS for the ICP_ε.

4.3.1 An APTAS for ICB of Large Circles

Here we present an APTAS, using augmented bins, for the ICB_ε. We remind the reader that deciding whether or not a list of circles can be packed in only one bin is an NP-hard

problem. However, given a list of circles, with a constant number of circles, it is possible to obtain an algorithm that obtains a packing in polynomial time in an augmented bin of width $1 + \delta$ for such set, for any constant $\delta > 0$.

To do so, we use an algorithm that solves a semi-algebraic system. Let (L, r, W) be an instance of the ICB problem, where W is the width of the bins, with $|L| = n$. Note that the decision version of the ICB can be formulated as deciding if there are real non-negative numbers x_i, y_i, for $1 \leq i \leq n$, that satisfies the following constraints.

$$\begin{aligned} (x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2 \qquad & \text{for } 1 \leq i < j \leq n, \\ r_i \leq x_i \leq W - r_1 \qquad & \text{for } 1 \leq i \leq n, \text{ and} \\ r_i = y_i \qquad & \text{for } 1 \leq i \leq n. \end{aligned}$$

The first set of constraints guarantees that there is no overlap between every two circles. The second set of constraints ensures that the circles are entirely contained inside the bin where it is packed, and the last set of constraints ensures that every circle touches the bottom of the bin where it is packed. Observe that the set of solutions to the system of the inequalities presented above is a semi-algebraic set in the field of the real numbers. So, deciding if there is an in-line bin packing for (L, r, W) can be interpreted as deciding whether this semi-algebraic set is empty or not. Thus, we can use an algorithm that solves an algebraic system to obtain a solution for our problem. We do not go into details of this algorithm, since it is not in the scope of this work to discuss the algebraic technicalities necessary to perform such algorithm. Nonetheless, the interested reader may find a more detailed discussion in the works of Miyazawa et al. [18] and Grigorev and Vorobjov [13]. In the following, we present a result that lays on the solution of the semi-algebraic system mentioned above.

Lemma 4.15. *Given a list $L = (1, \ldots, n)$ of circles, with n constant and each circle $i \in L$ with radius r_i, a bin of width 1 and a constant $\delta > 0$, then there exists a polynomial-time algorithm $\mathcal{A}$ which decides if L has an in-line packing into one bin of width 1. In the affirmative case, the algorithm returns rational positions $x_1, \ldots, x_n$ such that the packing of each circle i at the base position x_i yields a feasible packing into one bin of width $1 + \delta$.*

Given, on one side, the difficulty of deciding if a list of circles can be packed into one bin (see Section 3.1), but, on the other side, the possibility of deciding, in polynomial time, if a list of circles packable into one bin can be packed in an augmented bin, we design an APTAS for the case when it is possible to use augmented bins.

Let L be a list of circles, with a constant number of circles. Our approximation scheme consists of three main steps. One, we must restrict the problem to a version with limited number of different radii; two, we obtain a packing for the restricted version into augmented bins using not more than $\mathrm{OPT}(L)$ bins; and three, based on the packing obtained in step two, we build a packing (not-necessarily optimal) for the original list of circles. For the first step, we use a technique presented by de la Vega and Lueker [9], in 1981, for the one-dimensional bin packing problem. The technique is known as *linear grouping*, and the idea is to partition the list of items into groups and then round each item up to the size of the largest item in its group. We will use this idea with circles. One of the advantages of this type of rounding

is to obtain a restricted instance where there is a bounded number of different radii, which is a type of instance that we wish to have. For the second step, we will use an algorithm that generates all possible packings in augmented bins and chooses one of minimum size. Finally, for the third step, we will perform a simple substitution of rounded circles to the original ones. Later in this section we explore the rounding from the first step. For now we suppose the instance has a constant number of different radii and analyze the algorithm that obtains a packing for such instance.

Note that when each circle has radius at least a value given by a constant, as it is in the case of the ICB_ε problem, then it is possible to know the maximum number of circles that fit together in one bin. If, additionally, we also have a constant number of different radii, then it is possible to compute all possible different ways of how to pack circles inside one augmented bin. We refer to such ways as configurations. Observe that we can obtain a packing by selecting a multiset of configurations. So, the idea is to generate all possible configurations, then generate all possible packings and, among the feasible ones, choose one of minimum size.

Lemma 4.16. *Given a list L of n circles with a constant number z of different radii, each at least a positive constant ε, then there is a polynomial-time algorithm that produces a packing of L into at most $\text{OPT}(L)$ augmented bins.*

Proof. Because we are considering unit bins, the maximum number of items that fit in one bin is $b = \lfloor \frac{1}{\varepsilon} \rfloor$. Let $D = \{\rho_1, \ldots, \rho_z\}$ be the set of different radii in L, and m_i be the number of circles of radius equals to ρ_i. Then $n_i = \min\{m_i, b\}$ is the maximum number of circles of radius i that fits within one bin. We can represent a packing configuration of a bin by a sequence $(c_1, \ldots, c_b)$ where each c_i is a circle.

We first show that the number of all possible configurations is bounded by a constant. Any configuration cannot have more than n_i circles of radius ρ_i, so it is sufficient to consider a restricted instance where we have n_i circles of radius ρ_i. That instance has $N = \sum_{i=1}^{z} n_i$ items. In order to allow configurations with less than b circles, we add b dummy circles (of diameter 0) to the restricted instance and always choose exactly b circles among the $N + b$ circles. This leads to $T = (z+1)^b$ possible configurations, which is constant since both z and b are constant numbers. Note that one of these T configurations is an empty packing. We can verify if each one of these T configurations can in fact be packed in an augmented bin, using algorithm $\mathcal{A}$ from Lemma 4.15. Among the T configurations, we discard the ones that lead to unfeasible packings, obtaining then T' valid configurations.

Finally, we can combine the T' configurations to obtain possible packings. Let $C_1, \ldots, C_{T'}$ be a combination of the T' configurations. Given an arbitrary packing, each configuration C_i is used from 0 to n times in this packing. As a consequence, the number of possible packings (already discarding the ones that do not represent a packing for the input list) is bounded by $(n+1)(n+1)\ldots(n+1)$, which gives us $(n+1)^{T'}$. Since $T' \leq T$ is constant, we have a polynomial-time algorithm to generate all possible packings. Moreover, the number of possible packings is also polynomial, so we can choose a packing of minimum size also in polynomial time. □

It is important to notice that an instance of the ICB_ε satisfies only the restriction of having a lower bound on the diameter of the circles. So, for the scheme, we first modify the original instance so that it attends also the restriction of having a limited number of different radii. Suppose that, for each circle in the original list, we create an equivalent circle of radius greater than or equal to the radius of the original circle. Now, if we find a packing for the modified circles, then to obtain a packing of the original list, it suffices to substitute each modified circle for its corresponding of original radius. To create such modified circles, we use the technique called *linear grouping*. Let L be the list of circles from the input. We first partition the circles of L into m groups. Then, we create a new list L' where, for each group, we round the radius of every circle up to the radius of the largest circle of its group. Formally, given a list G of circles, we denote by $\overline{G}$ the list obtained from G by equalizing the radius of each circle to the radius of the largest circle in G. Note that, because of the rounding, L' has only m different radii. Since we want an instance with a constant number of different radii, we must choose the size of the groups so that m is bounded by a constant. One way to guarantee this is to set the size of each group as a fraction of the number of items in the input instance. Finally, observe that, since the radii of circles in L' are greater than or equal to the radii of the corresponding circles in L, we can obtain a packing $\mathcal{P}$ of L using the base positions of a packing $\mathcal{P}'$ of L'. However, *a priori*, the packing $\mathcal{P}$ could possibly be much larger than an optimal packing of L, since the circles in L' are larger than or equal to the circles in L. We show that this difference of size between $\mathcal{P}$ and an optimal packing of L is only a factor of $1+\varepsilon$. In the following, we present an APTAS for the ICB_ε.

Algorithm 8: $\mathcal{A}^{\text{ICB}_\varepsilon}$: an APTAS for the ICB_ε problem.

Input: A tuple (L, r) where L is a list of circles, r is a function of the radii and ε is a positive function.

Output: A packing $\mathcal{P}$ of L with $|\mathcal{P}| \leq (1+\varepsilon)\text{OPT}(L)$.

1 Sort $L = (1, \ldots, n)$ in non-increasing order of radius ($r_i \geq r_{i+1}$ for $i = 1, \ldots, n-1$).
2 $k \leftarrow \lceil n\varepsilon^2 \rceil$.
3 Divide L into sublists such that $L = G_0 \| \ldots \| G_t$, where $|G_i| = k$, for $i = 0, 1, \ldots, t-1$, and $|G_t| \leq k$.
4 $L' \leftarrow \overline{G_1} \| \ldots \| \overline{G_t}$.
5 Find a packing $\mathcal{P}'$ of L' into bins of width $1+\delta$ (algorithm from Lemma 4.16).
6 Let $\mathcal{P}$ be a packing obtained from $\mathcal{P}'$ by replacing each copied circle in L' by its corresponding circle in L.
7 Obtain a packing $\mathcal{P}_0$ by placing each circle in G_0 alone in a new bin.
8 **return** $\mathcal{P}_0 \| \mathcal{P}$.

Theorem 4.17. *Algorithm $\mathcal{A}^{\text{ICB}_\varepsilon}$ is an APTAS for the problem ICB_ε, using augmented bins. More precisely, $\mathcal{A}^{\text{ICB}_\varepsilon}(L) \leq (1+\varepsilon)\text{OPT}(L) + 1$.*

Proof. We want to show that, given a constant $\varepsilon > 0$ and a list L of circles in which each circle has radius at least ε, there exists a family $\{A_\varepsilon\}$ of algorithms such that, for every value

of ε, it holds that $A_\varepsilon(L) \leq (1+\varepsilon)\text{OPT}(L) + O(1)$.

Let (L, r) be an instance of ICB_ε, where L is a list of n circles, r is a function of their radii, and ε is a positive constant, and let $k = \lceil n\varepsilon^2 \rceil$. The algorithm first sorts the list in non-increasing order of radius, i.e., for every $i \in L$, with $i = 1, \ldots, n-1$, it holds that $r_i \geq r_{i+1}$. Then, maintaining the sorting, the algorithm partitions L into $m+1$ sublists G_i such that $|G_i| = k$, for $i = 0, \ldots, m-1$, and $|G_m| \leq k$. So every sublist, except possibly the last one, has exactly k circles. Based on L, the algorithm creates another list $L' = \overline{G_1} \| \ldots \| \overline{G_m}$, where $\overline{G_i}$ is obtained from G_i with every circle's radius increased to the radius of the largest circle in G_i. Let J be a sublist of L obtained by excluding the k largest circles from L, i.e., $J = G_1 \| \ldots \| G_m$. Thus $L = G_0 \| J$. The algorithm first finds a packing for J and G_0 separately and then merge them together to obtain a packing for L.

To obtain a packing for J the algorithm starts from the packing $\mathcal{P}'$ for L', using the algorithm described in Lemma 4.16. Since L' is the rounded version of J, every circle in L' has radius larger than or equal to the radius of its corresponding circle in J. So, the algorithm builds a packing $\mathcal{P}_J$ of J by replacing each circle of $\mathcal{P}'$ by its corresponding circle in J. That way, both $\mathcal{P}'$ and $\mathcal{P}_J$ use the same number of bins and therefore we can conclude that

$$|\mathcal{P}_J| \leq \text{OPT}(L'). \tag{4.18}$$

Now we must find a relation between the optimal value of L' and the optimal value of L so that we have a relation between $\text{OPT}(J)$ and $\text{OPT}(L)$.

We show that $\text{OPT}(L') \leq \text{OPT}(L)$. To see this, note that any circle of $\overline{G}_{i+1}$ has radius smaller than or equal to the radius of a circle in G_i. Let $\mathcal{P}^\star$ be an optimal packing of L. Then we can build a packing $\mathcal{P}'$ of L' as follows: replace the circles of each sublist G_i in $\mathcal{P}^\star$ by the circles of the sublist $\overline{G}_{i+1}$, and then eliminate the circles from the sublist G_m. This gives us a feasible packing $\mathcal{P}'$ that uses the same number of bins of $\mathcal{P}^\star$. Therefore,

$$\text{OPT}(L') \leq |\mathcal{P}'| \leq |\mathcal{P}^\star| = \text{OPT}(L). \tag{4.19}$$

So, from inequalities 4.18 and 4.19, we have

$$|\mathcal{P}_J| \leq \text{OPT}(L). \tag{4.20}$$

Lastly, the algorithm packs the circles of G_0 by opening a new bin for each circle, obtaining the packing $\mathcal{P}_0$. Note that the packing $\mathcal{P}_0$ uses k bins. From the input, we know each circle has radius at least ε. If all the n circles have radius equals to ε, then the optimal packing is given by every circle placed side by side inside the bins, and the number of bins used is given by the ceiling of $2n\varepsilon$ divided by the width of a bin, which is one. Therefore $\lceil 2n\varepsilon \rceil$ is a lower bound on the optimal value. Since $\mathcal{P}_0$ uses k bins, we have

$$|\mathcal{P}_0| = \lceil n\varepsilon^2 \rceil \leq \varepsilon(\varepsilon n) + 1 \leq \frac{\varepsilon}{2}\text{OPT}(L) + 1. \tag{4.21}$$

Finally, when we merge the packings $\mathcal{P}_J$ and $\mathcal{P}_0$, we obtain a packing $\mathcal{P}_J \| \mathcal{P}_0$ and, from

inequalities 4.20 and 4.21, it holds that

$$|\mathcal{P}_J| + |\mathcal{P}_0| \leq \mathrm{OPT}(L) + \frac{\varepsilon}{2}\mathrm{OPT}(L) + 1. \tag{4.22}$$

□

4.3.2 A PTAS for ICP of Large Circles

We first present an APTAS for the ICP_ε problem. To accomplish an APTAS for the ICP_ε problem, we will use as a subroutine the algorithm $\mathcal{A}^{\mathrm{ICB}_\varepsilon}$, which is an APTAS for the ICB_ε problem, as discussed previously. Without loss of generality, we consider that the largest circle has radius equal to one.

Algorithm 9: $\mathcal{A}^{\mathrm{ICP}_\varepsilon}$: an APTAS for the ICP_ε problem, where $\varepsilon \leq 1/8$.

Input: An instance (L, r) of the ICP_ε, where r is a function of the radii and ε is a positive constant.

Output: An in-line packing $\mathcal{P}$ of L.

1 $\mathcal{P}_B \leftarrow \mathcal{A}^{\mathrm{ICB}_\varepsilon}(L, W)$, where $W = 1/\varepsilon$.

2 Let $\mathcal{P}$ be a packing obtained from $\mathcal{P}_B$ by concatenating the in-line packings inside each bin of $\mathcal{P}_B$.

3 **return** $\mathcal{P}$.

Since we will refer to the optimum of both the ICP_ε and the ICB_ε problems, from now on we denote by $\mathrm{OPT}_B(L)$ the value of an optimal in-line bin packing of L and by $\mathrm{OPT}_I(L)$, the value of an optimal in-line packing of L.

Theorem 4.18. *There exists an APTAS for the* ICP_ε *problem. More precisely, there is an algorithm A for the* ICP_ε *problem such that $\mathcal{A}(L) \leq (1+\varepsilon)\mathrm{OPT}(L)+4/\varepsilon$, for any instance L.*

Proof. We show that the algorithm $\mathcal{A}^{\mathrm{ICP}_\varepsilon}$ is an APTAS for the ICP_ε problem. More precisely, we show that, given a constant $0 < \varepsilon \leq 1/8$ and a list of circles L in which each circle has radius at least ε, there exists a family $\{A_\varepsilon\}$ of algorithms such that, for every value of ε, the inequality $A_\varepsilon(L) \leq (1+\varepsilon)\mathrm{OPT}_I(L) + 1$ holds.

The first step of our proof is to find a relation between $\mathrm{OPT}_B(L)$ and $\mathrm{OPT}_I(L)$. Given an optimal in-line packing $\mathcal{P}_I^\star$ of L, we construct an in-line bin packing $\mathcal{P}_B$ of L as follows. We cut the packing $\mathcal{P}_I^\star$ with vertical lines $\ell_0, \ldots, \ell_m$ interspaced by a distance W, where the first line, ℓ_0, crosses the origin of the plane, i.e., the leftmost point of the packing. Each one of the following lines, $\ell_1, \ldots, \ell_m$, might cut a set of circles. We say a vertical line cuts a circle if it touches the circle in more than one point. We denote by $\mathcal{B}_i$ the packing extracted from $\mathcal{P}_I^\star$ of the set of circles cut by line ℓ_i, for $1 \leq i \leq m$, and we refer to each $\mathcal{B}_i$ as a block. Moreover, we denote by $\mathcal{B}'_i$ the packing extracted from $\mathcal{P}_I^\star$ of the set of circles whose interior is entirely contained between the lines ℓ_{i-1} and ℓ_i, for $1 \leq i \leq m$. Note that $\mathcal{B}'_i$ includes the circles that touches one of the two lines in exactly one point and excludes the circles that

are part of some block. Then, to pack the circles that are not part of any block, it suffices to put each packing $\mathcal{B}'_i$ into one bin, for $1 \leq i \leq m$. This leads to

$$\left\lceil \frac{|\mathcal{P}_I^\star|}{W} \right\rceil$$

bins.

It remains to pack into bins the circles that are part of some block. To do that, we repeatedly concatenate blocks inside one bin, until the concatenation of the next block results in a packing of size greater than W. In this case, the last block is packed in a new bin, which will become the current bin considered from now on. Because the radius of each circle is at most one, we have that the size $|\mathcal{B}_i| < 4$, for $1 \leq i \leq m$. Note that $\varepsilon < 1/8$, so a bin has size at least $\frac{2}{1/8} = 4$ and therefore it can accommodate at least one block. Furthermore, each bin contains at least $\left\lfloor \frac{W}{4} \right\rfloor$ blocks, except possibly the last bin, that may contain fewer blocks. Since each block comes from a vertical line, we have at most $\frac{|\mathcal{P}_I^\star|}{W}$ blocks. Thus, packing the circles which are part of some block as we described above yields an in-line bin packing using at most

$$\left\lceil \frac{\frac{|\mathcal{P}_I^\star|}{W}}{\left\lfloor \frac{W}{4} \right\rfloor} \right\rceil$$

bins.

Let $\mathcal{P}_B$ be a packing of L obtained as we described above. Because $\mathcal{P}_B$ is a feasible in-line bin packing of L, we know that $\mathrm{OPT}_B(L) \leq |\mathcal{P}_B|$. Therefore, replacing W by $1/\varepsilon$, we have

$$\begin{aligned}
\mathrm{OPT}_B(L) &\leq |\mathcal{P}_B| \\
&\leq \left\lceil \frac{|\mathcal{P}_I^\star|}{1/\varepsilon} \right\rceil + \left\lceil \frac{\frac{|\mathcal{P}_I^\star|}{1/\varepsilon}}{\left\lfloor \frac{1/\varepsilon}{4} \right\rfloor} \right\rceil \\
&\leq \frac{|\mathcal{P}_I^\star|}{1/\varepsilon} + 1 + \left\lceil \frac{\varepsilon |\mathcal{P}_I^\star|}{\frac{1/\varepsilon}{8}} \right\rceil \qquad (4.23) \\
&\leq \varepsilon \mathrm{OPT}_I(L) + 1 + \lceil 8\varepsilon^2 \mathrm{OPT}_I(L) \rceil \\
&\leq \varepsilon \mathrm{OPT}_I(L) + 8\varepsilon^2 \mathrm{OPT}_I(L) + 2. \qquad (4.24)
\end{aligned}$$

Note that the removal of the floors in inequality (4.23) is valid because $\varepsilon \leq \dfrac{1}{8}$. Now that we have a relation between $\mathrm{OPT}_B(L)$ and $\mathrm{OPT}_I(L)$, we shall analyze the algorithm. First, it obtains an in-line bin packing $\mathcal{P}_B$ of L using augmented bins of width $W + \varepsilon$ by applying algorithm $\mathcal{A}^{\mathrm{ICB}_\varepsilon}$. By Theorem 4.17, we have

$$|\mathcal{P}_B| \leq (1 + \varepsilon)\mathrm{OPT}_B(L) + 1. \qquad (4.25)$$

Observe that each augmented bin contains an in-line packing. So, from the packing $\mathcal{P}_B$, the algorithm constructs an in-line packing $\mathcal{P}$ of L by concatenating the in-line packings contained in the augmented bins used in $\mathcal{P}_B$. Thus, the span of $\mathcal{P}$ is at most the number of augmented bins used in $\mathcal{P}_B$ times the width of the bins, i.e., $|\mathcal{P}| \leq |\mathcal{P}_B|(W+\varepsilon)$. Therefore, replacing W by $1/\varepsilon$, we have

$$
\begin{aligned}
|\mathcal{P}| &\leq |\mathcal{P}_B| \left(\frac{1}{\varepsilon} + \varepsilon\right) \\
&\leq \left((1+\varepsilon)\mathrm{OPT}_B(L) + 1\right) \left(\frac{1}{\varepsilon} + \varepsilon\right) \\
&\leq \left((1+\varepsilon)(\varepsilon\mathrm{OPT}_I(L) + 8\varepsilon^2\mathrm{OPT}_I(L) + 2) + 1\right) \left(\frac{1}{\varepsilon} + \varepsilon\right) \\
&= \left((\varepsilon\mathrm{OPT}_I(L) + 8\varepsilon^2\mathrm{OPT}_I(L) + 2) + (\varepsilon^2\mathrm{OPT}_I(L) + 8\varepsilon^3\mathrm{OPT}_I(L) + 2\varepsilon) + 1\right) \left(\frac{1}{\varepsilon} + \varepsilon\right) \\
&\leq \mathrm{OPT}_I(L) + 9\varepsilon\mathrm{OPT}_I(L) + 9\varepsilon^2\mathrm{OPT}_I(L) + 9\varepsilon^3\mathrm{OPT}_I(L) + 8\varepsilon^4\mathrm{OPT}_I(L) + 3\varepsilon + 2\varepsilon^2 + \frac{3}{\varepsilon} + 2 \\
&\leq \mathrm{OPT}_I(L) + 10\varepsilon\mathrm{OPT}_I(L) + \frac{4}{\varepsilon},
\end{aligned}
$$

where the last inequality is valid because $\varepsilon < 1/8$. □

Now we propose a PTAS for the ICP_ε problem. To do that, we must eliminate the additive constant appearing in the APTAS we have just shown. The idea is to use the algorithm $\mathcal{A}^{\mathrm{ICP}_\varepsilon}$ as a subroutine, but only in cases where the size of the instance is large enough so that the additive constant is just a factor of ε of the optimum value. In this case, the additive constant in the APTAS named $\mathcal{A}^{\mathrm{ICP}_\varepsilon}$ is of the form $\frac{c}{\varepsilon}$, where c is a positive integer. The fact that the additive constant depends on ε is why it is possible to reduce it to a factor of ε of the optimum value. More precisely, we aim to find the size of the instance L for which it holds that $\varepsilon\mathrm{OPT}_I(L) \geq \frac{1}{\varepsilon}$. One size that guarantees this inequality is $|L| = \frac{1}{\varepsilon^3}$. Consider the algorithm $\mathcal{A}_P^{\mathrm{ICP}_\varepsilon}$.

Algorithm 10: $\mathcal{A}_P^{\mathrm{ICP}_\varepsilon}$: a PTAS for the ICP_ε problem.

Input: A tuple (L, r) where L is a list of circles, r is a function of the radii, and a positive constant $\varepsilon < \frac{1}{8}$.

Output: An in-line packing $\mathcal{P}$ of L.

if $|L| > 1/\varepsilon^3$ **then**

  $\mathcal{P} \leftarrow \mathcal{A}^{\mathrm{ICP}_\varepsilon}(L, r)$.

else

  Run a binary search in the interval $[0, |L| \max_i 2r_i]$ until the search reaches an interval $[a, b]$ such that $(b - a) \leq \varepsilon^2/3$, where algorithm $\mathcal{A}$ of Lemma 4.15, cannot pack L in a bin of size a, but can pack L in a bin of size $b + \delta$, where $\delta = \varepsilon^2/3$.

  Let $\mathcal{P}$ be a packing obtained by this binary search procedure.

return $\mathcal{P}$.

Theorem 4.19. *There exists a PTAS for the* ICP_ε *problem.*

Proof. We show that the algorithm $\mathcal{A}_P^{\mathrm{ICP}_\varepsilon}$ is a PTAS for the ICP_ε problem. More precisely, we show that, given a constant $0 < \varepsilon \leq 1/8$ and a list of circles L in which each circle has radius at least ε, there exists a family $\{A_\varepsilon\}$ of algorithms such that, for every value of ε, the inequality $A_\varepsilon(L) \leq (1 + \varepsilon)\mathrm{OPT}_I(L)$ holds.

As it happens in the algorithm, we divide the proof in two cases. The first case is when the size of the input list is greater than $1/\varepsilon^3$. Since the radius of each circle is at least ε, we have that $\mathrm{OPT}_I(L) \geq 2|L|\varepsilon$. Thus,

$$\mathrm{OPT}_I(L) \geq 2\varepsilon|L| > \frac{2}{\varepsilon^2}. \tag{4.26}$$

From the inequality 4.26, we can conclude that

$$\varepsilon\mathrm{OPT}_I(L) > \frac{2}{\varepsilon}. \tag{4.27}$$

The algorithm $\mathcal{A}_{\mathrm{P}}^{\mathrm{ICB}_\varepsilon}$ produces the packing $\mathcal{P}$ by applying the algorithm $\mathcal{A}^{\mathrm{ICP}_\varepsilon}$ over (L, r). Therefore, by Theorem 4.18, we have

$$\begin{aligned} |\mathcal{P}| &\leq (1 + 10\varepsilon)\mathrm{OPT}_I(L) + \frac{4}{\varepsilon} \\ &< (1 + 10\varepsilon)\mathrm{OPT}_I(L) + 2\varepsilon\mathrm{OPT}_I(L), \qquad (4.28) \\ &= (1 + 12\varepsilon)\mathrm{OPT}_I(L), \end{aligned}$$

where inequality (4.28) comes from inequality (4.27). Therefore, we have a packing $\mathcal{P}$ which has span at most $(1 + O(\varepsilon))$ of the optimal value, and the result follows.

The second case is when the size of the input list is less than or equal to $\frac{1}{\varepsilon^3}$. In this case,

we have at most $\frac{1}{\varepsilon^3}$ circles, which gives us an instance with a number of circles bounded by a constant, since ε is a constant. Let $\mathcal{P}$ be a packing obtained by the binary search procedure as described in the algorithm $\mathcal{A}_P^{\text{ICP}_\varepsilon}$. When the search stops, we have the following scenario. On one side, since algorithm $\mathcal{A}$ cannot pack L into a bin of size a, we have $\text{OPT}(L) \geq a - \varepsilon^2/3$, since $\delta = \varepsilon^2/3$. On the other side, as algorithm $\mathcal{A}$ can pack L into a bin of size $b + \varepsilon^2/3$, we have

$$\begin{aligned}
|\mathcal{P}| &\leq b + \varepsilon^2/3 \\
&\leq (a + \varepsilon^2/3) + \varepsilon^2/3 \\
&= a + 2\varepsilon^2/3 \\
&\leq \text{OPT}(L) + \varepsilon^2/3 + 2\varepsilon^2/3 \\
&= \text{OPT}(L) + \varepsilon^2 \\
&\leq \text{OPT}(L) + \varepsilon\text{OPT}(L),
\end{aligned}$$

where the last inequality is valid because $\text{OPT}(L) \geq \varepsilon$, as each circle has radius at least ε.

It remains to argue that the algorithm takes polynomial time. In the first case, the algorithm runs another algorithm over L, which is an APTAS for the ICP_ε problem, therefore, it takes polynomial time in this case. In the second case, the algorithm performs a binary search, which has polynomial time, and at each step of this search, it performs another polynomial time algorithm, so the total time is also polynomial. □

Chapter 5

Final Remarks

In this dissertation, we considered the In-line Circle Packing (ICP) problem, which we found to be not yet widely exploited in the literature. To the date of this writing, we have found only two works in the literature, one regarding triangles and another regarding circles. As a consequence, we reason that there is still plenty of space left for investigations. For the triangles, Dürr et al. [7] showed a QPTAS. One natural question is: is there a PTAS for such problem? If so, could it be generalized for other shapes? The same question is appropriate for the in-line packing of circles. Although we were not capable to find a PTAS for any of these two problems, we believe that they admit such approximation scheme. We particularly studied the version for circles. Our main challenge was determining a convenient lower bound for the value of an optimal solution. Alt et al. [1] introduced the idea of a so called *support interval* of a circle, and then proved a lower bound based on such concept. We tried to use their lower bound to design an approximation scheme but we did not succeed. The hazard was that the lower bound was not close enough to the optimal value.

The in-line packing problem (ICP) somewhat resembles the level strip packing problem, where the items must be packed within a strip but in levels, as if organizing objects in a shelf. The resemblance lays on the fact that, in the level strip packing, all items packed in a level must touch the bottom of that level. In this dissertation, we proposed a different version of the two-dimensional bin packing problem where every item has to touch the bottom of the bin in which it is packed. The same restriction can be extended to the knapsack problem. We can also consider recipients of other shapes. For example, consider a problem where the items are circles and the recipient is an ellipse. We can add the in-line restriction by requiring that every circle must touch the contour of the ellipse from inside, at exactly one point. The idea can be extend to basically any regular forms, taken as items and also as recipients. Another direction of generalizations may be considering higher dimensions. For instance, we can take spheres and try to pack them over a plane. We believe that, in a sense, we can consider n-dimensional items and try to pack them over an $(n-1)$-dimensional surface. Finally, despite not being regarded in this work, the ICP problem can be extended to versions where the items may touch the surface both from above or from below. One similar problem is the one considered by Buchin et al. [2], which they referred to as ordered strip packing. In this problem, the input is a maximum width W (representing the strip), and an ordered list of blocks of rectangles. In each block the rectangles are not horizontally

aligned and the objective is to pack the blocks into rows no larger than W, minimizing either the number of rows or the height of the strip.

As we briefly commented, the ICP problem has a very simple formulation, and admits several variants and generalization. However, we did not find many works in the literature about related problems and the investigations seem to be recent. The two works we found in the literature are both from 2018. The possibilities of exploitation are plentiful, and we remain interested in the problem.

Bibliography

[1] Helmut Alt, Kevin Buchin, Steven Chaplick, Otfried Cheong, Philipp Kindermann, Christian Knauer, and Fabian Stehn. Placing your coins on a shelf. *Journal of Computational Geometry*, 9:312–327, 2018.

[2] K. Buchin, D. Kosolobov, W. Sonke, B. Speckmann, and K. Verbeek. Ordered strip packing. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *LATIN 2020: Theoretical Informatics*, pages 258–270, Cham, 2020. Springer International Publishing.

[3] Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[4] E. G. Coffman, Garey M.R., and Johnson D.S. *Approximation Algorithms for Bin Packing Problems: A Survey*, volume 266. 1981.

[5] E. G. Coffman, Garey M.R., and Johnson D.S. *Approximation Algorithms for Bin-Packing — An Updated Survey*, volume 284. 1984.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] Christoph Dürr, Zdeněk Hanzálek, Christian Konrad, Yasmina Seddik, René Sitters, Óscar C. Vásquez, and Gerhard Woeginger. The triangle scheduling problem. *Journal of Scheduling*, 21:305–312, 2018.

[8] P. Erdős. Gráfok páros körüljárású részgráfjairól (on bipartite subgraphs of graphs). *Matematikai Lapok*, 18:2–85, 1967.

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

[10] Michael R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, pages 143–150. ACM, 1972.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[12] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966.

[13] D. Yu. Grigor'ev and N.N. Vorobjov. Solving systems of polynomial inequalities in subexponential time. *Journal of Symbolic Computation*, 5(1):37–64, 1988.

[14] Rolf Harren, Klaus Jansen, Lars Prädel, and Rob van Stee. A $(5/3+\varepsilon)$-approximation for strip packing. *Computational Geometry*, 47(2, Part B):248–267, 2014. The 12th International Symposium on Algorithms and Data Structures (WADS2011).

[15] Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 177–189, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[16] David S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 38–49. ACM, 1973.

[17] Andrea Lodi, Silvano Martello, Michele Monaci, and Daniele Vigo. *Two-Dimensional Bin Packing Problems*, pages 107–129. Wiley & Blackwell, 2014.

[18] 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. *Algorithmica*, 76(2):536–568, 2016.

[19] J. Iacono S. R. Allen. Packing identical simple polygons is NP-hard. *CoRR*, abs/1209.5307, 2012.

[20] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997.