



**Elias Fukim Lozano Ching**

**A geometric algorithm to generate random  
polydisperse dense arrangements of non  
over-lapping disk particles**

**Tese de Doutorado**

Thesis presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática.

Advisor: Prof. Marcelo Gattass

Rio de Janeiro  
August 2020



**Elias Fukim Lozano Ching**

**A geometric algorithm to generate random  
polydisperse dense arrangements of non  
over-lapping disk particles**

Thesis presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the Examination Committee.

**Prof. Marcelo Gattass**

Advisor

Departamento de Informática – PUC-Rio

**Prof. Anselmo Antunes Montenegro**

Departamento de Ciência da Computação – UFF

**Prof. Creto Augusto Vidal**

Departamento de Ciência da Computação – UFC

**Dr. Rodrigo de Souza Lima Espinha**

Instituto Tecgraf – PUC-Rio

**Prof. Waldemar Celes Filho**

Departamento de Informática – PUC-Rio

Rio de Janeiro, August 14th, 2020

All rights reserved.

**Elias Fukim Lozano Ching**

Graduated from the Universidad Nacional Mayor de San Marcos (UNMSM) in Systems engineering. Master in Informatics by the Pontifical Catholic University of Rio de Janeiro (PUC-Rio)

Bibliographic data

Lozano Ching, Elias Fukim

A geometric algorithm to generate random polydisperse dense arrangements of non over-lapping disk particles / Elias Fukim Lozano Ching; advisor: Marcelo Gattass. – 2020.

116 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2020.

Inclui bibliografia

1. Informática – Teses. 2. Empacotamento de partículas. 3. Frente de avanço. 4. Algoritmo geométrico. 5. Malha DEM inicial. I. Gattass, Marcelo. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To Doris Ching, my mother.

## Acknowledgments

I would first like to thank my advisor, Prof. Marcelo Gattass, for providing invaluable guidance and feedback throughout this research.

I thank Tecgraf/PUC-Rio Institute, for the financial support, without which this work would not have been realized.

Last but not least, I would like to thank my family, my parents, Doris and Elias; my brothers, Kiway and Jou; and my niece Camila for their support and encouragement.

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

## Abstract

Lozano Ching, Elias Fukim; Gattass, Marcelo (Advisor). **A geometric algorithm to generate random polydisperse dense arrangements of non over-lapping disk particles**. Rio de Janeiro, 2020. 116p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This work aims to present a new strategy for the non-overlapping disk packing problem to generate dense random assemblies. The geometric algorithm adopts an advancing front approach that uses new heuristics to determine the next positions for the incoming particles with the support of a polygonal mesh. Furthermore, we propose relocation schemes to improve the packing at the pack's interior and near the container borders. Experiments prove that our algorithm outperforms previous results, w.r.t the desired particle radii distribution function and increases the packing density and mean number of particle contacts.

## Keywords

Particle packing; Advancing-front approach; Geometric algorithm; Initial DEM mesh.

## Resumo

Lozano Ching, Elias Fukim; Gattass, Marcelo. **Um algoritmo geométrico gerador de arranjos polidispersos densos de discos sem sobreposição**. Rio de Janeiro, 2020. 116p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O objetivo deste trabalho é apresentar uma nova estratégia para o problema de empacotamento de discos sem sobreposição para gerar arranjos aleatórios densos. O algoritmo geométrico adota uma abordagem frente de avanço que, com o apoio de uma malha poligonal, utiliza novas heurísticas para determinar as próximas posições para as próximas partículas. Além disso, propomos esquemas de realocação para melhorar o empacotamento no interior do arranjo e perto das bordas dos objetos arbitrários que contêm as partículas. Os resultados provam que nosso algoritmo pode superar trabalhos anteriores, não apenas com a função de distribuição de raios de partículas desejada, mas também aumentando a densidade de empacotamento e o número médio de contatos.

## Palavras-chave

Empacotamento de partículas; Frente de avanço; Algoritmo geométrico; Malha DEM inicial.

## Table of contents

<b>1</b>	<b>Introduction</b>	<b>18</b>
1.1	Motivation	18
1.2	Two-dimensional random disk packing	20
1.3	Goals	21
1.4	Contributions	22
1.5	Organization	23
<b>2</b>	<b>Previous Work</b>	<b>24</b>
2.1	Geometric separation methods	25
2.2	Apollonian based methods	26
2.3	Mesh based methods	26
2.4	Advancing front methods	27
2.5	Mesh support in disk packing	30
<b>3</b>	<b>Topological and Spatial Data Structures</b>	<b>35</b>
3.1	Particle grid index	35
3.2	Polygonal mesh	37
3.3	Boundary detection	44
<b>4</b>	<b>Particle generation</b>	<b>50</b>
4.1	Outward strategy	50
4.2	Metrics	54
<b>5</b>	<b>Packing improvements</b>	<b>58</b>
5.1	Internal strategy (IS)	58
5.2	Outer loop strategy (OLS)	60
5.3	Boundary strategy (BS)	66
<b>6</b>	<b>Results</b>	<b>69</b>
6.1	Geometric characteristics	69
6.2	Algorithm variants	71
6.3	Analysis of the front removal parameters	72
6.4	Monodisperse and bidisperse arrangements	73
6.5	Variants comparison	75
6.6	Analysis of the OLS	81
6.7	Analysis of the BS	83
6.8	Stability tests	84
6.9	Benchmark with other approaches	85
6.10	Complex geometries	91
<b>7</b>	<b>Conclusions</b>	<b>100</b>
7.1	Future work	102
	<b>Bibliography</b>	<b>104</b>



<b>A</b>	<b>Mesh operations</b>	<b>113</b>
<b>B</b>	<b>Distance field for simple geometries</b>	<b>114</b>
<b>C</b>	<b>Flowcharts</b>	<b>115</b>

## List of figures

Figure 1.1	DEM application examples.	19
Figure 1.2	Regular packings.	20
Figure 1.3	In 2D, disks tend to form hexagonal patterns. (Source: [Aste <i>et al.</i> 2005])	21
Figure 2.1	Categories of geometric methods.	25
Figure 2.2	Wang's advancing front pack (modified from [Wang <i>et al.</i> 2007])	30
Figure 2.3	Liu's front loops (modified from [Liu 2008])	31
Figure 2.4	Liu's pack and mesh triangulation (modified from [Liu 2008])	31
Figure 2.5	Benabbou's packs produces packs with heterogeneous local density (modified from [Benabbou <i>et al.</i> 2008])	32
Figure 2.6	Liu's local Delaunay Tessellation (modified from [Liu <i>et al.</i> 2012])	33
Figure 2.7	Spetch's contact matrix (modified from [Specht 2015])	33
Figure 3.1	Neighborhood box logic around the current front.	36
Figure 3.2	Four particles, connected through four edges, define an irregular quad.	37
Figure 3.3	Left-hand rule criterion. Starting a path from $A \rightarrow B$ , the criterion closes the polygon ABCD.	39
Figure 3.4	Insertion on the outer loop.	40
Figure 3.5	Sequence of polygon search for three iterations.	41
Figure 3.6	Insertion in the outer loop (Continuation).	41
Figure 3.7	A polygonal mesh created along with a small pack.	42
Figure 3.8	Insertion in polygons. a) The particle F is placed inside the pentagon ABCDE. b) A temporal graph with the new contacts and the polygon.	43
Figure 3.9	Insertion in polygons. a) The left-hand rule finds two paths. b) The insertion yields two new polygons FDEAB and FBCD.	43
Figure 3.10	Particles $A$ and $C$ are boundary vertices until the insertion of particle $H$ . The <b>VertexStar(A)</b> procedure returns the particles B, C, D and E.	44
Figure 3.11	Tree subdivision – Dragon head.	48
Figure 3.12	Signed distance field – Dragon.	48
Figure 3.13	Slice of Micro-CT Sand pack LV60C. In black the solid phase and in white the porous space.	49
Figure 3.14	Signed distance field – Solid phase in a slice of Micro-CT Sand pack LV60C.	49
Figure 4.1	Placing a particle in contact with two other particles.	52
Figure 4.2	Particle B yields a single candidate position. Particle C yields two positions. Particle D and E do not generate points	52

Figure 4.3	Geometric position of the candidate points product of the intersection of both halos.	53
Figure 5.1	The maximum size of a disk tangent to the particles $A$ and $B$ depends on the closest third particle $C$ .	59
Figure 5.2	A pentagon formed by particles ABCDE with three Soddy circles without collisions.	59
Figure 5.3	OLS Example 1. Withdrawal step.	61
Figure 5.4	OLS Example 1 (Continuation). Good relocation with twelve triangles.	66
Figure 5.5	OLS Example 2. Bad relocation.	67
Figure 5.6	Border treatment example – First iteration.	67
Figure 5.7	Border treatment example – Second iteration.	68
Figure 6.1	Frequency histogram.	72
Figure 6.2	Front removal logic	73
Figure 6.3	Monodisperse pack inside a circle of $10u$ radius with disks of radius $0.25u$ with a refilling procedure ( $0.15u$ minrad).	74
Figure 6.4	Bidisperse pack of instance N°35 – Radius and contact frequencies.	74
Figure 6.5	Bidisperse pack of instance N°35 – Ratio 1:1.4	75
Figure 6.6	Uniform packs inside a rectangular container with three variants using the same data set.	76
Figure 6.7	Lognormal packs inside a circular container with four variants using the same data set.	77
Figure 6.8	Log-normal packs inside a circular container with four variants using the same data set (continuation).	77
Figure 6.9	Verification of radius frequencies and contact orientations for the log-normal test with AFMeshOLSIS	78
Figure 6.10	Variant comparison with different ratios.	78
Figure 6.11	Particles ( $N$ ) vs Time(s) for 1:2 and 1:3 ratios.	79
Figure 6.12	Particles ( $N$ ) vs Time(s) for the 1:4 ratio.	80
Figure 6.13	Particles ( $N$ ) vs Time(s) for 1:5 and 1:6 ratios.	80
Figure 6.14	Clustering of algorithm's outputs.	82
Figure 6.15	Clustering of algorithm's outputs (continuation).	82
Figure 6.16	Clustering of algorithm's outputs (continuation).	83
Figure 6.17	Border improvement. From top to bottom $Br_{min} = 0.20, 0.10, 0.08, 0.06, 0.04$ . Added particles in cyan.	84
Figure 6.18	Experiment with $Br_{min}$ .	84
Figure 6.19	Initial packs for simulations	85
Figure 6.20	Final packs after simulations.	85
Figure 6.21	Comparison 1 – Pack and mesh	87
Figure 6.22	Comparison 2 – AFMeshOLSIS instance N°4	88
Figure 6.23	AFMesh pack densification with IS (post process) + BS using $Br_{min} = 0.065$ . IS and BS particles are in red and cyan respectively.	91
Figure 6.24	Zhang's slope model.	92
Figure 6.25	Zoom at the slope model. Pack and mesh.	92
Figure 6.26	Slope model – Radius frequencies.	92

Figure 6.27	Dragon model.	93
Figure 6.28	Merging two fronts. Active fronts in red.	93
Figure 6.29	Dragon detail. Added particle with the BS improvement in cyan.	94
Figure 6.30	Octopus model	94
Figure 6.31	Octopus detail.	95
Figure 6.32	Octopus detail (Continuation).	95
Figure 6.33	Extracted models.	97
Figure 6.34	Extracted models (Continuation).	97
Figure 6.35	Pack on images.	98
Figure 6.36	Pack on images (Continuation).	98
Figure 6.37	LV60C and carbonate pack zooms.	99
Figure 6.38	Berea and ketton pack zooms.	99
Figure C.1	Packing algorithm flowchart.	115
Figure C.2	Radius rejection and front removal logic.	116

## List of tables

Table 3.1	Distance field - Models.	47
Table 6.1	Bidisperse packs with AFMeshOLS – Average of 100 instances.	74
Table 6.2	Uniform pack results summary.	76
Table 6.3	Lognormal pack results summary.	76
Table 6.4	Combination of relocation triggers in 16 scenarios.	81
Table 6.5	Boundary strategy results varying the $Br_{min}$ value.	83
Table 6.6	Benabbou desired frequencies	86
Table 6.7	Our frequencies with the highest and lowest chi-square value.	86
Table 6.8	Comparison 1 – Summary of 100 instances.	86
Table 6.9	Comparison 2 – Literature results.	87
Table 6.10	Comparison 2 – AFMeshOLS – Summary of 100 instances.	88
Table 6.11	Comparison 2 – AFMeshOLSIS – Summary of 100 instances.	88
Table 6.12	Comparison 2 – Polygon count of AFMeshOLSIS instance N°4.	89
Table 6.13	Comparison 2 – Relocations and front logic output data.	89
Table 6.14	Comparison 3 – Literature results.	90
Table 6.15	Comparison 3 – Densification results.	90
Table 6.16	Dual contouring inputs and outputs.	96
Table 6.17	Summary of rock image packing.	96

## List of algorithms

1	ADF construction	47
2	Assembly Generation Algorithm	55
3	HandleFrontRemoval	55
4	ProcessFronts( <i>newFronts</i> , $L_p$ )	63
5	SearchCandidates( $f$ , $l_p$ , <i>ToRemove</i> , <i>candidates</i> , <i>bestCandidate</i> )	64
6	Update( <i>newParticle</i> , <i>fronts</i> , $L_p$ , <i>candidates</i> , <i>bestCandidate</i> )	64
7	Relocate( <i>pack</i> , <i>grid</i> , <i>mesh</i> , <i>fronts</i> , $L_p$ )	65
8	Circle distance field	114
9	Rectangle distance field	114

## List of Symbols

$N$  – Number of particles in the pack

$r_{min}$  – Minimum radius in the pack

$r_{max}$  – Maximum radius in the pack

PDF – Distribution function for the radii creation

$r_{new}$  – Radius for the new particle

$r_{curr}$  – Radius of the current particle in the loop

$r_{halo}$  – Radius of the halo of a particle

$\delta_{box}$  – Size of the neighborhood box

$qnr$  – Queue of newly rejected radii

$qpr$  – Queue of previously rejected radii

DF – Distance to first metric

IN – Inside outer loop metric

NP – Number of polygon metric

PS – Polygon size metric

$\psi_1$  – Number of sides of a polygon

ADF – Adaptively distance field

OLS – Outer loop relocation strategy

IS – Internal insertion strategy

BS – Boundary insertion strategy

$Br_{min}$  – Minimum radius used in the BS strategy

AF16 – The algorithm described in Lozano *et al.* 2016

AFMesh – Current algorithm with the new heuristics

AFMeshIS – The AFMesh algorithm with the IS improvement

AFMeshOLS – The AFMesh algorithm with the OLS improvement

AFMeshOLSIS – The AFMeshOLS algorithm with the IS improvement





*A complex structure is the result of and to a  
large extend the records of its past.*

**Cyril Stanley Smith (1903-1992),** *The pursuit of perfect packing.*

# 1

## Introduction

Granular materials are ubiquitous in nature. For the last decades, it has been a topic of active research from different perspectives due to its academic and industrial importance. The research scope ranges from mining, 3d printing, to pharmaceutical processing. Depending on the research area, granular material varies from small scales as molecules, to dust, even up to celestial bodies.

Among the numerical methods that study granular materials, there is the Discrete Element Method (DEM) formulated by [Cundall *et al.* 1979]. The method assumes that granular medium, within a domain, can be represented by a large set of individual particles. Particles can be modeled with different sizes and shapes, but are commonly represented by disks in 2D or spheres in 3D. The particles interact with each other, and the method predicts positions, velocities, accelerations, and contacts of all the particles by solving Newton's equations of motion.

### 1.1

#### Motivation

The DEM has a wide range of applications. Among the recent, we identify a few: modeling of grinding wheels [Osa *et al.* 2018]; modeling of brittle elastic materials [Nguyen *et al.* 2019]; simulation of ice floes and floating structures [Liu *et al.* 2018]; synthetic rock mass modeling [Vallejos *et al.* 2016]; manufacturing process simulation in the pharmaceutical industry [Yeom *et al.* 2019]; modeling fracture in rock [Azevedo *et al.* 2013]; bi-dimensional simulation of scree-slope dynamics [Bithell *et al.* 2014]; simulation of earthquake surface fault rupture [Taniyama 2017, Garcia 2018]; simulation of sheared granular fault systems with a two-dimensional DEM combined with FDEM [Gao *et al.* 2018]; tissue modeling with deformable cells [Gardiner *et al.* 2015]; modeling polyethylene pipes subjected to axial ground movement [Meidani *et al.* 2018]; simulation of damage evolution in coatings under uniaxial substrate tension [Ghasemi *et al.* 2020]; study of the mechanics of two-dimensional nanoparticle assemblies [Marchi *et al.* 2019].

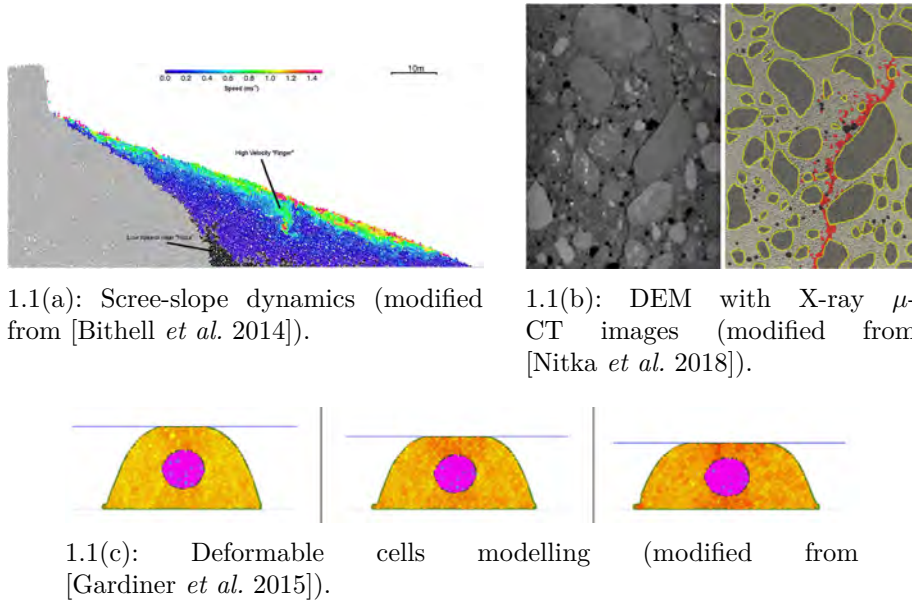


Figure 1.1: DEM application examples.

Regardless of the application, the first step in a DEM simulation is the generation of the particle specimen (a packing is defined by a set of positions and radii within the domain of simulation). The initial arrangement must comply with some requirements, such as the radial distribution, volume density, geometric isotropy, and homogeneity to be useful in realistic simulations.

There are two main approaches to this initial step. The first contemplates the filling of the space with the DEM simulation itself. The disadvantage of this approach lies on the inherent computational complexity. A simulation could take hours or days for a required time step and number of particles that can reach the order of millions.

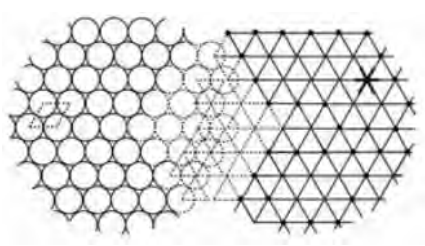
The second approach generates the specimen with geometric procedures. While the static algorithms do not always guarantee completely stable arrangements, they are faster than the dynamic methods in several orders of magnitude, especially when handling complex containers and particles with a high detail such as irregular polygons or polyhedron. Besides, they are better capable of controlling geometrical properties such as radii size distribution and porosity.

Geometric algorithms have been proposed for DEM applications for the last two decades. Currently, there is still ongoing research on the subject due to its importance in engineering applications. The research pays attention to the generation of samples with a high number of particles with lower computational time. In this work, we focused on the generation of disk particles, a common particle shape simplification used in the literature.

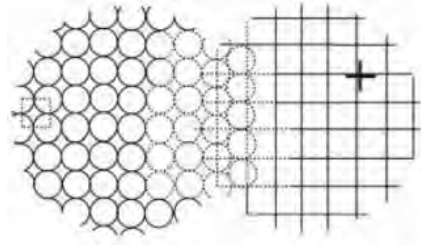
## 1.2

### Two-dimensional random disk packing

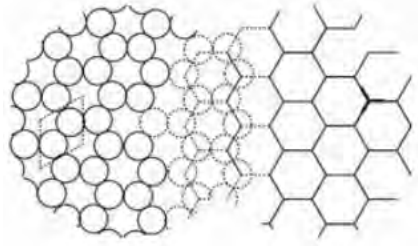
For centuries, mathematicians and physicists have been studying how to efficiently pack spheres or circles to maximize the occupied volume or area. There are three types of disk packing of equal size forming regular tessellations composed of triangles, squares and hexagons. Each one reaching a density of  $\frac{\pi}{\sqrt{12}} \simeq 0.906...$ ,  $\frac{\pi}{4} \simeq 0.785...$  and  $\frac{\pi}{\sqrt{27}} \simeq 0.604...$  respectively [Weaire *et al.* 2008].



1.2(a): Triangle tiling.



1.2(b): Square tiling.



1.2(c): Hexagon tiling.

Figure 1.2: Regular packings.

The maximum density, the fraction of the space covered by the particles, in 2D for circular packings, with a valence of six for each disk, is

$$\frac{\pi}{\sqrt{12}} \simeq 0.9069$$

Joseph Louis Lagrange proved the maximum density for this type of configuration in 1773 but did not consider other non-lattice arrangements. Axel Thue proposed his theorem about the subject in 1890, but his proof was not accepted as it was found to be incomplete. Finally, the Hungarian mathematician László Fejes Tóth provided the first complete proof for the general circle packing problem in 1940 [Chang *et al.* 2010].

Even though some natural elements present hexagonal or face-centered cubic packs [Weaire *et al.* 2008], they are not used for simulation of granular matter.

A packing process is said to be random if it does not favor any specific region of the uncovered part of the domain when adding a new particle [Ebeida *et al.* 2016].

The term “closed packed” implies that the particles are placed within a given domain with the highest possible number of contacts with other particles on average, a property also called as the mean coordination number [Torquato *et al.* 2000].

[Berryman 1983], among other works, predicted a maximum density of  $\rho_{rcp} \simeq 0.82$  for random close packings (rcp). [Atkinson *et al.* 2014] proved the existence of a maximally random jammed packing (MRJ, a concept introduced by [Torquato *et al.* 2000]) with a density of  $\rho \simeq 0.826$  using a sequential linear programming algorithm.

In practice, disordered packings of monodisperse disks are difficult to investigate in 2D. Even for a large number of particles, there is a strong tendency to obtain crystal formations. Therefore, to study random packing in 2D, we use bidisperse, or polydisperse arrangements, a mixture of disks of different radii. Such randomly packed non-uniform mixtures are usually stable against ordering.

In the case of bidisperse packing, simulations have proven a maximum density  $\rho_{rcp} \simeq 0.842$  for a ratio of 1.4 [O’Hern *et al.* 2002, Donev *et al.* 2004, Henkes *et al.* 2007, Meyer *et al.* 2010].

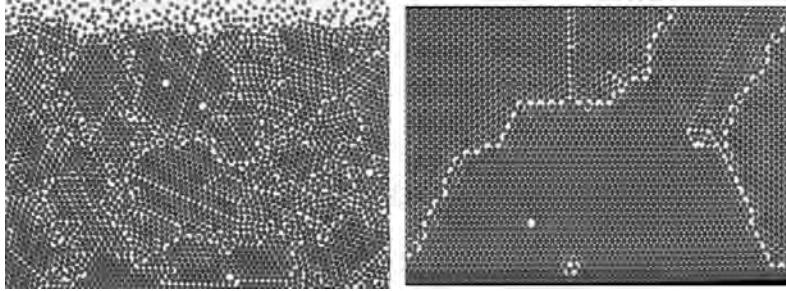


Figure 1.3: In 2D, disks tend to form hexagonal patterns. (Source: [Aste *et al.* 2005])

### 1.3 Goals

The main goal of this work is to propose a new dense particle generation method based on an advancing front method approach to pack disk particles, from now on referred to as particles, following a given radii probability distribution function inside arbitrary domains represented by connected segments. To accomplish this, we identified the following specific objectives:

- Construct a polygonal mesh along with the particle generation. The resulting polygons will allow us to characterize the void space of the whole pack at every iteration.
- Propose new heuristics to determine the next particle's position that not only encourages a compact generation but also seeks to increase contacts between particles.
- Define new strategies to improve the particle positions to increment the packing density during and after the generation using additional information provided by the auxiliary structures. Among the improvements, we contemplate a local optimization, insertions anywhere in the pack, and boundary contact improvement.
- Handle domains with complex non-convex geometries with holes and composed with more than one component.

## 1.4

### Contributions

In a previous work, we proposed a fast 3D sphere package generator called (*Packgen*) [Lozano *et al.* 2016]. To test this idea in practical cases, we collaborate with Marcelo Sampaio de Simone Teixeira and Prof. Deane Roehl in applying *Packgen* as the first step in a DEM simulation. This work resulted in a paper entitled “The influence of sample generation and model resolution on mechanical properties obtained from DEM simulations” that was submitted to the Computers and Geotechnics Journal <sup>1</sup> and is currently under review. This paper's development reinforced the importance of increasing the coordination number and the number of particles in the pack. Given the difficulty of achieving these goals, we decided to start this evolution in the 2D space.

The current work introduces a polygonal mesh where the edges represent particles in contact. This introduction of the mesh brought new metrics, such as the number of sides in the polygons. The use of these metrics yielded a small improvement in the packing density and allowed new strategies for the progression of the front procedure. These strategies significantly improve the geometric properties of the arrangements while maintaining good computational time. They are the main contribution of this thesis.

<sup>1</sup><https://www.journals.elsevier.com/computers-and-geotechnics/>

## 1.5

### Organization

This thesis is structured as outlined below.

In Chapter 2, it is introduced some of the most relevant work related to the disk and sphere packing problem. Related methods are classified into four types of geometric approaches. The chapter also identifies the methods that use topological meshes in the context of particle packing.

In Chapter 3, the topological and spatial structures essential for our method are introduced. First, a grid index for the neighbor and contact search is detailed. Second, it is explained the construction of a polygonal mesh along with the particle generation. The last section describes an adaptive distance field implementation to handle the packing of complex geometries.

In Chapter 4 it is described the core of the advancing front approach. The chapter introduces new metrics to decide the next positions for new particles using the polygonal mesh's information.

In Chapter 5, three strategies to improve the packing are presented. The first strategy describes how to improve the packing inside polygons. The second strategy details how to optimize the packing procedure at the borders of the pack. The last procedure improves the packing near the borders of the container.

Chapter 6 presents the results for the proposed method and the different adopted strategies in different scenarios. The chapter analyzes the geometric properties of the packs employing diverse particle sizes. A comparison to other methods, through reproducible test scenarios, is given. The last section shows the results for complex geometries.

Finally, in Chapter 7, it is given a summary of the problems and solutions presented in this thesis. The chapter concludes with a list of potential future work.

## 2

### Previous Work

Packing algorithms in the literature are mainly classified into two categories: dynamic and constructive.

The dynamic approach uses physical simulations to create the particle assemblies (such as event-driven or a DEM simulation). One group of these simulations uses a multi-layer deposition to progressively fill the domain with particles, letting them interact with each other according to the Newtonian equations of motion [Jian *et al.* 2003, Campello *et al.* 2016]. Another group adopts a rearrangement technique, by a concurrent generation of particle positions within the domain. Then, assigns them an initial velocity and continually increases their sizes until a stable arrangement is reached [Lubachevsky *et al.* 1990].

These algorithms yield satisfactory results regarding producing stable arrangements with higher densities and coordination number, and providing contact force information for every particle. However, their drawbacks are the same ones inherent to any DEM implementation: They are computationally intensive and time-consuming. E.g., [Bagi 1993] proposed a depositional method that can take up to 139 hours to generate 39,000 disk particles. This limitation restricts the number of particles in the arrangement. Moreover, with a dynamic algorithm, it is hard to control packing properties, such as density. Dynamic approaches are impractical in situations where particles are needed in the order of millions, and also when research requires more than one random assembly, with the same porosity and grain size distribution. E.g., [Van der Linden *et al.* 2016] generates 536 packings of spherical particles to study the relationship between fluid flow and the internal pore structure.

Constructive approaches are based solely on geometrical computations, and for this reason, they are also known as “geometric methods”. Therefore, they are more efficient in terms of execution time (in several orders of magnitude) and yield better control package properties, such as the particles’ radius size distribution or the final porosity within the domain. They also can create arrangements without overlaps, resulting in stress-free packs. On the other hand, they are not always in equilibrium under gravitational forces because of the pure geometric criteria.



The majority of the literature's geometric methods can be classified into four main subcategories: geometric separation, Apollonian based, mesh-based, and advancing front.

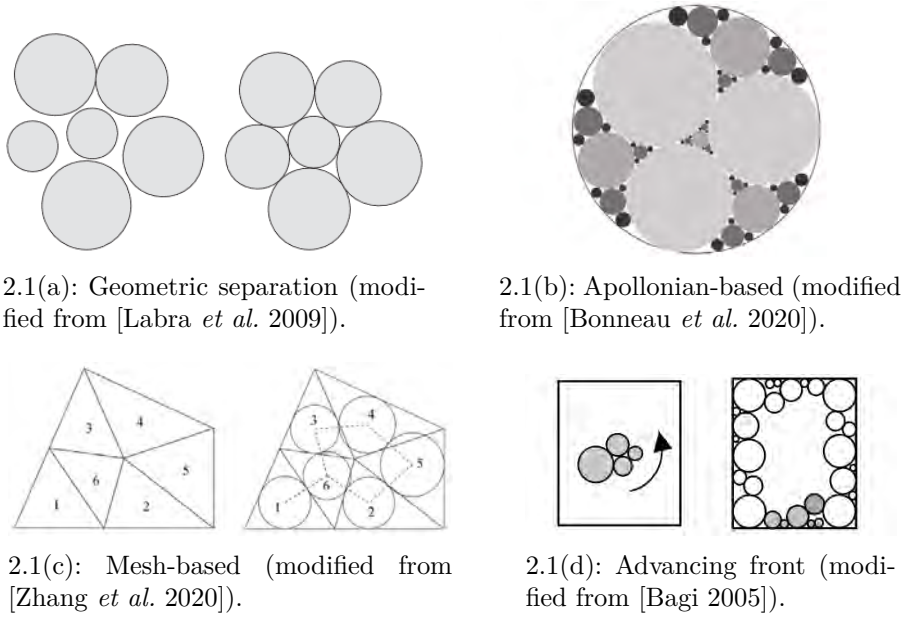


Figure 2.1: Categories of geometric methods.

## 2.1

### Geometric separation methods

Methods in this category iteratively modify the position of a set of particles. [Labra *et al.* 2009] presents an optimization algorithm to improve particle assemblies in order to obtain dense configurations. Labra uses the Stienen model [Stoyan98], a fast geometrical algorithm that generates a non-dense set of stationary Poisson points to then proceed with the void space minimization obtaining a dense pack inside a finite element mesh. The package optimization considers the minimization of the distances between particles and the distance between particles and the boundaries.

[Lopes *et al.* 2020] proposed an algorithm in 2D to fill arbitrary shapes with given particle size distributions and prescribed filling ratio, ensuring homogeneity for the spatial distribution. Their work reproduces granular models' characteristics, like soils, and uses the experimental data used by [Frery *et al.* 2012] to validate the results. The method starts with the random insertion of particles in the domain and then, with a geometric separation mechanism, reduces the overlapping iteratively by displacing the disks. To further reduce the overlaps, the algorithm also relocates and removes some particles. These particles

are re-inserted to low filling rate areas. To avoid the filling rate error caused by the permanent removal of particles, they propose using a response surface.

## 2.2

### Apollonian based methods

The Apollonian packing is a fractal generation method that initially places three tangent disks, each tangent two the other two. The method repeatedly places disks in the largest possible circular hole from the remaining empty space [Borkovec *et al.* 1994].

[Weller *et al.* 2010] introduced the “Protosphere” algorithm based in the Apollonian sphere packing to fill arbitrary objects with non-overlapping spheres. The method fills models with spheres as big as possible that can be placed in the empty regions. For a random point, named as a “prototype”, the method defines the final position and radius of the point based on an optimization process that pushes it away for the closest particle or boundary. A particle displacement reduction guarantees the convergence of the process. The method defines a grid of cells to accelerate the particle insertion, and for each cell creates a “prototype”. Then, in parallel, it let the points converge inside their respective cells. The algorithm inserts the particle with the biggest radius. The grid of prototypes is updated to continue again with a new convergence step. The algorithm ends as soon as the pack reaches a given maximum number of sphere particles.

[Teuber *et al.* 2013] extends Weller’s method with an adaptive grid approach for the prototype cells to improve performance.

Recently, [Bonneau *et al.* 2020] proposed an algorithm for generating mechanically sound sphere packings in geological models that solve the drawback of the “Protosphere” algorithm for uniform distributions between  $[r_{min} - r_{max}]$ . Their optimization process differs by minimizing the distance to other particles instead of maximizing it.

The main disadvantage of these methods is that they follow a greedy insertion and cannot follow a given arbitrary radii distribution function.

## 2.3

### Mesh based methods

The second group of constructive algorithms focuses on the generation of particles inside non-trivial domain boundaries defined by triangular or tetrahedral meshes. These methods exploit the internal structure (vertices, edges, and faces) to place new particles.

[Cui *et al.* 2003] developed a relatively fast and simple method to generate

packs of spheres. The method starts by creating random points inside the domain. Then, using these points, triangles (or tetrahedrons in 3D) are constructed using a Delaunay triangulation (Qhull algorithm [Barber *et al.* 1996]). Then, a single circle (sphere) is inserted in the position of the “inscribed circle” (“inscribed sphere”) of each triangle (tetrahedron). Finally, new circles or spheres are placed on the vertices of the respective elements without collisions. As a result, the pack quality depends on the tetrahedralization algorithm; regular tetrahedra achieve higher particle contacts, and higher densities are obtained using more initial points. A derivation of this work is the algorithm introduced by [Jerier *et al.* 2009, Jerier *et al.* 2010] to create polydisperse sphere packs within tetrahedral meshes. They define a geometric procedure to place particles on nodes and in the middle of the edges; next, adds new particles in contact with four near particles. In the first version [Jerier *et al.* 2009], an inversion function is used, and in the second [Jerier *et al.* 2010] version, an equation system is used. An inconvenience related to the tetrahedron approach is the size of the pack radii. New tetrahedrons or triangles need to be generated to obtain a new pack with a different radius range. These works experiment with meshes that models open-pit mines with benches (1,000,000 spheres) and meshes obtained from portions of tomographies of porous ceramic.

Recently, [Zhang *et al.* 2020] presented two geometric algorithms modifying the Cui’s method to improve the coordination number of disk particle arrangements within triangular meshes. The first algorithm places one disk per triangle. However, unlike Cui’s method, it does not always compute the triangle in-circle. It looks to generate a circle in contact with disks in the adjacent triangles if they exist. The second algorithm has the same logic but inserting three disks per triangle. The system of equations is solved with the Newton-Raphson method. The method also performs a vertex filling procedure to place particles at the triangles’ vertices using the biggest particle in contact with three disks in the neighborhood around the vertex.

A disadvantage of this category of packing algorithms is the dependence of a mesh generator to pack a free form domain. Besides, the particle radii distribution depends on the quality of the given mesh.

## 2.4

### Advancing front methods

The advancing front technique was initially employed in mesh generation procedures [Lo 1985]. For the particle generation problem, the strategy iteratively places new particles in contact with previously inserted particles. They compute the new particles’ positions using the information of local regions of

the recently formed pack. The fronts are updated, either by removing or adding elements until there are no more fronts to process.

### 2.4.1

#### Outward expansion

In one of the first constructive algorithms, [Feng *et al.* 2003] presents an advancing front approach to obtain disks' arrangements. The proposed “closed form” starts with the placement of three particles in the center of the domain. Then, for each front (a segment connecting two particles), a new disk is placed in contact with two disks. Newly formed segments connecting the new particle are added to the front list. The algorithm ends when there are no more segments to process. [Feng *et al.* 2003] also proposed a second method, the “open form”. It begins the packing from the bottom to the top of the container and mainly differs from the previous method in that the front of segments is not closely connected. Later, [Zsaki 2009a, Zsaki 2009b] defines a similar method to create disks layer by layer from the bottom. It even explores the parallel generation of packs by subdividing the domain into subdomains.

Feng's “closed form” approach that starts the packing inside the domain to the frontiers is later known as the “outward packing method” and variations are widely used in the literature.

[Liu *et al.* 2012] proposed the “seed expansion method”. Using a local Delaunay tessellation (Qhull algorithm [Barber *et al.* 1996]) and a distance function with boolean operations on simple geometries, the package assigns a level number to the particles generated in each layer and use a given number of layers as the fronts. With this criterion, the method is capable of inserting particles inside the pack restricted to the last generated layers.

[Lozano *et al.* 2016] developed an algorithm (*Packgen*<sup>1</sup>) to create random particle assemblies of a given sphere size distribution function within volumes of arbitrary shape. In every iteration, the algorithm uses the sphere at the top of the queue of front spheres to compute new positions using a radius size retrieved from the radii generator. This queue is updated with the insertion of new particles and the removal of front spheres that cannot generate more candidate points. The process continues until the queue of fronts is empty. The computation of the new position uses the halo concept of [Ferreira 2009].

A disadvantage commonly described by the outward advancing front methods is the poor placement of particles near the boundaries. Recently, [Dong *et al.* 2020] extended the work of Feng by developing a strategy to solve the gap between disk particles close to the frontiers of the container by defin-

<sup>1</sup><https://git.tecgraf.puc-rio.br/elozano/packgen>

ing a critical distance  $D$ . When the distance from a particle's center to the container is within  $D$ , the particle radius is adjusted to create a contact with the boundary.

### 2.4.2

#### Inward expansion

[Bagi 2005], on the other hand, proposed the “inwards packing method”, for rectangular domains to create particles from the domain frontiers to its center. The method starts by placing disks in a continuous chain in contact with the walls. This initial set is known as the “initial front”. For a current front particle, the method looks for a new particle in contact with two previously inserted disks. The neighborhood is the set of particles preceding and following the current front along the front chain. Based on this strategy, [Ferreira 2009] also designed an algorithm to pack disks based on an initial front composed of circles representing rectangular and circular containers. Ferreira introduced the concept of “particle halo” as a geometric support to compute candidate positions for the new disks. Also, the method combines the packing with a genetic algorithm to obtain arrangements with desired porosity.

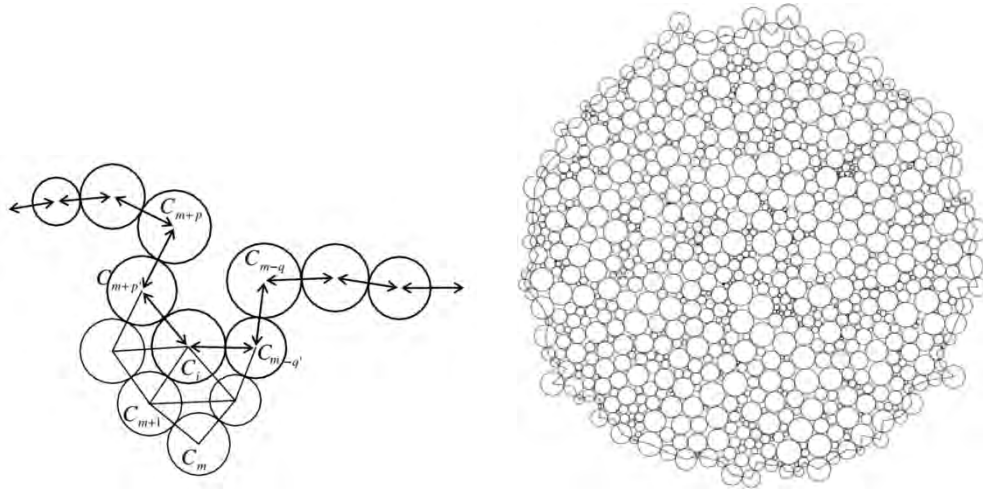
[Benabbou *et al.* 2008, Benabbou *et al.* 2009] presented another variation of inward packing that initially places particles within the container and in contact with the walls. Consequently, the front initially representing the domain is a set of segments in 2D and triangles in 3D. Then, new particles are inserted tangent to the other spheres, and the front list is updated. Benabbou introduced the concept of “front level” to guarantee the convergence of the algorithm. The level of a front is defined as the sum of the levels of its particles. A relevant feature of this algorithm is its ability to pack the particles with a given radii distribution.

Despite being useful for parallelepipeds and simple geometries, most of the above solutions have difficulties addressing complex domains because placing particles on the frontiers is a complicated task. One solution to this problem is discussed by [Liu 2008], where a disk and sphere packing algorithm is proposed to generate high-quality meshes within B-rep domain representations. The method describes how to create an initial front by placing particles on the vertices, edges, and faces (for the container's 3D case). However, Liu's approach accepts particle overlappings.

## 2.5

### Mesh support in disk packing

Mesh and particle generation are related topics given that the advancing front packing method comes from mesh generation methods. Some packing techniques in the literature are mainly designed to create meshes. Other methods use a mesh or graph connectivity as a support structure constructed after or along with the particles' generation.



2.2(a): Front chain and polygon triangulation after  $C_i$  insertion.

2.2(b): Final pack.

Figure 2.2: Wang's advancing front pack (modified from [Wang *et al.* 2007])

[Wang *et al.* 2007] creates particles for unbounded domains. Thus, the algorithm uses a maximum radial distance and a maximum number of particles as a termination condition. As the output, the method creates a triangular mesh for FEM applications. The front concept is composed of a linked list of circles, similar to [Feng *et al.* 2003]. The algorithm adds new particles on top of the front, and as a result, it creates new triangles. If the insertion creates polygons between the front and the new particles, they are subdivided with a triangulation procedure, as we see in Figure 2.2(a).

[Liu 2008] uses an inward advancing front method with a circle packing to create triangle meshes. The front list in this work is composed of the main loop, representing the boundary borders, and several inner loops to model possible holes in the domain. During the generation, the method detects closed regions with few particles that are removed from the front, and the detected polygon is subdivided into triangles. Also, it can merge the main loop with inner loops if there is a circle connecting them. As Figure 2.4 illustrates, the resulting pack can present particles with overlappings.

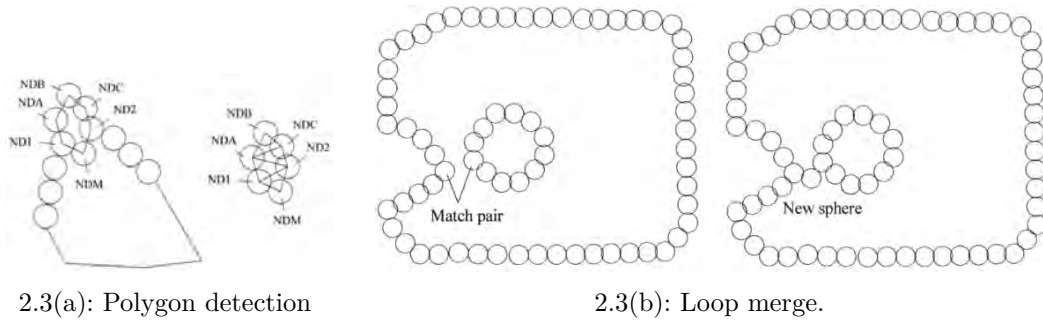


Figure 2.3: Liu's front loops (modified from [Liu 2008])

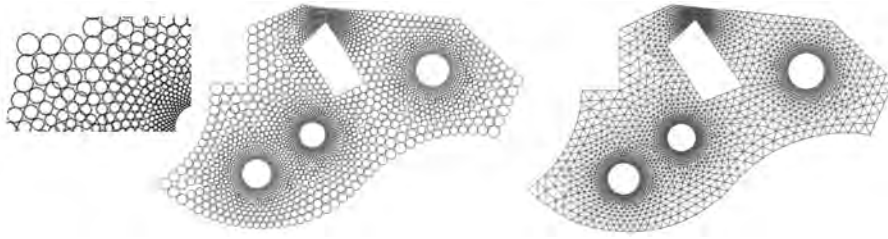
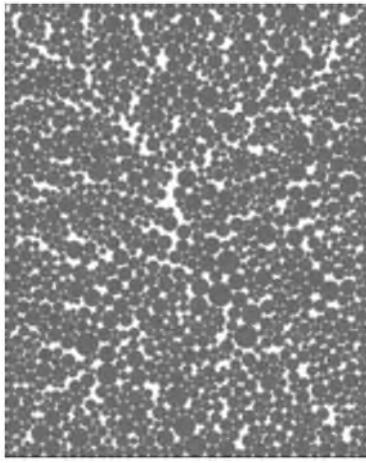


Figure 2.4: Liu's pack and mesh triangulation (modified from [Liu 2008])

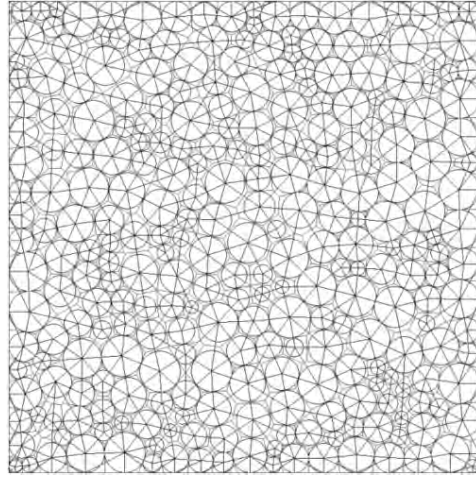
As described, [Wang *et al.* 2007, Liu 2008] used the disk packing as an application to produce triangular meshes and not to create arrangements for particle simulations. [Wang *et al.* 2007] produces packs with low density, and [Liu 2008] allows the particle overlappings to adjust the triangulation to the model boundaries. Both methods tend to form closed loops that need to be triangulated, that in the context of disk packing, it leads to a low coordination number of the particles. Besides, both methods, as the usual advancing front, only produce particles in contact with the front, leaving potential empty spaces generated in the previous regions that are unable to be efficiently filled.

[Benabbou *et al.* 2008, Benabbou *et al.* 2009] acknowledge that their packs have heterogeneous local density, as we can see in Figure 2.5(a). To reduce impacts on physical simulations, the method defines a point relocation algorithm to equally distribute empty areas around each particle. It builds a weighted Delaunay triangulation with the particle centers as point clouds with their radii as weights and considers points at the boundary with zero radii. The points at the boundary are carefully computed so that the cells of the Laguerre diagram (dual of the triangulation) contain the particles entirely. The relocation procedure iteratively moves the particles within the space defined by its nearest neighbors. The process must check that no particle can overlap others.

Despite producing packs with evenly distributed void space in a post-process step, its main problem resides in the packing method itself, allowing the gen-



2.5(a): 2D nanostructure.



2.5(b): Weighted Delaunay triangulation and Laguerre diagram.

Figure 2.5: Benabbou’s packs produces packs with heterogeneous local density (modified from [Benabbou *et al.* 2008])

eration of significant empty areas. A modification of the approach could use the initial Delaunay triangulation to insert additional particles in the biggest void spaces and improve the pack’s density.

[Liu *et al.* 2012] used a Delaunay tessellation on the outer layers of the pack. The number of layers is a parameter given by the user and influences the density and computing time. With more layers, the algorithm reaches more regions inside the pack and, in consequence, increases the density and execution time. The algorithm has a linear complexity based on the number of layers. Figure 2.6 shows a small pack with a local Delaunay tessellation for two-layer of particles. Liu explores the generation of packs using up to eight layers for the Delaunay tessellation. The work also offers a refilling process to place additional particles inside the pack. Potential positions for new disks are identified by performing a range search of neighboring particles around a disk. This process is time consuming and furthermore, it does not respect the desired PDF.

From a computer modeling and simulation perspective, [Specht 2015] packs particles inside circular containers for the polydisperse circle packing problem. The approach constructs a graph representing the connections of the particles. Specht identifies the cycles surrounding void space with the graph. Several jumping strategies, swaps, and shift heuristics are defined to move particles around the container and increase density.

The work defines an embedding depth of two circles

$$\omega_{ij} = d_{ij} - (r_i + r_j), \quad 1 \leq j < i \leq N$$



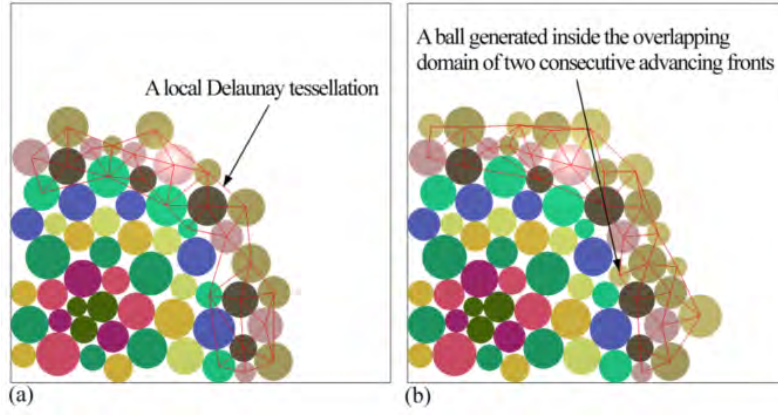


Figure 2.6: Liu's local Delaunay Tessellation (modified from [Liu *et al.* 2012])

And due to a floating point number implementation, it introduces a small numerical tolerance  $\epsilon \approx 10^{-12}$  that leads to three cases:

- for  $\omega_{ij} > \epsilon$  : a strict separation,
- for  $|\omega_{ij}| \leq \epsilon$  : a precise contact,
- for  $\omega_{ij} < -\epsilon$  : an overlap.

Even though the work never mentions the creation of a polygonal mesh, the loop cycle collection can be treated as a mesh. Figure 2.7 presents an example of a small instance and its corresponding contact matrix. Besides the limitation of only considering the packing of circular containers, the needed computational time to achieve packs for a high number of particles makes this approach impractical for our purposes. E.g. Packing 100,000 particles can take up to 24,700 seconds  $\approx$  411 hours.

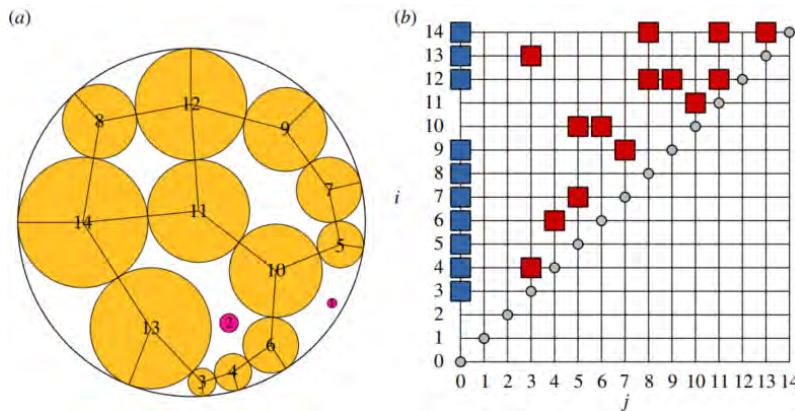


Figure 2.7: Specht's contact matrix (modified from [Specht 2015])

The importance of tools from network science and related mathematical subjects in the study of granular material properties gained importance in the last years. They are powerful approaches to study granular matter and to

enhance understanding of their underlying physics. [Papadopoulos *et al.* 2018] studies granular systems from a complex network theory’s perspective.

In this work, we propose an evolution of an outward advancing front method proposed by the authors in [Lozano *et al.* 2016]. Among the improvements, we aim to construct a polygonal mesh, along with the particle generation, following [Specht 2015] void cycle definition. The mesh allows to establish new heuristics for the calculation of the locus for the incoming particles. To the best of our knowledge, no other pack algorithm creates a global scope contact network during the pack construction.

### 3

## Topological and Spatial Data Structures

Geometric algorithms often require distance or neighborhood information, and to efficiently retrieve this information, spatial and/or topological data structures are required. We use three data structures to support the packing generation. The first is a spatial grid index to handle particle collisions, insertions, and deletions from the pack. The second is a topological polygonal mesh that stores the connections between the particles. Finally, the third is a spatial hierarchical distance field to deal with complex boundary geometries.

### 3.1

#### Particle grid index

In a DEM simulation, it is necessary to build a list of neighboring particles as candidates for contact detection. Without an auxiliary data structure, this search has a quadratic complexity on the number of particles  $O(n^2)$ . Efficient algorithms use different strategies to improve this algorithm detection. The neighboring-cell contact scheme [Munjiza *et al.* 1998], the nearest-neighbor contact detection scheme [Panigraphy *et al.* 2008] and sweep and prune [Cohen *et al.* 1995].

Similarly, finding the neighboring particles around a point is a recurring and important task in packing generation algorithms according to the literature [Han *et al.* 2005, Ferreira 2009, Liu *et al.* 2012, Morfa *et al.* 2018, Li *et al.* 2018, Lopes *et al.* 2020].

In this work, neighboring and contact detection are procedures that lie in our algorithm's heart, and they are repeated many times. Here, we use a hash grid [Eitz *et al.* 2007, Miao *et al.* 2014], a uniform grid variation that divides the space into a set of equal size cells. This variation does not explicitly store the grid and only creates cells on-demand at the regions where they are required. This strategy reduces the use of RAM and is especially important for complex and non-convex containers.

As suggested by [Ericson 2004], the optimal cell size is the smallest size that can hold the largest object in the scene in all rotated positions. Since we are only dealing with disk particles, the cell size is twice the maximum radius  $r_{max}$ , given as an input.

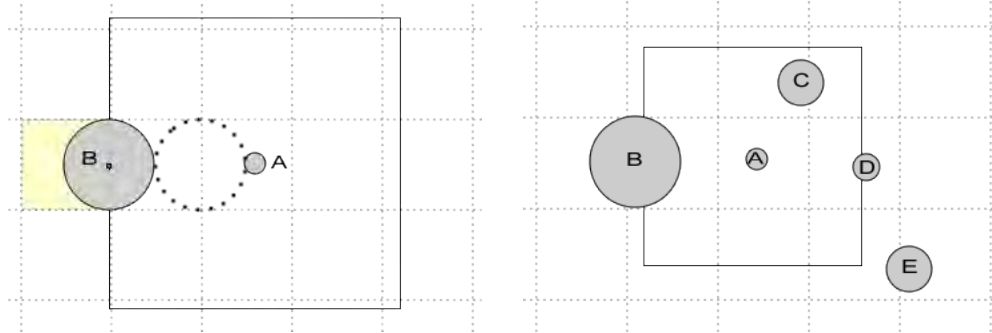
In a grid with the origin at  $\mathbf{o} = (o_x, o_y)$  and with  $n_{px}$ , and  $n_{py}$  cells on the  $x$  and  $y$  axes, respectively, a given point  $\mathbf{p} = (p_x, p_y)$  is assigned to the cell  $(i, j)$ , where:

$$(i, j) = (\lfloor \frac{(p_x - o_x)}{r_{max}} \rfloor, \lfloor \frac{(p_y - o_y)}{r_{max}} \rfloor) \quad (3-1)$$

We map occupied grid cells to a 1D table with the following function:

$$c = (i + j * n_{px}) \quad (3-2)$$

There are two possible strategies to store disks in the spatial grid. The first is to store a reference to a disk in all cells it intercepts. The second strategy consists of storing a disk reference only in the cell that contains its center. With the first strategy, to build a list of neighbors, one must test all particles referenced in all cells the current particle intersects. This procedure yields repeated references to the same particle. The repetition of references increases the cost of this core procedure and thus is undesirable for our algorithm. So, for performance reasons, we use the second strategy, and a particle reference is stored only to the cell that contains its center.



3.1(a): The extreme case that places a particle between two other particles A and B.

3.1(b): Neighbor particles: B, C, and D. Not neighbor: E.

Figure 3.1: Neighborhood box logic around the current front.

To select the particles in the neighborhood of the “current front” disk,  $r_{curr}$ , we create a square box, centered at the  $r_{curr}$  position with the size,  $\delta_{box}$ , given by:

$$\delta_{box} = 2(r_{curr} + 2r_{new} + r_{max}) \quad (3-3)$$

All disks in the grid cells that intersect this square are neighbor candidates to  $r_{curr}$ .

The formula adds one unit of  $r_{curr}$  to cover the area of the front, plus two units of  $r_{new}$  (the radius of the new particle) to contemplate the extreme case where there is one contact point with a neighbor, and an additional unit

of the  $r_{max}$  to cover the bins of the extreme case. Figure 3.1(a) shows the front  $A$  and a search box around it seeking to insert a particle with the maximum size  $r_{max}$ . The box is large enough to detect the particle  $B$  that, along with the particle  $A$ , generates a contact point for the new particle. In Figure 3.1(b), we see a small particle  $A$  in the middle with its search box. Only particles  $B$ ,  $C$  and  $D$  are considered neighbors because they belong to grid cells colliding with the search box. Particle  $E$  belongs to a bin that does not intersect the search box; therefore, it is not considered a neighbor.

### 3.2

#### Polygonal mesh

We include a mesh structure, with irregular polygons, build alongside the pack generation. The mesh reflects the network connectivity among the particles and assists the main algorithm in identifying the container's wasted space.

A vertex in the mesh represents a particle in the pack. The edges represent the contact between a pair of particles. The polygons are associated to a void space formed by a loop of contacts between particles.

**Definition 3.1** A vertex  $v_i$  corresponds to the center of the particle  $d_i$  in the pack arrangement  $PA$ .

**Definition 3.2** An edge  $e_{ij}$  is a connection between two vertices  $v_i$  and  $v_j$  whose respective particles  $d_i$  and  $d_j$  are in contact.

**Definition 3.3** Let  $po$  be a polygon of  $n$  sides composed of a circular list of edges,  $e_{ab}, e_{bc}, \dots, e_{na}$ . A polygon has no repeated vertices or edges.

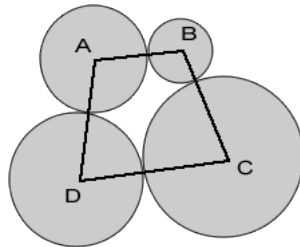


Figure 3.2: Four particles, connected through four edges, define an irregular quad.

**Definition 3.4** An edge is a boundary edge if it is shared by only one polygon. It is an internal edge if two polygons share it.

**Definition 3.5** *A polygon is an internal polygon if all its edges are internal. It is a boundary polygon if at least one of its edges is boundary.*

**Definition 3.6** *A vertex is an internal vertex if all its incident edges are internal. Otherwise, it is a boundary vertex.*

We identify a polygon linking only boundary vertices at the frontier of the pack (connecting boundary edges); we refer to this polygon as the “outer loop”. Most of the new particles will be added to the “outer loop”. Most outward advancing front algorithms use equivalent structures as the front list.

**Definition 3.7** *An outer loop is a two-connected set of edges linking boundary edges.*

An issue identified during the generation of particle arrangements PA, with the approach in [Lozano *et al.* 2016], is the creation of a high number of large polygons and polygons with high porosity, that waste space, reducing the density and, in some cases, the mean coordination number of the pack. Consequently, the arrangement allows a higher mobility of the particles under physical simulations.

With the mesh construction, it is now possible to detect weak configurations and make local corrections immediately. Furthermore, with the new geometrical information available, we propose new heuristics to define the new particles’ positions.

### 3.2.1

#### Mesh construction

The creation of the polygonal mesh data structure, PM, starts after the third disk’s insertion into the pack. From that point on, every particle is initially in contact with two other particles, known as the parents. Thus, we start with constructing a triangle, and then, with every disk inclusion, we build new polygons and add them to the mesh.

We keep a list of vertices per polygon and another map of polygons per vertex. Each vertex  $\mathbf{v}$  and polygon  $\mathbf{po}$  have a unique id represented by an integer number. All vertices share the id of their corresponding particle.

$$\mathbf{v}_{id} \rightarrow \mathbf{po}_{i_1}, \mathbf{po}_{i_2}, \dots, \mathbf{po}_{i_{npi}}$$

$$\mathbf{po}_{id} \rightarrow \mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_{nvi}}$$

where  $npi$  is the number of polygons that share  $\mathbf{v}_{id}$  and  $nvi$  is the number of vertices of polygon  $\mathbf{po}_{id}$ . Here, by convention, the sequence of vertices in the polygons follows a clockwise order.

We handle the “outer loop” as a directed graph where each association reflects a frontier edge. Note that the first and last vertices are connected:

$$\mathbf{v}_a^* \rightarrow \mathbf{v}_b^*, \quad \mathbf{v}_b^* \rightarrow \mathbf{v}_c^*, \quad \dots, \quad \mathbf{v}_x^* \rightarrow \mathbf{v}_a^*$$

As mentioned before, initially, the outer loop is a triangle. Before and after the insertion of new particles, every node must be strictly connected by two other nodes in the graph (2-connected). During an insertion, this condition is temporarily broken to find the new polygon’s position. Following the clockwise convention established for the inner polygons, the sequence of vertices in the “outer loop” follows a counterclockwise order.

The following sections describe in detail the two possible scenarios handling the polygon mesh.

### 3.2.1.1

#### Finding polygons on graphs

To find polygons using the network information of the arrangement, we use the “left-hand rule” criterion. Given an initial edge  $\mathbf{e}_{ab}$ , the next edge in the path is chosen by comparing the angles between  $\mathbf{e}_{ba}$  and all the edges  $\mathbf{e}_{bc_i}$  where  $c_i$  is a vertex connected to  $b$ , but different than  $a$ . To obtain clockwise polygons, we select the edge with the highest angle. The search adds edges to the path until it returns to the initial vertex of the first edge.

For example, in Figure 3.3 starting from the edge  $\mathbf{e}_{AB}$ , the method selects the edge  $\mathbf{e}_{BC}$  as the new edge in the path because it has the highest angle among all other directions from the vertex B. In further iterations the path walks over the edges  $\mathbf{e}_{CD}$  and  $\mathbf{e}_{DA}$  applying the same criterion. The path returns to the vertex A, closing a quad.

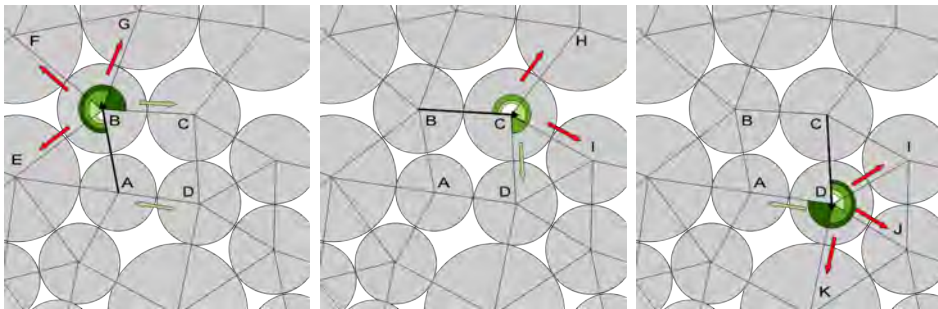


Figure 3.3: Left-hand rule criterion. Starting a path from  $A \rightarrow B$ , the criterion closes the polygon ABCD.

## 3.2.1.2

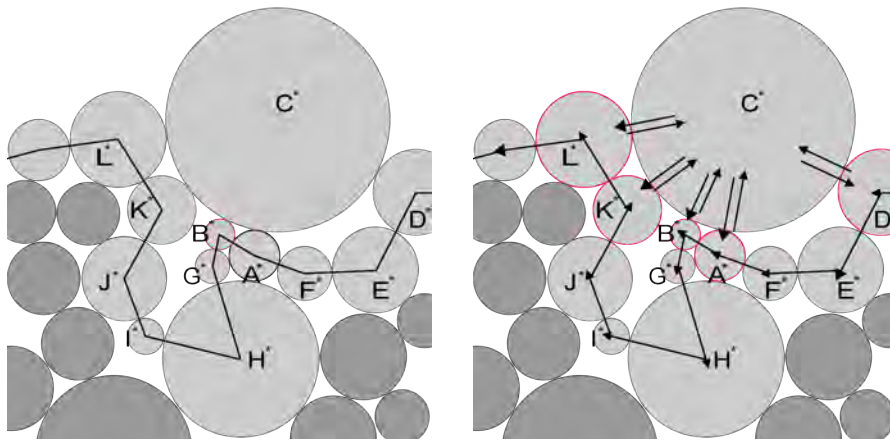
**Insertion on the outer loop**

New disks are inserted in contact to the parent particles, but they may also be in contact to other neighbors. At each insertion, we must perform a search in the surrounding area to detect other possible contacts. Figure 3.4(a) depicts an example where a new particle  $C$  is placed in contact with the parents  $A$  and  $B$ . A search in the insertion procedure discovers three other contacts with the disks  $D$ ,  $K$ , and  $L$ .

When we insert a new particle in an outer loop, and it touches  $n$  border particles, the algorithm must create  $(n - 1)$  polygons and repair the outer loop graph.

Our solution expands the outer loop graph inserting two new edges between the new particle and every particle it touches, i.e., an edge goes from the new particle to the one it touches in the outer loop, and another edge goes in the opposite direction. These new edges temporarily break the 2-connectivity condition of the outer loop.

The algorithm proceeds to find a polygon for each edge that leaves the new particle, and with the “left-hand rule”, traverses the outer loop edges until it returns to the new particle. The edges of the new polygons are removed from the outer loop. In the end, the outer loop’s 2-connectivity is restored. In Figure 3.4(b), ten edges are included in the graph, two for each of the five particles in contact  $D^*$ ,  $A^*$ ,  $B^*$ ,  $K^*$  and  $L^*$ .



3.4(a): Front  $A$  and neighbor  $B$  yield the particle  $C$ .

3.4(b): Every contact adds two edges into the “outer loop”.

Figure 3.4: Insertion on the outer loop.

Figure 3.5 shows that the parallel walk on the extended graph closes two paths in three iterations.



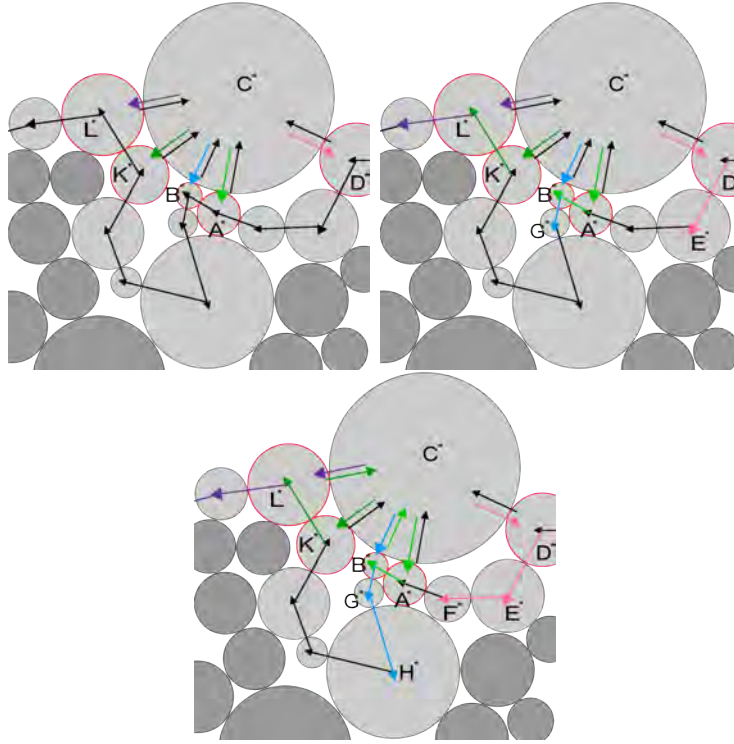
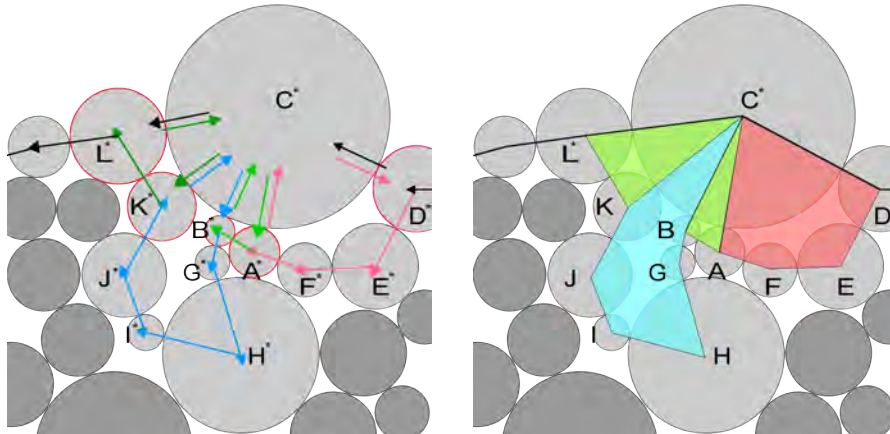


Figure 3.5: Sequence of polygon search for three iterations.



3.6(a): Valid paths to polygon construction.

3.6(b): Addition of polygons and update of the “outer loop”.

Figure 3.6: Insertion in the outer loop (Continuation).

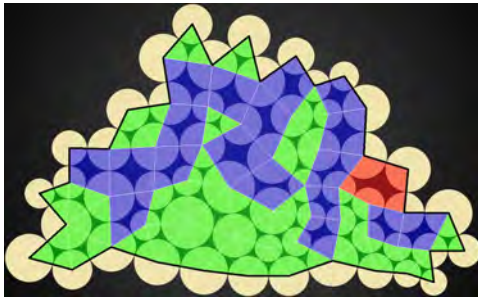
Figure 3.6(a) illustrates the identification of the four valid paths that leads to the construction of the respective polygons after seven iterations. The new polygons receive new indices, and each polygon registers the indices of the particles involved (mapping of vertices per polygon and polygons per vertex).

Then, the edges connecting the polygons are removed from the directed graph. The removal restores the 2-connected property of the “outer loop”, a single path connecting all the particles at the border of the mesh. Note that, in Figure 3.6(b), the edges  $D^* \rightarrow C^*$  and  $C^* \rightarrow L^*$ , that were not considered

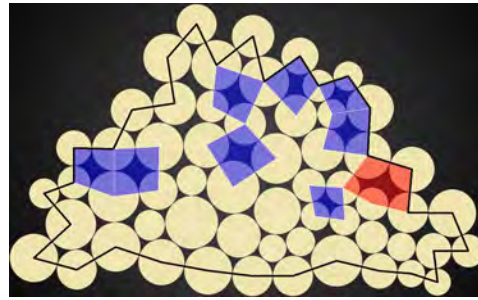
for the new polygons, are now part of the “outer loop”.

Another important procedure in the stage is the estimation of the polygon’s porosity or void ratio. This information is essential for a local optimization that reorders small portions of the arrangement. When a polygon is created, we compute its area ( $A_p$ ) and the sum of the solid portions of disks inside it ( $S_p$ ).

Figure 3.7(a) shows a small set of polygons for a disk arrangement in the background. The mesh is composed of triangles ( $\triangle$ ), quads ( $\diamond$ ), and pentagons ( $\circ$ ). The thick black line represents the outer loop. In Figure 3.7(b), we only shade the polygons with a porosity ( $1 - S_p/A_s$ ) higher than 19.3%. Note that none of the triangles is shaded.



3.7(a): Polygons on top of the pack.



3.7(b): Polygons with porosity higher than 19.3%.

Figure 3.7: A polygonal mesh created along with a small pack. It supports the identification of wasted space in certain areas. The “outer loop” is presented as a thick black line.

### 3.2.1.3

#### Insertion inside a polygon

Heterogeneous arrangements, with a high ratio between the maximum  $r_{max}$  and the minimum  $r_{min}$  size, are prone to generate void spaces large enough to place small particles inside. A packing method needs to consider those potential places to add more particles to increase the density and contacts in the arrangement.

As in the previous section, we detect other possible contacts, besides the parent particles, among the particles at the polygon’s vertices. Then, we build a directed graph with the edges of the polygon. Next, the graph is expanded with two edges between the new particle and the disks in contact. To split the polygon, here we also traverse the graph using the “left-hand rule” for each edge that leaves the new particle, until it creates a polygon. Insertions inside polygons create  $n$  polygons for  $n$  contacts, not  $n - 1$  as in the border case. So, here, every contact creates a new polygon.

The original polygon,  $po$ , and the vertex indices' references to  $po$  must be removed from the respective data structures. The new PM data structure must include the new polygons  $po_i$ , and all affected vertices must be updated.

Figure 3.8 and Figure 3.9 illustrate an inner insertion where the procedure splits an irregular pentagon into two polygons, a quad, and a pentagon.

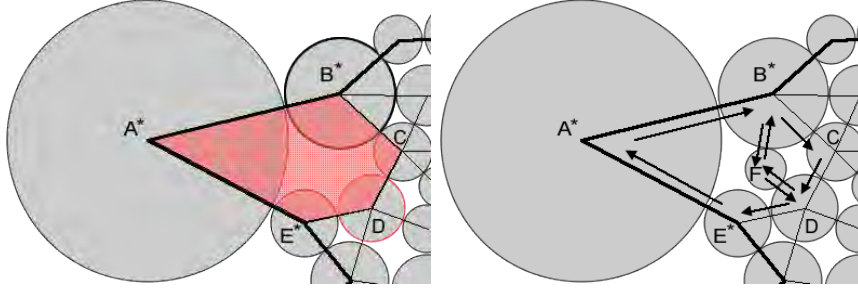


Figure 3.8: Insertion in polygons. a) The particle F is placed inside the pentagon ABCDE. b) A temporal graph with the new contacts and the polygon.

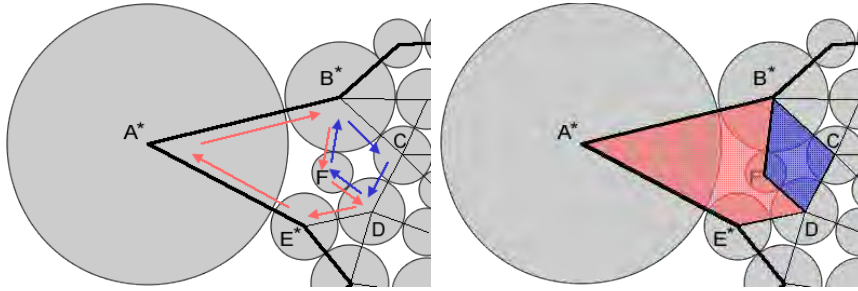


Figure 3.9: Insertion in polygons. a) The left-hand rule finds two paths. b) The insertion yields two new polygons FDEAB and FBCD.

### 3.2.2

#### Neighborhood search for particles inserted in polygons

Our insertion algorithm uses a “current front” particle and another neighbor particle as parents to the new disk. In the “outer loop” insertion, we use a spatial grid structure to get this neighbor. For the internal particles, our strategy is to use the **VertexStar**( $v$ ) procedure instead. We believe this choice is better than the range search because it avoids unfeasible particles, as illustrated in Figure 3.10. In this figure,  $A$  is the “current front” particle. Particles  $F$  and  $G$  are not feasible parents with  $A$ , but they would be candidates for neighbors if the search were based on the spatial grid index. It is impossible to generate new valid positions with one of them and  $A$ .

Figure 3.10 also illustrates how a “current front” particle can end up in the mesh’s interior.  $A$  was part of the “outer loop” before the insertion of the particle  $H$ .

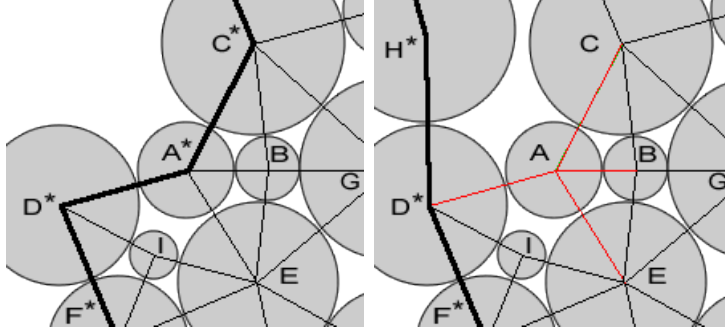


Figure 3.10: Particles  $A$  and  $C$  are boundary vertices until the insertion of particle  $H$ . The  $\text{VertexStar}(A)$  procedure returns the particles  $B$ ,  $C$ ,  $D$  and  $E$ .

### 3.3

#### Boundary detection

All packing methods need to efficiently determine whether a particle is entirely inside a given static domain. This work considers two types of boundary representations.

For simple known geometries it is straightforward to define if a disk is in collision or contained in the domain. We refer to Appendix B for the details of the implementation for circles and rectangles.

For the case of complex geometries, we consider non-convex closed domains composed of connected line segments.

We can compute the closest distance from a query point  $x$  to the boundary's geometry  $\partial O$  identifying the closest primitive:

$$CP(x, O) = o_{cp} \in O \mid \text{dist}(x, o_{cp}) = \min_{\forall o \in O} (\text{dist}(x, o))$$

where  $\text{dist}(x, o)$  is the distance between the point  $x$  and the primitive  $o$ , which in our case are line segments.

**Definition 3.8** *Let  $x$  be a point in space. The exact distance of  $x$ ,  $ED(x)$ , to  $\partial O$  is  $\text{dist}(x, CP(x, O))$ , the distance to the closest segment.*

The exact distance computation for a point has a linear complexity  $O(n)$  on the number of primitives  $n$  in  $O$ . A brute-force approach for the packing generation is impractical given the number of required queries.

Since the packing algorithm needs an efficient query with a constant time to determine if a particle is inside the object  $O$ , we use a distance field, a scalar field that specifies the minimum distance to a shape.

Considering that we are just interested in distances inside the container and that the distances will be compared with the particle radii, we use the convention that the sign is positive for inside points.

$$\text{Sign}(x) = \begin{cases} 1 & \text{if } x \in O \\ -1 & \text{otherwise} \end{cases}$$

Furthermore, for those points beyond the frontiers of the region of interest, we just assign the invalid negative distance of `-MAX_DISTANCE`.

We implement an adaptive distance field, ADF, based on a top-bottom subdivision strategy [Friskén *et al.* 2000] with some variations for our context. Specifically, we need an accurate distance for a region equal to the maximum radius in the pack  $r_{max}$  inside  $\partial O$ . Beyond this offset, it is acceptable to obtain inaccurate distances.

To recompute the approximate distance of an arbitrary point  $x$  in space to  $\partial O$ , we use a bilinear interpolation with the corner values of the quadtree cell enclosing  $x$ .

The tree construction needs to find the closest primitive in  $O$  for specific points to determine their exact distance. To speed up these computations, we use a dynamic grid partition of the primitives of  $O$  proposed by [Scrimieri *et al.* 2014]. After the signed distance tree construction, the memory allocated by the grid partitioning can be released.

### 3.3.1

#### Tree division strategy

A *cell* is defined as an axis-aligned rectangular area. Hereafter, the quadtree cells are going to be noted as cells. We identify nine points of interest within a cell: its center, four middle edges, and four corners.

A first criterion classifies the cells based on their closeness to  $\partial O$ .

**Definition 3.9** *Let  $C$  be a cell and  $C.c$  its center.  $C$  is a boundary cell if the exact distance of the center,  $ED(C.c)$ , is smaller than the half diagonal of  $C$ . These cells need to be subdivided up to the maximum level  $L_{max}$  if they cannot approximate the distances of their middle edges and center.*

Non-boundary cells are also classified based on the signs of their corner distances. They are either entirely inside or outside  $\partial O$ .

**Definition 3.10**  *$C$  is an inner cell or an outer cell if it is non-boundary and the exact distances of its four corners have positive or negative distances, respectively.*

Only if the four corners have negative values (outside the region of interest), we do not subdivide the cell.

**Rule 1** *Subdivision rule: If  $C$  is an outer cell. It should not be subdivided because it encloses a region that is not relevant for the packing purpose.*

Similarly, for an inner cell, if none of its four corners have an exact distance smaller than the maximum radius  $r_{max}$ , we do not subdivide the cell. The interpolation distance for a point within the cell will be inaccurate. However, it will be higher than  $r_{max}$ , indicating that a particle with a radius in the range of  $[r_{min} - r_{max}]$  could be placed without boundary collision. In consequence, particles with the center inside these cells do not need the interpolation computation.

**Rule 2** *Subdivision rule: If  $C$  is an inner cell and none of its corners have an exact distance smaller than the maximum radius  $r_{max}$ , it should not be subdivided because it is far from the boundary.*

Finally, we perform the subdivision on differences like [Friskien *et al.* 2000]. We compute the exact distance at five positions: the four middle edges and the center. We then interpolate distances for these points with a bilinear interpolation using the exact distances at the four corners. If, for some of these interpolations, the error is higher than the threshold  $\epsilon = r_{min} \times 10^{-10}$ , the cell is marked for division.

**Rule 3** *Subdivision rule: A cell  $C$  should be subdivided if some of its sample points cannot be correctly interpolated. If  $\epsilon > threshold$*

To test if a point is inside or outside the boundary, it suffices to check whether its interpolation yields negative or positive values. If a particle centered at  $p$  with radius  $r$  is inside, its distance must satisfy the following:

$$distance(p) \geq r \quad (3-4)$$

Algorithm 1 summarizes the signed distance field subdivision. The recursive approach involves the computation of the exact distance of shared positions by neighboring cells. To avoid recomputing the distance of the points of interest, they are stored in a hash table. The hash values are based on the integer world coordinates (i,j) of positions (x,y) considering a uniform grid with the maximum subdivision level  $L_{max}$ . The hash value is computed as

$$H(i, j) = (i \cdot p_1 \oplus j \cdot p_2) \mod m$$

where  $p_1$ ,  $p_2$  are large prime numbers, in our case 73856093, 19349669, respectively. The value  $m$  is the hash table size.

**Algorithm 1:** ADF construction**Input** : Geometry  $O$ . Maximum tree level  $L_{max}$ . Error threshold  $\epsilon$ **Output:** The signed distance field

---

```

1 Create an empty cellQueue;
2 Create a root cell and add it into the cellQueue;
3 while (cellQueue is not empty) do
4   cell  $\leftarrow$  retrieve(cellQueue);
5   if cell.level ==  $L_{max}$  then
6     continue;
7   Compute cell's center distance:  $ed0 = ED(cell.center)$ ;
8   Compute corner distances:  $ed1, ed2, ed3, ed4$ ;
9   if  $|cell.ed0| > cell.diagonal/2$  then // Not a boundary cell
10    if  $cell.ed1 < 0$  and  $cell.ed2 < 0$  and
11       $cell.ed3 < 0$  and  $cell.ed4 < 0$  then
12      continue; // Rule 1
13    if  $cell.ed1 > r_{max}$  and  $cell.ed2 > r_{max}$  and
14       $cell.ed3 > r_{max}$  and  $cell.ed4 > r_{max}$  then
15      continue; // Rule 2
16    Compute cell's edge distances:  $ed5, ed6, ed7, ed8$ ;
17    Compute approximate distances:  $ad0, ad5, ad6, ad7, ad8$ ;
18    if  $|cell.ed0 - ad0| > \epsilon$  or  $|cell.ed5 - ad5| > \epsilon$  or
19       $|cell.ed6 - ad6| > \epsilon$  or  $|cell.ed7 - ad7| > \epsilon$  or
20       $|cell.ed8 - ad8| > \epsilon$  then // Rule 3
21      for  $i = 1$  to 4 do
22        Create child  $i$  of cell and add it to cellQueue

```

---

**3.3.2****Distance field examples**

This section presents examples of the ADF implementation for two sample models. The first model is a silhouette of the dragon, and the second is a slice of a volumetric micro-computed tomography image (micro-CT).

Table 3.1 summarizes the model properties and input data to create their respective trees for specific  $r_{max}$  values.

It is essential for our method to identify the subset of connected segments surrounding void areas and contained in a solid area. We refer to these loops as the domain holes  $DH$ .

	Dimensions	Segments	Components	Holes	$L_{max}$	$r_{max}$
<b>Dragon</b>	[254, 254]	2,398	1	1	7	0.72
<b>Rock - Solid phase</b>	[450, 450]	21,029	152	43	9	0.40

Table 3.1: Distance field - Models.

Also, given that the “outer loop” of the arrangement must be a single loop,



we use a DFS-based algorithm to determine the connected components of the model. The packing procedure is applied to each component independently. This applies to complex geometries like models extracted from micro-CT images of microstructures.

Figure 3.11 shows the difference of two trees for two  $r_{max}$  values. Note that with a higher value, the ADF generates more leaf cells close to the model boundaries.

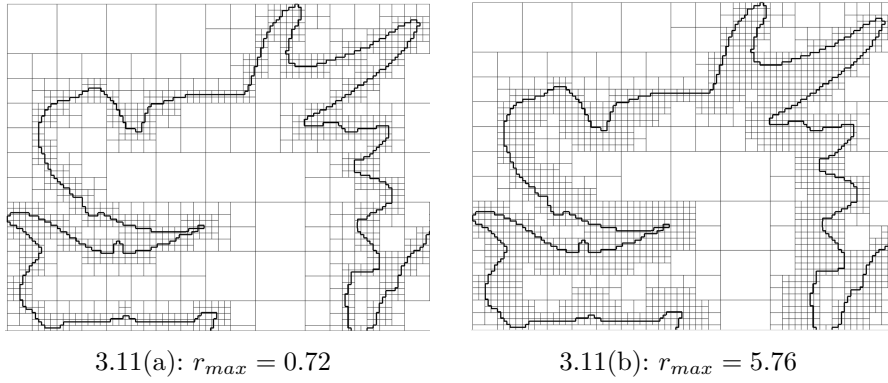


Figure 3.11: Tree subdivision – Dragon head.

We present in Figure 3.12 and Figure 3.14 the distance fields and errors of the sample models. Note that the error regions are located in inner cells far from the boundaries and with distances higher than the  $r_{max}$ . Domain holes  $DH$  are represented in red loops. Figure 3.13 presents a slice of a micro-CT image. It also highlights some domain holes surrounded by red segments. Even though we show the complete ADF of the solid phase in Figure 3.14(a), for the pack generation, we compute the ADF for each connected component.

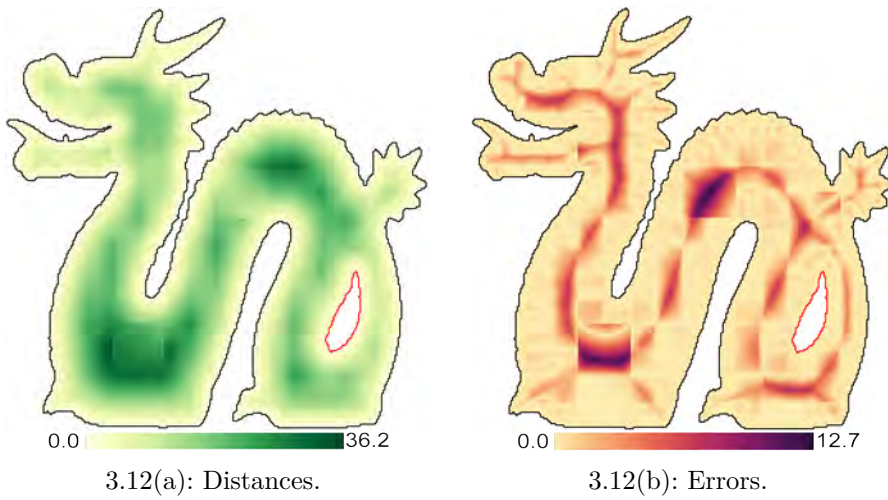


Figure 3.12: Signed distance field – Dragon.



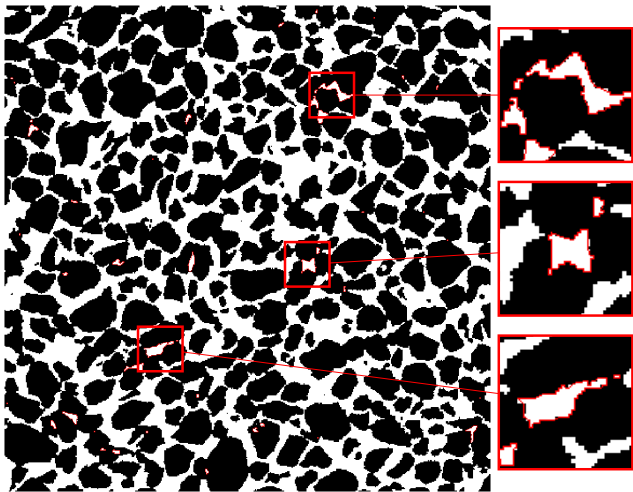
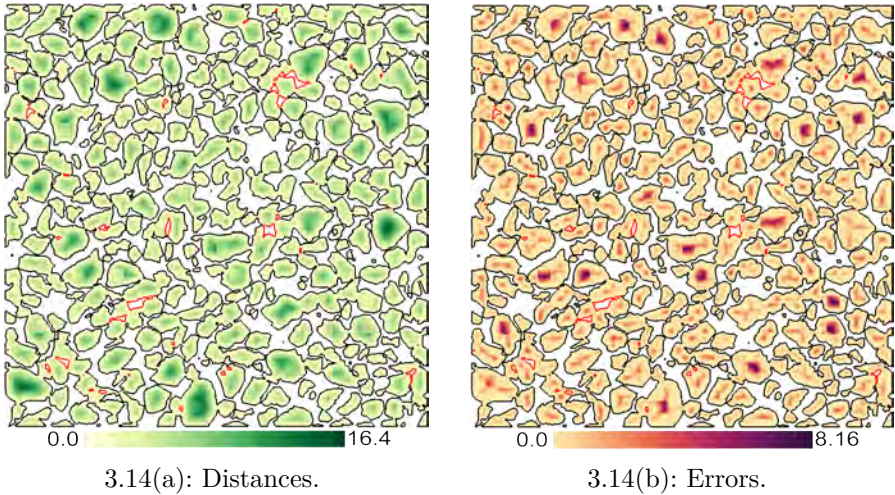


Figure 3.13: Slice of Micro-CT Sand pack LV60C. In black the solid phase and in white the porous space.



3.14(a): Distances.

3.14(b): Errors.

Figure 3.14: Signed distance field – Solid phase in a slice of Micro-CT Sand pack LV60C.

## 4

## Particle generation

### 4.1

#### Outward strategy

The algorithm proposed here is an evolution of an incremental placement of particles with an advancing front strategy. On every step, a particle front generates a new particle placed in contact with two parent particles, the front and another particle in the neighborhood. The algorithm makes better decisions about how to select the new particle positions based on additional information provided by the polygonal mesh. It also introduces two parameters to correct an early removal of particle fronts. All these changes do not hinder the algorithm's capacity to follow a given particle size probability distribution function, PDF, specified by the user.

#### 4.1.1

##### Particle size distribution

Our algorithm produces packs of particles with a set of radius sizes,  $r$ , by varying between  $r_{min}$  and  $r_{max}$  according to a desired PDF. The distribution can be defined in two forms: a known function, such as a Bernoulli or truncated Log-normal distributions, or by a histogram of the radius size.

To achieve the desired PDF our algorithm uses a random generation function to select the particle radius  $r$ . However, most computer languages only include functions to produce uniform random numbers in their standard library, and we use a simple procedure to adapt the PDF. Given a set of discrete radii and their corresponding probability, the algorithm partitions the interval of the uniform variable into sub-intervals of different sizes, in accordance with the probability of the corresponding radius,  $r$ . When the uniform variable value falls in a sub-interval, the corresponding radius is selected. The probability of a given radius to be selected is proportional to the size of its sub-interval. For a large number of radius selections, this procedure yields, for each radius, a probability that is proportional to the size of its sub-interval, thus achieving the desired PDF [Devroye *et al.* 1986].

A large  $r$  may be rejected in our algorithm if it does not fit in the neighborhood

of the current front, e.g., a front close to the container boundaries. This rejection cannot be, however, permanent to avoid the degradation of the resulting PDF. The rejected radii must be used in future insertions when the front moves to a new position. Due to this requirement, our generation scheme contemplates the usage of waiting queues of rejected radii.

#### 4.1.2

##### Generation loop

The current algorithm follows an outward front approach. Particles in the assembly represent the fronts, and they are stored in a queue. The initial front queue is typically composed of two particles tangent to each other and entirely located inside the container. They are also known as seed particles. For practical reasons, as a default position, we choose the center of mass of the container as the starting point. The user may specify any position inside the boundary to start the generation of the assembly. For complex boundaries, where the center of mass is not an interior point, the initial position is a necessary input parameter.

In every loop iteration, the algorithm finds a valid position for a new particle using the information of the neighborhood particles. More precisely, the algorithm selects a particle at the top of the active front to be the “current front” and tries to create a new one touching it and another particle in the surroundings.

To determine the candidate positions for the new particle center,  $\mathbf{c}_{new}$ , the proposed algorithm uses the concept of “halo” [Ferreira 2009]. A circular halo of a particle,  $s$ , is an expanded concentric circle with the radius determined by

$$r_{halo} = r_s + r_{new} \quad (4-1)$$

The algorithm matches the “current front”  $r_{curr}$  with the particles inside the neighborhood box. For every pair, we intersect their halos. To check if two circles  $(\mathbf{c}_a, r_a)$  and  $(\mathbf{c}_b, r_b)$  intersect, we compute the possible overlap by

$$\omega_{ab} = \|\mathbf{c}_b - \mathbf{c}_a\| - (r_a + r_b) \quad (4-2)$$

We use a small overlapping tolerance,  $\epsilon_o = r_{min} \times 10^{-10}$ , that leads to three scenarios:

- for  $\omega_{ab} > \epsilon_o$  : the circles do not overlap and there are no intersections points,
- for  $|\omega_{ab}| < \epsilon_o$  : the circles are touching at one point,
- for  $\omega_{ab} < -\epsilon_o$  : there are two intersection points.

In Figure 4.1, we see three scenarios between two halos.

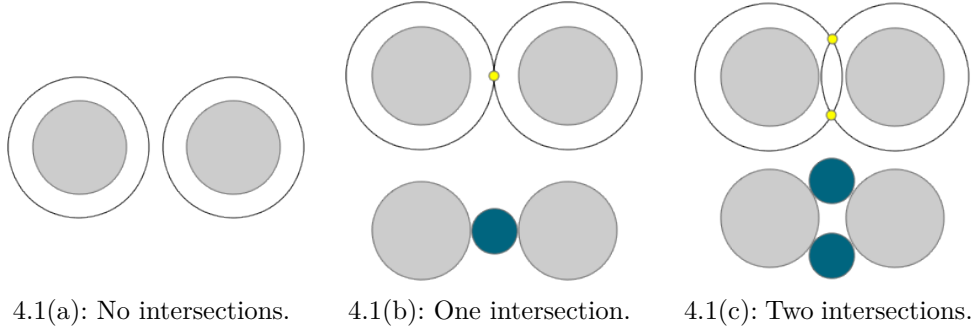


Figure 4.1: Placing a particle in contact with two other particles. The black circles represent the halos. Small points are shown in yellow. a) No valid intersection between the halos. b) A single point of intersection between the halos. c) Two points of intersection between the halos. After the gathering of all the points, the algorithm decides which one is the “winner” for insertion.

Figure 4.2 illustrates a small example for a particle  $A$  in the middle with its search box. Only particles  $B$  and  $C$  are neighbor particles that yield candidate points. The particle  $D$  is collected with the search box, but its halo does not collide  $A$ ’s halo. Particle  $E$  belongs to a bin that does not intersect the search box; therefore, it is not considered as a neighbor.

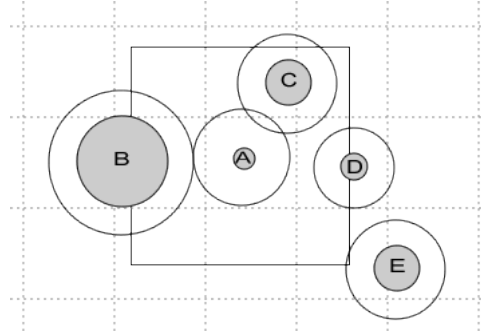


Figure 4.2: Particle B yields a single candidate position. Particle C yields two positions. Particle D and E do not generate points

Using the equations of a pair of halos, with the positions of the particles they belong, and their respective  $r_{halo}$ , it is straightforward to find the intersection points of both circles, known as the “candidate points”.

The center point  $\mathbf{c}_m$ , product of the intersection between two halos  $(\mathbf{c}_a, r_a)$  and  $(\mathbf{c}_b, r_b)$  on a plane with normal  $\vec{n}$ , as illustrated in Figure 4.3, is given by

$$\mathbf{c}_m = (1 - \alpha)\mathbf{c}_a + \alpha\mathbf{c}_b \quad (4-3)$$

with

$$\alpha = \frac{1}{2} \left( 1 - \frac{r_b^2 - r_a^2}{\|\mathbf{c}_a \mathbf{c}_b\|^2} \right) \quad (4-4)$$

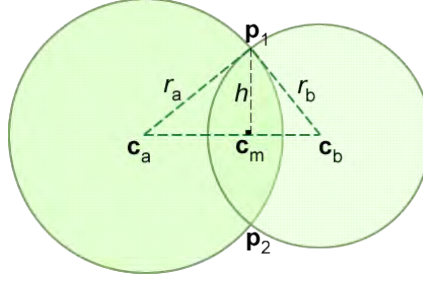


Figure 4.3: Geometric position of the candidate points product of the intersection of both halos.

The height  $h$  is computed using the Pythagorean theorem with any of the right triangles:

$$h = \sqrt{r_a^2 - \|\mathbf{c}_a \mathbf{c}_m\|^2} \quad (4-5)$$

Finally, the intersection points  $\mathbf{p}_1$  and  $\mathbf{p}_2$  are computed by

$$\mathbf{p}_{1,2} = \mathbf{c}_m \pm h \frac{\vec{n} \times (\mathbf{c}_a \mathbf{c}_m)}{\|\vec{n} \times (\mathbf{c}_a \mathbf{c}_m)\|} \quad (4-6)$$

Not all the candidates may be valid positions for the new particle. We must test if the new particle in that position does not collide with any other particle in the neighborhood and is also entirely contained by the container. Once again, we use the grid to detect possible collisions with the surrounding particles, considering the same overlapping threshold used for the halos. We employ the distance field for complex containers to test if the new particle is inside the boundaries.

The number of points left in the candidate's list, after the removal of invalid positions, decides the algorithm's next actions.

If there is one or more valid candidate points, the algorithm proceeds to select the best position. To obtain tight arrangements, the algorithm needs to select the next position based on criteria detailed in Section 4.2.

When the algorithm finds the optimum valid candidate position, it inserts the newly created particle in the pack. Regarding the data structures of the algorithm, this means that there are four insertions: (a) in the assembly list, (b) in the active front queue, (c) in the grid index, and (d) in the polygonal mesh.

A front should not be directly removed from the queue of fronts if it cannot generate valid points for a given radius because this radius can be a large number for specific neighboring conditions, such as boundary proximity. The removal could lead to a premature halt of the generation near that location.

To mitigate this situation, we introduce the parameter, *maxRejections*, that represents the maximum number of rejections for a front.

If the number of valid points is less than 2, and there exists a collision with the boundary, the algorithm spares the front from removal and increases its rejection counter. In the next iteration, the algorithm sorts a new radius for insertion. We only proceed with the front removal from the queue if there are no boundary collisions, or the counter of rejections reaches the maximum allowed value.

Also, when the algorithm does not use a sorted radius for a current front, it is not discarded. To avoid the disposal of radii, that would yield a distribution that does not agree with the prescribed PDF; the algorithm manages two queues: a queue of previously rejected particles, *prevRejectedQueue* (qpr), and a queue of newly rejected particles, *newlyRejectedQueue* (qnr). The first one contains a queue of all the particle sizes rejected in previous loop cycles. When a new radius is needed, the stored radius is tested again. If it is rejected again, it is stored in the *newlyRejectedQueue*. Only when the *prevRejectedQueue* is empty the algorithm uses the probability distribution function PDF to select a new radius. When an insertion is successful, the algorithm removes all values stored in *newlyRejectedQueue* and inserts them in the *prevRejectedQueue*. With this procedure, we seek to preserve the PDF, ensuring persistent testing of rejected particles until its final insertion or packing completion.

As a consequence of the new front rejection logic (which is summarized in Algorithm 3), it is possible that a series of big radii will cause the increment of the rejection counter of a front and will also increase the queue of newly rejected radii. To avoid a big queue, we add another parameter, *maxQueueSize*, that when is reached, all the radii from *newlyRejectedQueue* go to *prevRejectedQueue*, so that the number generator could reuse these radii in the subsequent iterations and respect the desired PDF.

The algorithm continues with the insertion and removal of particles from the active front until it is empty. The pseudo-code in Algorithm 2 recaps the generation loop.

## 4.2 Metrics

Given the set of candidate points, the algorithm must select the best point as the position for the new particle (Line 13 in Algorithm 2). With the support of the polygonal mesh, build on top of the pack, we can obtain additional geometric information of how the mesh will change regarding the insertion of each candidate point in the pack. We identify the following metrics

**Algorithm 2:** Assembly Generation Algorithm

---

```

1 Create an empty frontQueue, assemblyList, gridIndex, prevRejectedQueue
  and newlyRejectedQueue;
2 Create two seed particles in seedPosition and add them in the
  frontQueue, assemblyList and gridIndex;
3 while (frontQueue is not empty) do
4   currentParticle  $\leftarrow$  retrieve(frontQueue);
5   rnew  $\leftarrow$  selectARadius(PDF, rmin, rmax, prevRejectedQueue);
6   box  $\leftarrow$  computeNeighborhoodBox(currentParticle, rnew, rmax);
7   neighboringParticles  $\leftarrow$  gatherParticlesThatIntersect(gridIndex,
    box);
8   candidatePointList  $\leftarrow$  all interceptions of halos of two
    neighboringParticles and the currentParticle;
9   hasBorderCollisions  $\leftarrow$  Exists a position in candidatePointList where
    newParticle intercepts boundaryMesh;
10  remove from candidatePointList all positions where newParticle
    intercepts boundaryMesh or any other neighboringParticles;
11  numberOfValidPoints  $\leftarrow$  numberOfPointsIn(candidatePointList);
12  if (numberOfValidPoints > 0) then
13    bestPoint  $\leftarrow$  bestPointIn(candidatePointList);
14    newParticle  $\leftarrow$  particle(bestPoint, rnew);
15    insert newParticle in frontQueue, assemblyList and gridIndex;
16    remove all particles from newlyRejectedQueue and insert them in
      prevRejectedQueue;
17    MeshUpdate(newParticle, parentsOf(newParticle));
18  else
19    add rnew in newlyRejectedQueue;
20    if (newlyRejectedQueue.size() > maxQueueSize) then
21      remove all particles from newlyRejectedQueue and insert them
        in prevRejectedQueue;
22  HandleFrontRemoval(); // See Algorithm 3

```

---

**Algorithm 3:** HandleFrontRemoval

---

```

1 if (numberOfValidPoints < 2) then
2   if (maxRejections > 0) then
3     if (!hasBorderCollisions) then
4       remove currentParticle from frontQueue;
5     else
6       if (currentParticle.rejections >= maxRejections) then
7         remove currentParticle from frontQueue;
8       else
9         currentParticle.rejections++;
10  else
11    remove currentParticle from frontQueue;

```

---

considering a candidate point  $\mathbf{CP}$  insertion:

- Distance to first (DF): Distance from the candidate point  $\mathbf{CP}$  to the first particle placed in the pack. This metric, used in [Wang *et al.* 2007, Lozano *et al.* 2016], promotes a concentric generation around the first particle.
- Inside or outside the outer loop (IN): The point  $\mathbf{CP}$  in most cases is located outside the outer loop; but sometimes can be located inside a polygon in the mesh. Collisions with neighbor particles are not possible because candidate points have no collisions.
- Number of new polygons (NP): The number of polygons that a new particle can create is computed based on the connections with other disks. Initially, the new particle has two exact contacts, so a range search is necessary to detect other contacts.
- Highest polygon size (PS): The highest polygon size among all the new polygons.

Some metrics are minimized others maximized:

- $\min(\text{DF})$  A shorter distance to the first particle yields a concentric pack.
- $\max(\text{IN})$  Inside candidates are preferable to outside candidates because they are filling empty spaces generated by previous insertions.
- $\max(\text{NP})$  More polygons are desirable because it implies that the candidate has more contacts with other particles in the pack.
- $\min(\text{PS})$  A polygon with fewer sides is preferred because it wastes less space than large-sized polygons. The ideal polygon is the triangle.

### 4.2.1

#### Ranking

We proceed to rank the points from best to worst with the metrics. The method classifies the candidates in two sets based on the IN metric value, the IN\_SET and the OUT\_SET. When there is one or more points classified as “inside candidates”, only these are considered for the ranking. Otherwise, the method chooses a point from the OUT\_SET.

If,  $\text{IN\_SET} \neq \emptyset$ , the algorithm chooses a  $\mathbf{CP} \in \text{IN\_SET}$  with the metrics:

$$\max(\text{NP}(p_i)), \quad \min(\text{DF}(p_i)) \quad (4-7)$$

Otherwise, the algorithm chooses a  $\mathbf{CP} \in \text{OUT\_SET}$  with the metrics:

$$\min(\text{PS}(p_i)), \quad \max(\text{NP}(p_i)), \quad \min(\text{DF}(p_i)) \quad (4-8)$$



We consider the metrics in the left to right order given above. If there is a tie, then we evaluate the next metric. Note that our new criteria give higher priority to the number of contacts and also seeks to reduce the creation of large polygons. We still use the distance to the first particle as the last metric to reinforce the concentric generation.

## 5

### Packing improvements

Repacking is a complex task because, depending on the particle distribution, it may be challenging to find radii within the  $[r_{min} - r_{max}]$  range that could be inserted within the pack or near the container boundary. Moreover, if we only accept small radii or even radii below  $r_{min}$ , the pack will not respect the desired PDF.

We devise three optional repacking procedures that will increase the packing density and coordination number. The first two procedures could be used during the iterative loop of the generation. The third is proposed as a post-packing procedure.

#### 5.1

##### Internal strategy (IS)

This procedure aims to place incoming particles, during the main loop of the algorithm, inside previously constructed polygons. The procedure contemplates any polygon in the pack, not only polygons associated with the current front, which are already considered in the main loop.

We prescribe this strategy for arrangements with heterogeneous particle size or a large radii range where it is likely to place particles inside the resultant polygons' void space. Here, the method will handle a set of polygon holes created and consumed during the pack creation.

##### 5.1.1

##### Creation of the set of polygon holes

Immediately after a polygon is added to the mesh, we compute the longest radius, within the given particle size range, that could be placed inside the polygon in contact with two vertices.

First, we match the associated particles of the polygon in triplets. For every combination of three disks, we compute the Soddy's circle, a fourth particle tangent to the triplet. The radii of the four disks are related by an equation known as the Descartes' theorem [Lagarias *et al.* 2002]

$$(k_1 + k_2 + k_3 + k_s)^2 = 2(k_1^2 + k_2^2 + k_3^2 + k_s^2)$$

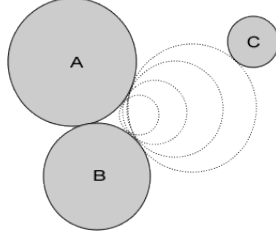


Figure 5.1: The maximum size of a disk tangent to the particles  $A$  and  $B$  depends on the closest third particle  $C$ .

where  $k_s = 1/r_s$  and  $r_s$  are the fourth disk's curvature and radius respectively. However, the Descartes' theorem only considers mutually tangent circles, so, to compute the fourth internal particle in general situations, we use the three equations of the triplet, i.e.

$$\begin{aligned}(x - x_1)^2 + (y - y_1)^2 &= (r + r_1)^2 \\(x - x_2)^2 + (y - y_2)^2 &= (r + r_2)^2 \\(x - x_3)^2 + (y - y_3)^2 &= (r + r_3)^2\end{aligned}$$

The fourth particle is used as a reference for the upper bound radius for every pair of particles in each triplet. In Figure 5.1, we present two particles  $A$  and  $B$  that yield a new disk in contact. The maximum radius for a particle in contact with both disks will be determined by a third particle  $C$ .

We only collect disks with a size larger than  $r_{min}$  and that do not collide with other particles in the polygon. The resulting set of disks helps us identify the largest radius for each pair of vertices in the triplet.

Figure 5.2 shows a pentagon and the potential positions and radii for a 3-contact particle. Each 3-contact defines a hole: AED, ADC, and BAC. Following a greedy criterion, we just register the biggest radius into the set of holes. In this case, the hole composed of the particles ADC.

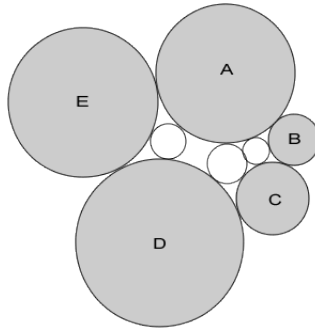


Figure 5.2: A pentagon formed by particles ABCDE with three Soddy circles without collisions.

### 5.1.2

#### Queries on the set of polygon holes

The query for a polygon hole is performed right after a particle radius is sorted. We aim to avoid all the candidate metric computations and rankings just to find a right polygon to be split with the new disk.

If the current radius  $r_{new}$  is bigger than the biggest radius in the set of holes, then the main algorithm proceeds as usual. Otherwise, we perform a binary search to find a valid hole with a radius value close to the  $r_{new}$ . The algorithm computes the candidate positions for each pair of particles in the triplet of the hole. In the end, the selected hole is removed from the set.

The selected candidate position will split the polygon associated with the hole; therefore, the method triggers the creation of the respective holes for the new polygons.

## 5.2

### Outer loop strategy (OLS)

The incremental mesh construction allows keeping track of the external frontier and the polygon additions after the insertion of new particles. This strategy aims to reorder particles near the polygonal mesh border immediately upon the detection of a new polygon that fits specific properties. The trigger condition may be based on the number of sides or the porosity of the new polygon.

This strategy contemplates three primary phases. The first step performs the temporary removal of some disks. Next, we run a local sub-packing algorithm to obtain new positions for the removed particles. Finally, we compare the previous configuration to the new one to determine if a rollback operation is needed.

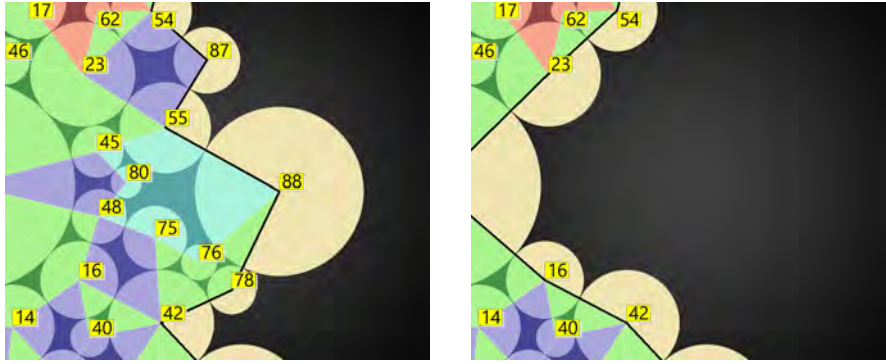
### 5.2.1

#### Withdrawal

The algorithm removes the disks  $p_i$  of the polygon that triggered the local reordering. Furthermore, to keep the 2-connectivity property of the segments in the “outer loop”, some surrounding particles should be pulled out from the pack and polygonal mesh. In addition to the removal of the respective vertices and polygons, the grid must reflect the withdrawal of particles from the container. Observe that the method does not need to remove the particles from the pack structure. We only need to assign to the involved particles an invalid position value, such as a coordinate outside the container. From now on, we will refer to these particles as the limbo particles  $L_p$ .

We can see, in Figure 5.3(a), a scenario where the generation of polygons with more than six sides executes this strategy. The polygonal mesh detects the creation of a heptagon after the placement of the 88-th particle. The withdrawal procedure removes the particles in the heptagon (88–76–75–48–80–45–55). Also, the 87-th and 78-th particles are removed because they belong to polygons destroyed as a consequence of the heptagon removal. They break the 2-connectivity rule.

Finding non-2-connected segments in the “outer loop” graph and identifying isolated particle clusters can be computationally expensive, considering that the loop grows with the assembly. Aiming to avoid this potential bottleneck and acknowledging that our method tries to optimize the positions of a set of particles in a local region, we extract a portion of the loop, centered at the removed polygon. The method performs the queries to verify the connectivity condition on this subgraph.



5.3(a): Particle 88 triggers the relocation.

5.3(b): Withdrawal.

Figure 5.3: OLS Example 1. Withdrawal step.

- (a) The placement of the 88-th particle triggers the relocation procedure; (b) Temporal withdrawal of particles 88, 76, 75, 48, 80, 45 and 55. Additional removed particles: 87 and 78; The procedure removed one irregular heptagon, three quads, and six triangles. ( $\triangle$  ■,  $\diamond$  ■,  $\square$  ■, He ■).

The algorithm stores the removed vertices, polygons and edges of the “outer loop” in local maps.

### 5.2.2 Relocation

The new relocation procedure is based on a greedy strategy, similar to the algorithm proposed by [Akeb 2014]. The “3DMHD greedy heuristic” solves the packing of spheres into 3D bins of fixed height and depth but variable length. This problem is known as the 3D strip packing problem. The algorithm employs a constructive greedy heuristic that computes a set of candidate positions for

the particles that are not yet placed into the bin. After the insertion of a new sphere, the set of candidate positions is updated. The procedure ends when all the particles are inside the bin.

We adapt Akeb's algorithm to our scheme and handle the limbo particles' reinsertion as a sub-packing problem.

Unlike our main iterative approach in Algorithm 2, here, the local optimization computes more candidate points taking advantage of the small set of incoming radii and the small number of local fronts composed of new particles at the border of the pack. The following sections describe the relocation strategy decomposed into three sub-procedures.

### 5.2.2.1

#### Local front processing

Compared to our main scheme that calculates positions for a single front and a single new radius, the local optimization computes more candidate locations by matching all the local fronts to the remaining radii at the limbo set. That is, for every front and radius combination, we collect new candidate points. In particular, for this relocation strategy, a candidate is identified by the triplet  $(rad_{id}, rad_{val}, pos)$ . Where  $rad_{id}$  corresponds to the id of a limbo particle,  $rad_{val}$  is the respective radius and  $pos$  is the new computed position. During the removal, we identify the new edges in the "outer loop" and use the particles connecting them as the local front from where the new particles will be relocated. In Figure 5.3(b) the new edges in the outer loop are (42-16; 16-x; x-23; 23-54). Based on this set of edges, we create a local front list with the particles (42-16-x-23-54).

To obtain the initial points, the `ProcessFronts` procedure pairs the front particles and radii described in the particle removal step. Additionally, as shown in Algorithm 4, it is possible that during the search for candidates, some fronts will not be able to generate new locations so, they are marked for removal from the local front set at the end of the procedure.

### 5.2.2.2

#### Candidates search

The search of candidates and the computation of points are analogous to the scheme used in Algorithm 2. A front  $f$  builds a box around its center to find the particles in its proximity. Then, computes the positions of the intersection of the halos between itself and another particle nearby with the radius  $r_{new}$ . Next, every point is tested with the container to guarantee that the new particle will be completely inside. We must verify that the new particle will not collide

**Algorithm 4:** ProcessFronts(*newFronts*,  $L_p$ )**Input** : front particles *newFronts* and set of limbo particles  $L_p$ .**Output:** Initial list of candidates for insertion and best candidate.

---

```

1 Create a set of fronts to remove ToRemove;
2 Create an empty set of candidates;
3 for  $l_p$  in  $L_p$  do
4   for  $f$  in newFronts do
5     SearchCandidates( $f$ ,  $l_p$ , ToRemove, candidates,
6                       bestCandidate);
6 Remove all fronts in ToRemove from newFronts;
7 Return candidates, bestCandidate;
```

---

with other particles in the pack. The procedure keeps a counter of the number of valid positions and the number of collisions with the container. Only if a front does not generate valid points for  $r_{new}$  and has no border collisions is marked for removal. The second condition avoids the premature removal of a front close to the boundaries of the container. However, if the front particle produces candidates for another limbo radius in future iterations, it is erased from the *ToRemove* set.

Finally, the new computed points are added to the set of candidates. The new *bestCandidate* is also identified.

Given that the relocation procedure computes more candidate position and can be CPU intensive, we use a simpler metric ranking in comparison to the ranking defined in Section 4.2.1. Here, we rank the number of polygons (NP) and distance to first (DF) metrics in the left to right order:

$$\max(\text{NP}(p_i)), \quad \min(\text{DF}(p_i))$$

Algorithm 5 summarizes the **SearchCandidates** procedure.

**5.2.2.3****Candidate's update**

Besides the modification in the underlying structures such as the assembly list, uniform grid, and polygonal mesh, additional updates are necessary for the context of the repacking.

First, all the candidates colliding with the new particle must be removed. The procedure must update the *bestCandidate* too. And second, the **SearchCandidates** procedure is used to compute new candidates product of the intersection of *newParticle* with its neighbors. If some fronts do not generate candidate points during this search, they are removed from the front list.

---

**Algorithm 5:** SearchCandidates( $f, l_p, ToRemove, candidates, bestCandidate$ )

---

**Input** : A current front  $f$  and a limbo particle  $l_p$ .

**Output:** Detects new *candidates* and fronts *ToRemove*. Identifies the *bestCandidate*.

```

1 Create an empty candidatePointList;
2  $box \leftarrow \text{computeNeighborhoodBox}(f, l_p.\text{rad}, r_{\max})$ ;
3  $closerParticles \leftarrow \text{gatherParticlesThatIntersect}(\text{gridIndex}, box)$ ;
4 for ( $neighbor$  in  $closerParticles$ ) do
5    $candidatePointList \leftarrow \text{add interceptions of halos of } neighbor \text{ and}$ 
    $\text{the front } f \text{ with } l_p.\text{rad}$ ;
6  $hasBorderCollisions \leftarrow \text{Exists a position in } candidatePointList$ 
   $\text{where } newParticle \text{ intercepts } boundaryMesh$ ;
7 remove from  $candidatePointList$  all positions where  $newParticle$ 
  intercepts  $boundaryMesh$  or any other  $closerParticles$ ;
8  $numberOfValidPoints \leftarrow \text{numberOfPointsIn}(candidatePointList)$ ;
9 if ( $numberOfValidPoints = 0$ ) then
10   if ( $\neg hasBorderCollisions$ ) then
11     add  $f$  to  $ToRemove$ ;
12 else
13   remove  $f$  from  $ToRemove$ ;
14  $candidates \leftarrow \text{add triples with } candidatePointList \text{ and } (l_p.\text{id}, l_p.\text{rad})$ ;
15  $bestCandidate \leftarrow \text{bestCandidateIn}(candidates)$ ;

```

---

Algorithm 6 summarizes the **Update** procedure.

---

**Algorithm 6:** Update( $newParticle, fronts, L_p, candidates, bestCandidate$ )

---

**Input** : A  $newParticle$ , the local *fronts*, the set of limbo particles  $L_p$ , the current set of *candidates* and the *bestCandidate*.

**Output:** The *candidates*, *fronts* and *bestCandidate* updated.

```

1 remove all the points in  $candidates$  that collide with the  $newParticle$ ;
2  $bestCandidate \leftarrow \text{bestCandidateIn}(candidates)$ ;
3 Create an empty set  $ToRemove$ ;
4 for ( $l_p$  in  $L_p$ ) do
5   SearchCandidates( $newParticle, l_p, ToRemove, candidates,$ 
    $bestCandidate$ );
6 remove all fronts in  $ToRemove$  from  $fronts$ 

```

---

#### 5.2.2.4

##### Local packing

Finally, a broader view of the **Relocate** procedure is summarized in Algorithm 7. The procedure receives a set of limbo particles  $L_p$  and a set of



*local fronts*. The former consists of the removed particles; the latter contains the particles of the edges at the pack's borders near that region. In the beginning, the **ProcessFronts** procedure computes the initial candidate points using the local front and the set of the new radius. Then, in every iteration, the best candidate becomes a particle and is reinserted into the pack, grid index, and mesh. Note that the *bestCandidate* is being updated continuously during the whole process. Then, the procedure removes from the *candidates* and  $L_p$  sets all the values matching the radius' id of the *bestCandidate*. Next, the **Update** procedure is responsible for eliminating colliding candidates with the new particle and the computation of the new candidates. The loop ends when there are no more available candidates.

---

**Algorithm 7:** Relocate(*pack*, *grid*, *mesh*, *fronts*,  $L_p$ )

---

**Input** : The new *fronts* and  $L_p$ .

**Output:** The updated *pack*, *grid* and *mesh*.

```

1 candidates, bestCandidate  $\leftarrow$  ProcessFronts(fronts,  $L_p$ );
2 while (candidates is not empty) do
3   Insert the bestCandidate into the pack, grid and mesh;
4   remove from candidates all values matching  $L_p$ .radiusId;
5   remove bestCandidate.radid from  $L_p$ ;
6   Update(bestCandidate, fronts,  $L_p$ , candidates, bestCandidate);
```

---

This procedure also records the added vertices, polygons and “outer loop” edges in local maps.

### 5.2.3 Rollback

Sometimes the new configuration of the relocated particles is worse than the undesired detected configuration. We use the void space area as a metric to compare scenarios before the withdrawal step and after the relocation step. We compute the area unoccupied by the particles on the removed and added polygons for both scenarios.

A small particle may subdivide a polygon inside the frontier (an inside polygon) during the relocation. In that scenario, the solid areas of the inserted disks are considered as negative void areas for the added polygons because they reduce the packing porosity instead of increasing it. On the other side, if an added polygon outside the frontier is split during the sub-packing, only the new polygons must be included in the computation.

If the new total void area is less than the removed polygons' void area, then we keep the new configuration. Otherwise, we rollback the whole operation and

keep the old setting. More precisely, we revert the added local maps and apply the removed local maps.

Figure 5.4, continues the example in Figure 5.3, the new configuration improves the local density of the pack. Note that particles 75, 76, 80, 48, and 45 become frontier particles and part of the “outer loop”, in consequence, they must be pushed into the queue of fronts of the main algorithm, so that they can be used to produce more particles around them in future iterations.

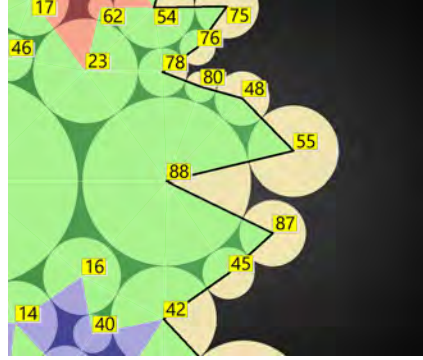


Figure 5.4: OLS Example 1 (Continuation). Good relocation with twelve triangles.

A good relocation yields a configuration that reduces the porosity of the pack. The method keeps the new positions. ( $\triangle$ ■,  $\diamond$ ■,  $\square$ ■).

On the other side, Figure 5.5 shows a case where the new positions of the particles increase the pack’s porosity.

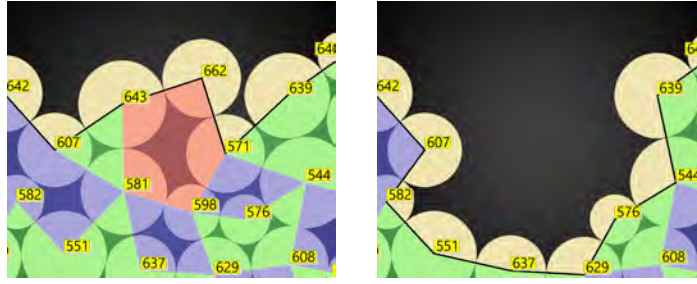
### 5.3

#### Boundary strategy (BS)

A common problem in advancing front generation methods is the presence of unoccupied areas between the final arrangements and the domain’s frontiers.

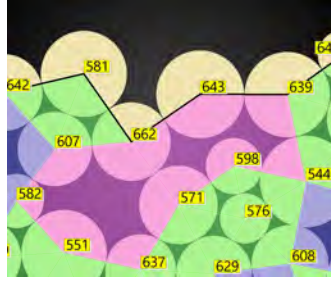
Some works offer methods to increase the contact of the disks with the container. [Liu *et al.* 2012] allows the generation of particles colliding with the frontiers and then reduces their radii to fit the gap region. [Dong *et al.* 2020] proposed a strategy that replaces particles within a certain distance to the borders, with particles of a greater size that are tangent to the boundaries.

In this work, we propose a local and simple procedure to continue the packing near the borders looking for new particles tangent to the frontier to improve arrangement stability under physical simulations. The procedure also increases the density. However, to keep adding small particles affects the given PDF. To mitigate the effect in the desired radii distribution, we specify a minimum allowed radius  $Br_{min}$ . Sizes below this parameter are rejected, and the border improvement terminates in that region.



5.5(a): Particle 662 triggers the relocation.

5.5(b): Particle withdraw.



5.5(c): Bad relocation. Adds two hexagons.

Figure 5.5: OLS Example 2. Bad relocation.

The greedy strategy generates two hexagons that waste more space than the previous arrangement. The algorithm rollbacks to the previous configuration.

( $\triangle$ ,  $\square$ ,  $\diamond$ ,  $\square$ ,  $\square$ ,  $\square$ ).

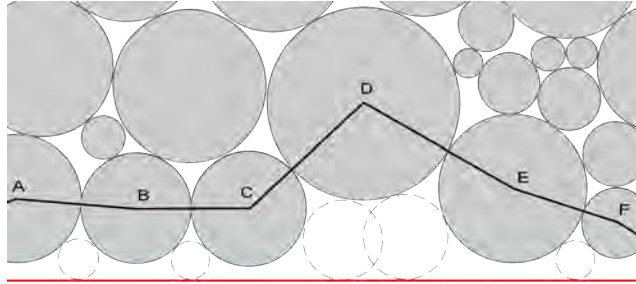


Figure 5.6: Border treatment example – First iteration.

In every iteration, the method goes through the “outer loop” pairing consecutive particles and particles at the extreme of consecutive triplets, trying to handle concave regions.

For every pair, we compute the minimum disk tangent to both particles without overlapping other disks. We grow its radius until the new particle is tangent to the boundaries. If the particle generates a collision or if the final disk size is smaller than  $Br_{min}$ , it is discarded.

All the computed particles are sorted by radii size in decreasing order. The strategy has a greedy criterion and gives priority to larger particles. Before placing a new disk, the method checks for collisions with recent insertions,

e.g., in Figure 5.6, the pairs CD and DE yield two particles tangent to the container. Only the biggest of both disks is inserted, the smallest will generate an overlapping.

After each insertion, the method updates the pack assembly, the “outer loop”, the polygonal mesh, and the grid index. The method is repeated until the pairs yield no new valid disk larger or equal than  $Br_{min}$ .

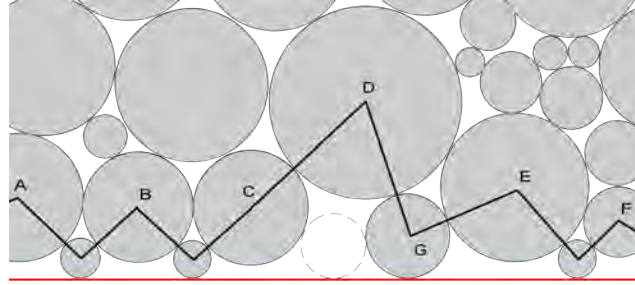


Figure 5.7: Border treatment example – Second iteration.

## 6 Results

This chapter presents the results of our method in different types of containers and particle distributions. Here we also analyze and tune the input parameters. Whenever possible, we compare the packs yielded by our algorithm in available scenarios for testing defined in previous works.

The first part of this chapter considers the packing inside simple and commonly used container geometries. In the second part, we focus on complex geometries extracted from slices of micro-CT images of rocks.

We analyze our algorithm, both in terms of the quality of the resulting pack and its efficiency. To perform a quality evaluation, we use several geometric properties presented below. To illustrate the efficiency, we present the algorithm's execution time in an Intel Core i5-3330 @ 3.00 GHz with 8 GB of RAM in Windows 7 64-bit.

### 6.1 Geometric characteristics

The evaluation of the quality of the packs uses the following geometrical properties:

- Density( $\rho$ ): Is the ratio between the total area of the  $N$  particles of the arrangement and the container area  $A_c$ .

$$\rho = \frac{1}{A_c} \sum_{i=1}^N \pi r_i^2 \quad (6-1)$$

- Porosity( $\phi$ ): Is the ratio of the total area of the void space and the container area  $A_c$ . It can be computed as the complement of the density.

$$\phi = 1.0 - \rho \quad (6-2)$$

- Mean coordination number( $Z$ ): Characterizes the packing's local density and determines the topological connectivity of the system. It is an important property because of its relation to the local mechanical stability and rigidity of the assembly [Papadopoulos *et al.* 2018]. The coordination number is a disk property representing the number of neighbor particles in contact ( $c$ ). The mean value is computed as

$$Z = \frac{1}{N} \sum_{i=1}^N c_i \quad (6-3)$$









- Fabric tensor: A directional quantity to characterize directionally dependence in microstructures. It quantifies the alignment of the solid and pore space, the fabric of the microstructure. It is used for granular assemblies to determine if the pack exhibits a random (isotropic) distribution or a degree of directional preference. If  $N_c$  are the total number of contacts, the fabric tensor is given by

$$\varphi_{ij} = \frac{1}{N_c} \sum_{\alpha=1}^{N_c} n_i n_j, \quad (6-4)$$

where  $n_i$  and  $n_j$  are the components of the 2D unit vector  $\vec{n}$  of the contact  $\alpha$ . Equation 6-4 yields a 2x2 matrix. The eigenvalues of this matrix represent the anisotropy of the assembly. An isotropic assembly has eigenvalues ( $\beta_1 = 1/2, \beta_2 = 1/2$ ). For random particle arrangements in DEM, the packing should ensure the geometric isotropy and homogeneity of the medium [Ghasemi *et al.* 2020].

- Angular distribution: Is the orientation of particles around its near neighbors. The distribution illustrates if the arrangement prefers any particular angle [Tulluri 2003].

Additionally, since the polygonal mesh is an output of the algorithm that reflects the connections between the particles around the void spaces, we consider:

- Polygon frequencies: We quantify the number of triangles ( $\triangle$  ) , quads ( $\diamond$  ) , pentagons ( $\lozenge$  ) , hexagons ( $\circ$  ) , heptagons (He ) , octagons ( $\bigcirc$  ) , nonagons (No ) and decagons (De ) of the mesh.

### 6.1.1

#### Contact and collision thresholds

Contacts among particles are commonly studied in the literature of granular packings. And, despite being a straightforward geometrical computation that uses the positions and sizes of the disks, studies in the literature use threshold contact values to determine if two particles are touching. [Tulluri 2003] and [Aste *et al.* 1992] report the use of  $0.05d_m$ , where  $d_m$  is the diameter of the particle, to include nearly touching particles as contacts. [Ardanza *et al.* 2014] explores contact thresholds in the range of  $[d_m, 1.12d_m]$  to analyze the morphological structure of granular samples in mechanical equilibrium.

Considering two particles  $i$  and  $j$  with a distance  $d_{ij}$  between their centers, we consider the overlapping as:

$$\omega_{ij} = d_{ij} - (r_i + r_j), \quad 1 \leq j < i \leq N$$

In the present work, a particle  $i$  is in contact with another particle  $j$  if the distance between their centers is within 5% of the sum of both radii. We compute the contact threshold as:

$$\epsilon_Z(i, j) = 0.05 \times (r_i + r_j)$$

and comparing the overlapping versus the contact threshold,

$$\begin{aligned} \text{if } \omega_{ij} > \epsilon_Z(i, j) : i \text{ and } j \text{ are separated,} \\ \text{if } \omega_{ij} < \epsilon_Z(i, j) : i \text{ and } j \text{ are in contact.} \end{aligned}$$

To detect collisions between particles we use the threshold value defined in Section 4.1.2.

## 6.2

### Algorithm variants

To evaluate how the new heuristics and optimizations perform from a geometric perspective, six main algorithm variants will be considered:

- AF16: The direct implementation of the advancing front algorithm described in [Lozano *et al.* 2016] that only uses the distance to first (DF) criteria to select the next best position.
- AFMesh: Our algorithm with the new front removal logic and the new heuristics.
- AFMeshIS: The AFMesh variant plus the internal strategy (IS).
- AFMeshOLS: The AFMesh variant plus the outer loop improvement (OLS).
- AFMeshOLSIS: The AFMeshOLS variant plus the internal strategy (IS).
- AFMeshOLSISBS: The AFMeshOLSIS variant plus the boundary treatment (BS).

### 6.3

#### Analysis of the front removal parameters

This section examines the inclusion of the *maxRejections* and *maxQueueSize* parameters on the AF16 variant. The new parameters are introduced to avoid the premature removal of fronts generating particles colliding with the domain boundaries.

The test scenario uses a rectangular domain with dimensions  $[50u, 50u]$  with a radius range between  $[0.1u - 0.5u]$ . Figure 6.1 illustrates the histogram for the radius distribution. We consider six values for the maximum number of times a front can be rejected (1, 2, 4, 6 and 8) and seven values for the maximum size of the queue of newly rejected particles *qnr* (1, 2, 4, 8, 10, 12 and  $\infty$ ).

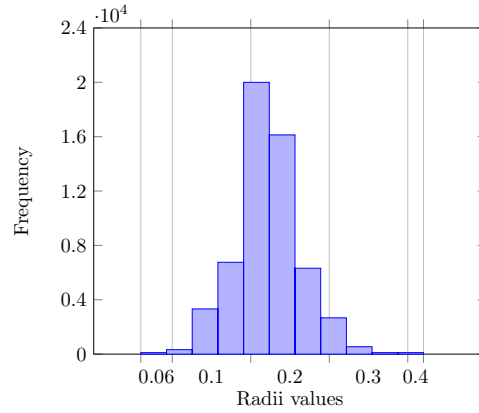


Figure 6.1: Frequency histogram.

For each parameter combination, we execute 50 runs and present their mean values. Specifically, we are interested in four outputs: the number of particles  $N$ , the number of elements in the queues *qnr* and *qpr*, and the mean distance of the particles in the “outer loop”. The mean distance indicates how close to the domain frontier are the particles at the border of the arrangement. The heat maps in Figure 6.2 depict the results of the executions varying the two new parameters. Higher values are preferable in the case of the number of particles. In contrast, in the other three outputs, lower values are desirable. We see that increasing the number of rejections increases the number of disks and, inversely, reduces the mean distance. More disks are inserted near the borders. The size of the queues *qnr* and *qpr* grows too, but in this scenario, the impact of the  $\approx 50$  particles ( $qnr + qpr$ ) in the PDF is small given that it is 0.001% of 43,000 particles. A bigger impact is expected for smaller domains or bigger particles. To increase the packing near the boundaries and to respect the sizes in the order provided by the number generator maintaining small queues, the following experiments will consider a fixed *maxQueueSize* of 10 and a *maxRejections* of 6.



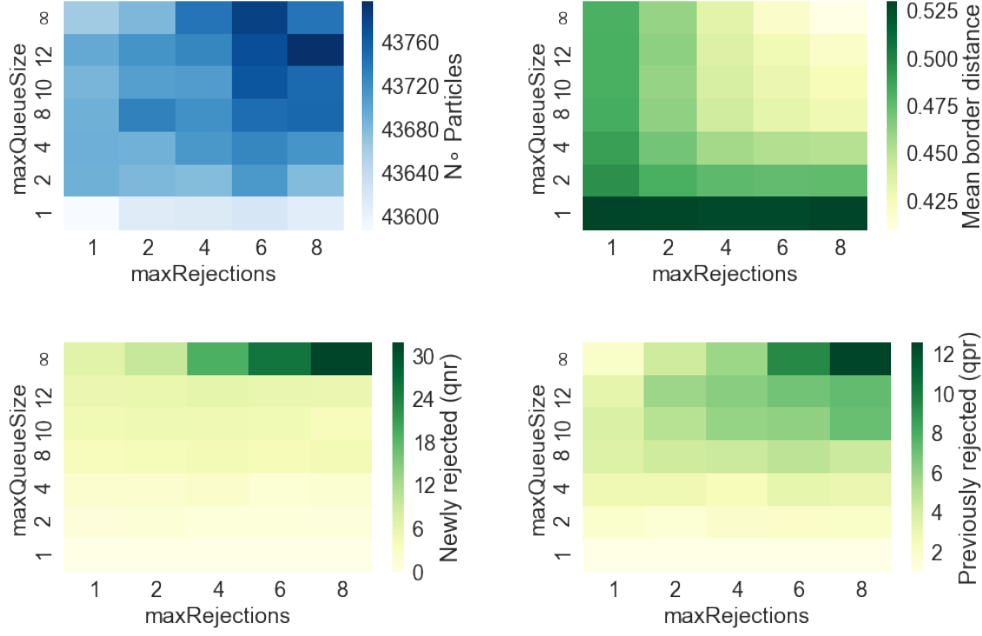


Figure 6.2: Front removal logic

#### 6.4 Monodisperse and bidisperse arrangements

We present a pack inside a circular container of radius  $10u$  using a constant size radius of  $0.25u$ . For the monodisperse scenario, there is no difference between the AF16 and the AFMesh variants. Due to the criteria that place new particles in contact with two disks, the algorithms yield regular packs. In hexagonal arrangements, every inner particle is surrounded by six disks. The preferred contact angles for the pack are reflected in Figure 6.3. Those angles are defined by the positions of the two first seed particles. In the example, the first disk was placed at the center of the circle while the position of the second was randomly selected.

The scenario obtains a density of  $\rho = 0.870$  and a mean coordination number of  $Z = 5.80$ . After the execution of the refilling procedure BS with a minimum size of  $Br_{min} = 0.15u$  the pack inserts 36 new disks achieving  $\rho = 0.883$  and  $Z = 5.73$ .

To obtain bidisperse arrangements, we use the AFMeshOLS variant and a Bernoulli random number generator with a 50.0% of probability for the maximum  $r_{max}$  and minimum  $r_{min}$  radius. The experiment explores packs with a 1:1.4 ratio that, according to physical simulations in [O'Hern *et al.* 2002, Donev *et al.* 2004, Henkes *et al.* 2007, Meyer *et al.* 2010] report a density of  $\rho \approx 0.84$ .

We display the average outputs of 100 instances in Table 6.1. Figure 6.4 and

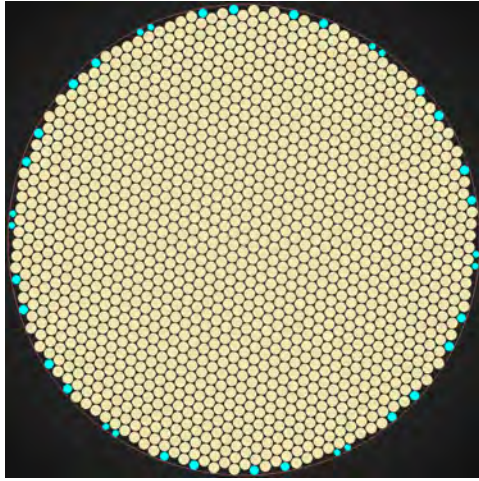
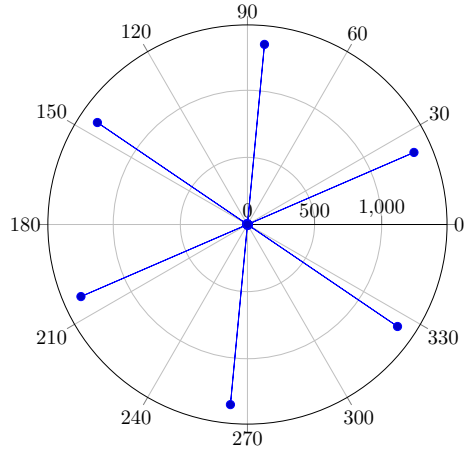
6.3(a): Pack with refill.  $\rho = 0.883$ .6.3(b): Contact orientation.  $Z = 5.73$ .

Figure 6.3: Monodisperse pack inside a circle of  $10u$  radius with disks of radius  $0.25u$  with a refilling procedure ( $0.15u$  minrad).

Figure 6.5 depict an instance of the runs. The latter illustrates the left bottom corner of the rectangular container. It shows a pack with 22,220 disks with  $r_{min}$  and 22,080 disks with  $r_{max}$ . On average, our bidisperse packs achieved a 0.822 density. Compared to the fixed size pack, the contact angle distribution shows that the arrangements have no preferred angles.

Density ( $\rho$ )	Coord Number ( $Z$ )	F. Tensor ( $\beta_1, \beta_2$ )	Particles ( $N$ )	Time (s)
0.8212	4.2782	0.50,0.49	44,165	5.65

Table 6.1: Bidisperse packs with AFMeshOLS – Average of 100 instances.

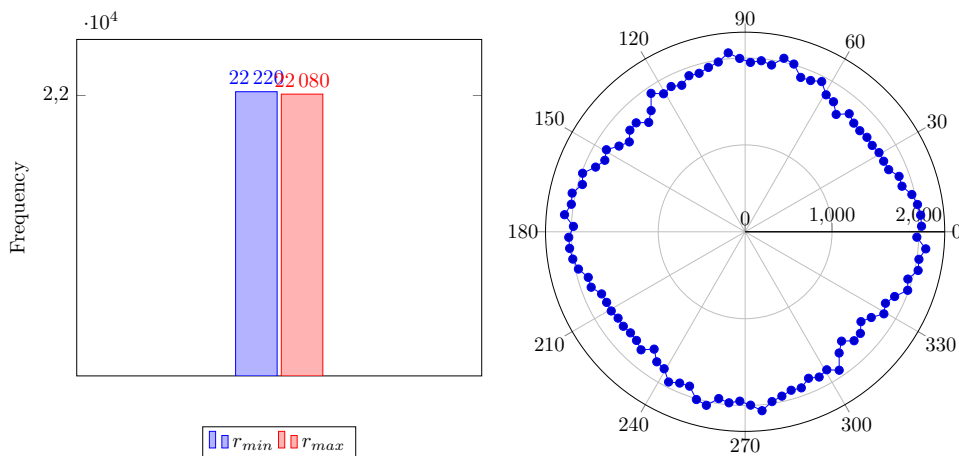


Figure 6.4: Bidisperse pack of instance N°35 – Radius and contact frequencies.

The following sections will focus on the generation of heterogeneous arrangements.

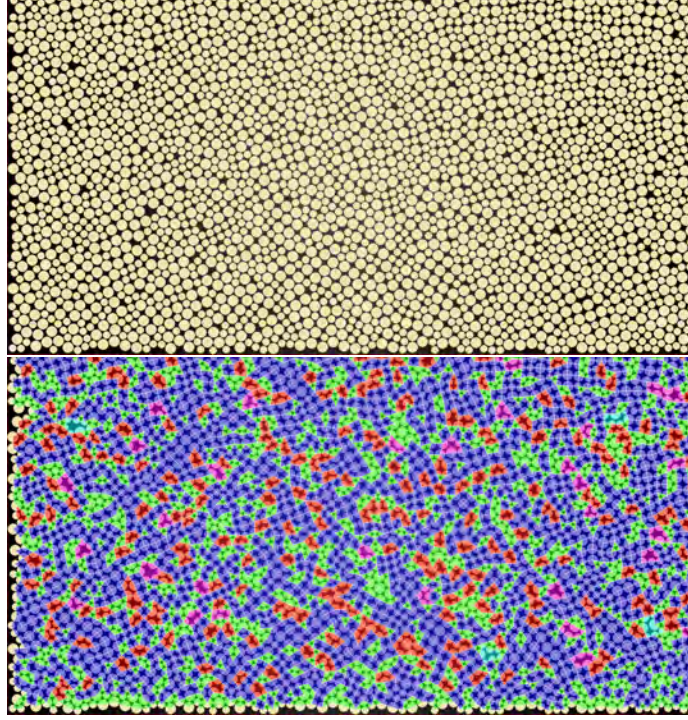


Figure 6.5: Bidisperse pack of instance N°35 – Ratio 1:1.4  
( $\triangle$  ■,  $\diamond$  ■,  $\square$  ■,  $\square$  ■, He ■).

## 6.5

### Variants comparison

To compare the variants, and only for the first two experiments in this section, we remove the method's random component. To obtain a deterministic behavior implies that the three-particle seeds are the same in all the variants (size and position). We have also created a data set of random numbers for the uniform  $([0.2u - 0.4u])$  and truncated log-normal  $([0.03u, 0.50u], \mu = -2.0$  and  $\sigma = 0.5)$  distributions. The algorithms use these numbers in the same order, and the needed numbers to fill the domains were computed based on the area of the containers and the radii distributions, plus a 25%.

First, we present a small test with a rectangular container of dimensions  $[15u, 35u]$  and fill it with disks following the uniform distribution. Table 6.2 presents some geometric properties of the packs and polygonal meshes and Figure 6.6 illustrates the respective results. We observe that the AFMesh and AFMeshOLS variants increased the density and the mean coordination number by incrementing the number of the triangles and decreasing the number of the other polygons.

Next, we present the scenario where we pack particles following a truncated lognormal distribution inside a circular container with a radius of  $10u$  employing four variants. Figure 6.7 illustrates the outcomes and Table 6.3



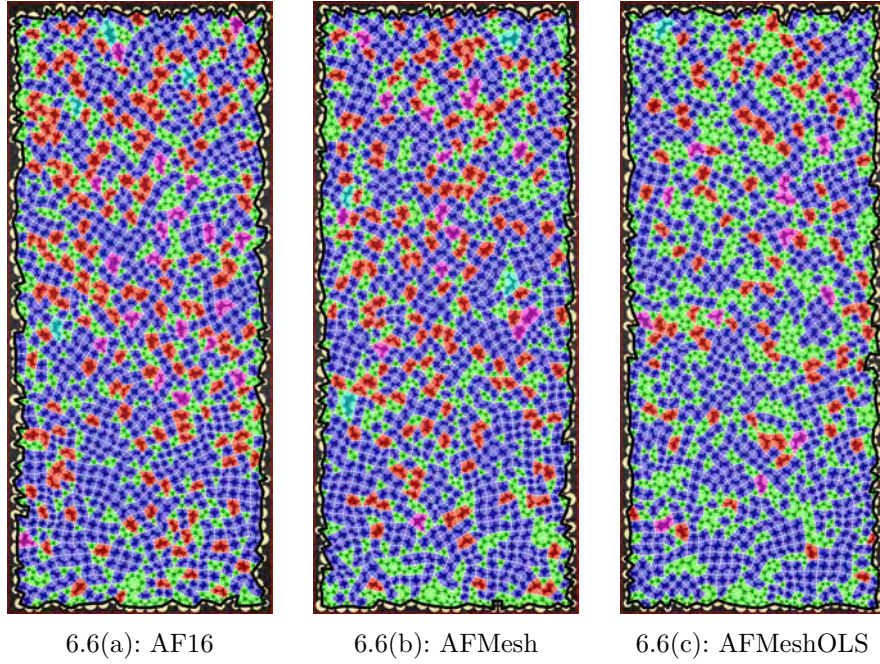


Figure 6.6: Uniform packs inside a rectangular container with three variants using the same data set.

( $\triangle$  ■,  $\diamond$  ■,  $\square$  ■,  $\square$  ■).

	$N$	$\rho$	$Z$	$\psi_i$				
				$\triangle$	$\diamond$	$\square$	$\square$	$\geq 7$
AF16	1,424	0.788	4.102	635	717	<b>159</b>	<b>20</b>	<b>4</b>
AFMesh	1,449	0.800	4.135	669	<b>761</b>	139	15	<b>4</b>
AFMeshOLS	<b>1,464</b>	<b>0.808</b>	<b>4.281</b>	<b>954</b>	725	88	13	1

Table 6.2: Uniform pack results summary.

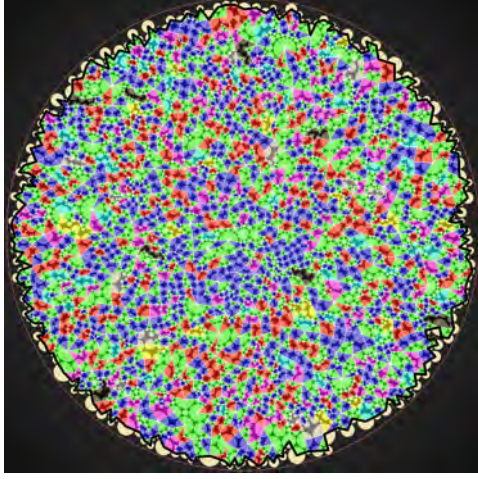
	$N$	$\rho$	$Z$	$\psi_i$				
				$\triangle$	$\diamond$	$\square$	$\square$	$\geq 7$
AF16	2,811	0.808	4.192	1,827	811	<b>317</b>	118	<b>85</b>
AFMesh	2,878	0.824	4.189	1,819	935	313	<b>135</b>	57
AFMeshOLS	2,917	0.840	4.539	2,792	951	194	48	20
AFMeshOLSIS	<b>2,953</b>	<b>0.853</b>	<b>4.633</b>	<b>3,014</b>	<b>1,041</b>	126	32	10

Table 6.3: Lognormal pack results summary.

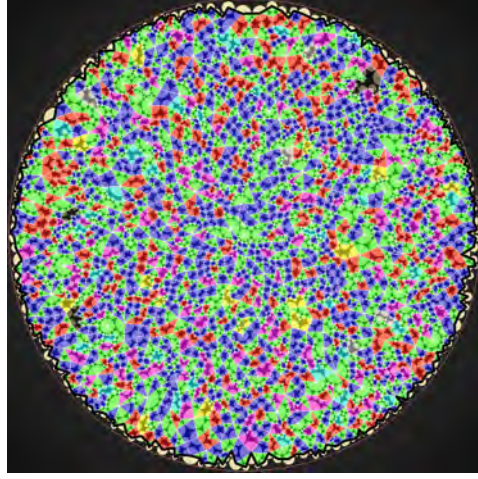
summarizes the geometric properties.

In both examples, we can see that the AF16 variant using only the DF metric yields many polygons with a high number of sides. Consequently, they present a low density and mean coordination number compared to the other two variants. The new metrics' use slightly improves the properties and particularly increases the number of quads in the mesh.

The OLS procedure can locally rearrange the pack and produced more triangles and quads, leading to an increase in the mean coordination number and pack



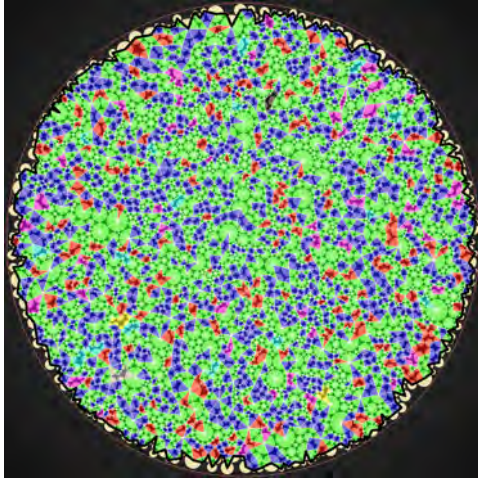
6.7(a): AF16



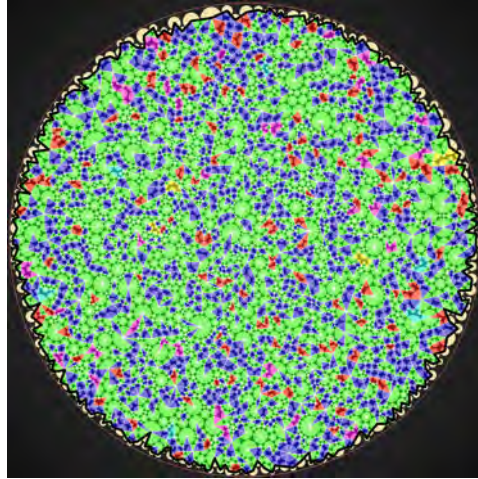
6.7(b): AFMesh

Figure 6.7: Lognormal packs inside a circular container with four variants using the same data set.

( $\triangle$ ,  $\diamond$ ,  $\square$ ,  $\square$ , He,  $\square$ , No, De).



6.8(a): AFMeshOLS



6.8(b): AFMeshOLSIS

Figure 6.8: Log-normal packs inside a circular container with four variants using the same data set (continuation).

density.

We did not test the IS procedure to the rectangle example. In scenarios using a short-range uniform distribution, the algorithm produces few polygons (or not polygons at all) with enough space to place disks inside. On the other hand, for highly heterogeneous arrangements, it can further improve the geometric properties.

Despite the increase of particles inside the container, our method still respects the given PDF. We display in Figure 6.9(a) a comparison of the lognormal curve versus the radius frequencies, in 40 bins, of the particles of the AFMeshOLSIS variant. We also plot the contact orientations in Figure 6.9(b),

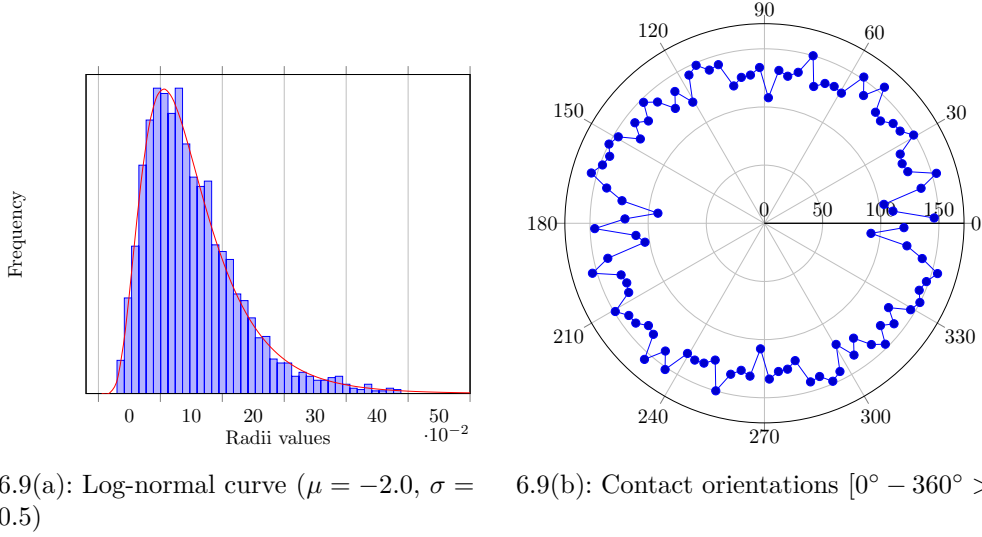
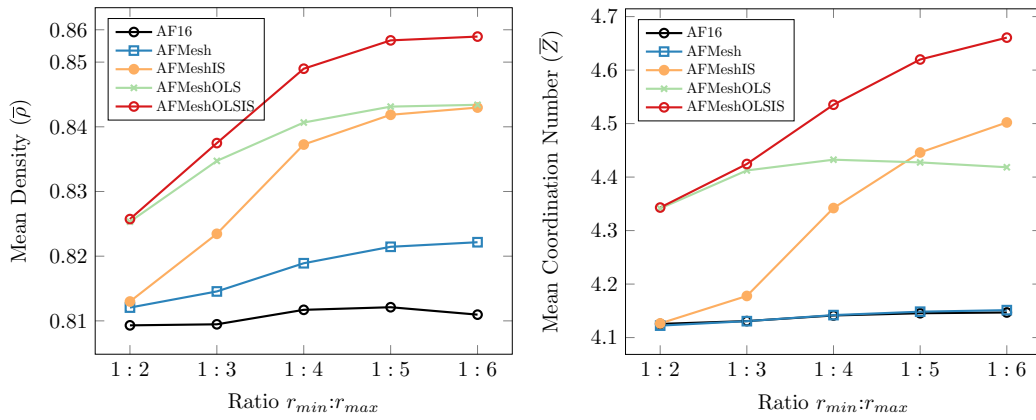


Figure 6.9: Verification of radius frequencies and contact orientations for the log-normal test with AFMeshOLSIS

it shows that there are no preferred contact angles. The respective eigenvalues of the pack are  $(\beta_1 = 0.498, \beta_2 = 0.505)$ , the pack is isotropic.

In a third experiment, we aim to measure the basic geometric pack properties, density and mean coordination, and the execution time performance of our method, varying the minimum and maximum radius ratio.

The test scenario uses a rectangular container with fixed height (50u) and variable width (50u - 2,500u). At each run we increase the width in 50u and create arrangements with a uniform distribution using five variants and five ratios; 1:2 for [0.2u-0.4u], 1:3 for [0.2u-0.6u], 1:4 for [0.2u-0.8u], 1:5 for [0.2u-1.0u] and 1:6 for [0.2u-1.2u]. For the OLS improvement we use the trigger condition  $(\phi_p \geq 0.222$  or  $\psi_p \geq 5)$ .



6.10(a): Density  $\rho$ .

6.10(b): Mean coordination number  $Z$ .

Figure 6.10: Variant comparison with different ratios.



Figure 6.10 presents the averages of the 50 runs for the variants and ratio cases. The AF16 method exhibits the lowest values and only matches the mean coordination number of the AFMesh variant. We observe that the internal strategy (IS) does not improve the geometric properties for the 1:2 ratio because the method cannot generate polygons with enough space to place particles inside. When the ratio increases, all the variants using the improvements increment both the density and mean coordination number in comparison to the AFMesh variant.

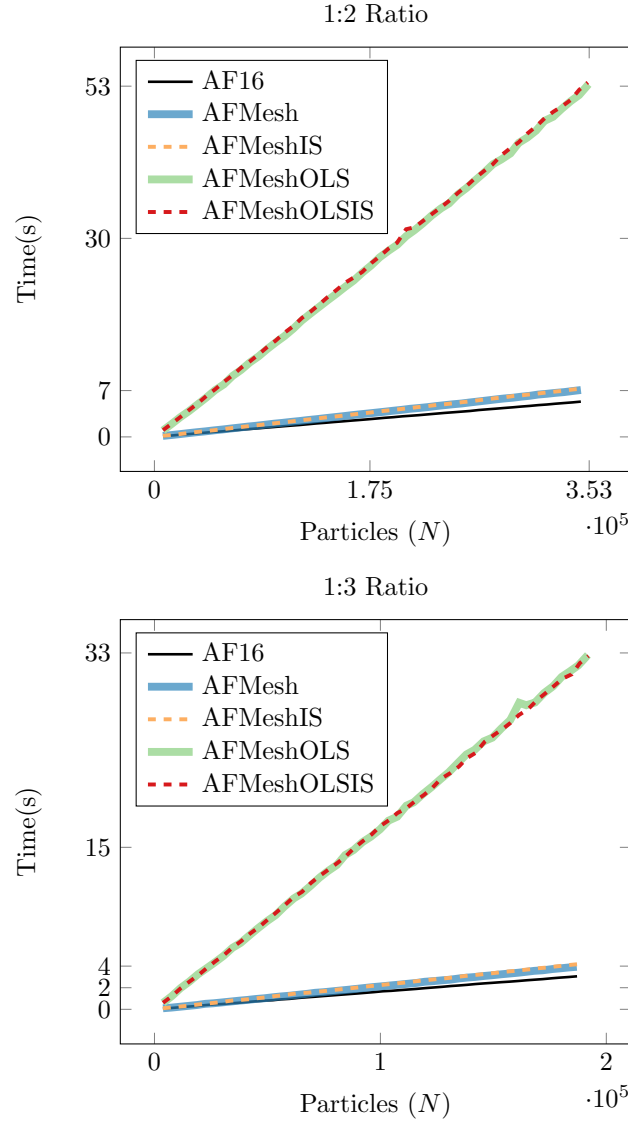


Figure 6.11: Particles ( $N$ ) vs Time(s) for 1:2 and 1:3 ratios.

In Figure 6.11, Figure 6.12, and Figure 6.13, we depict the execution times for the runs of each rectangle packing. For these experiments, the algorithm shows a linear behavior on the number of particles  $N$  for all variants and ratios. We see that for the first three ratios, the use of the IS improvement does not impact the performance. The plotted lines overlap. The influence of

the IS is stronger for the last two ratios because a wider particle size range produces bigger polygon holes. The additional consumed time is acceptable based on the gains in density and mean coordination number.

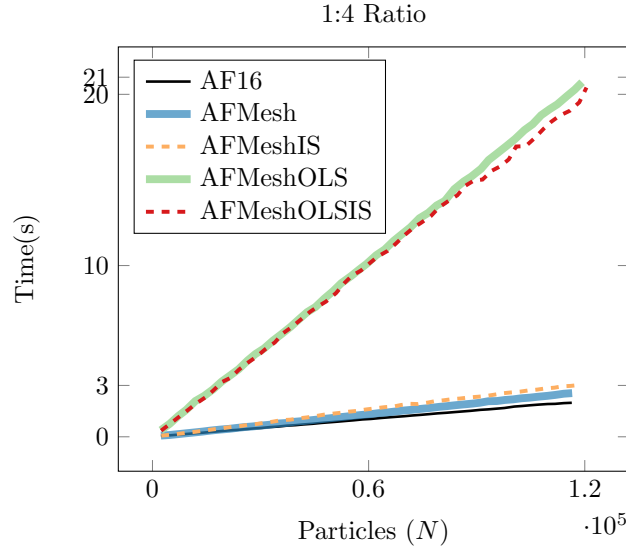


Figure 6.12: Particles ( $N$ ) vs Time(s) for the 1:4 ratio.

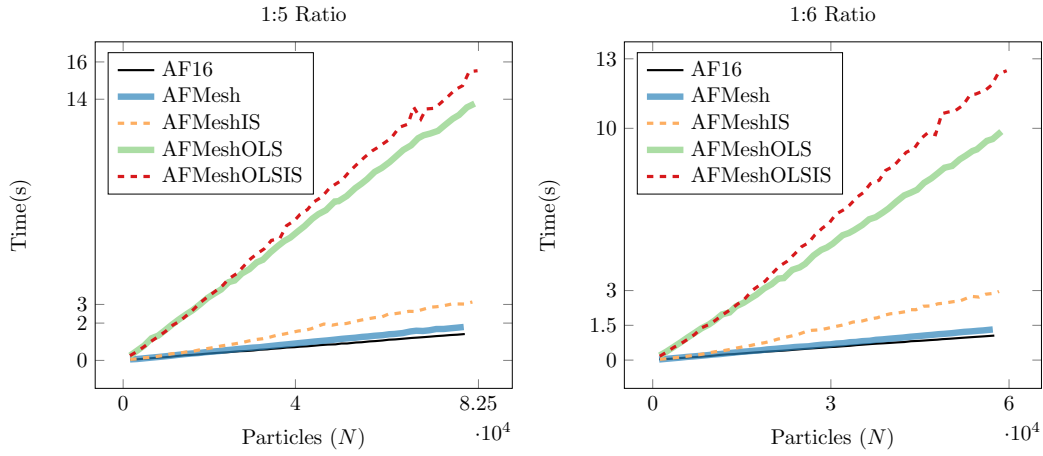


Figure 6.13: Particles ( $N$ ) vs Time(s) for 1:5 and 1:6 ratios.

The OLS improvement outperforms most of the other variants. Only using a 1:5 ratio, the AFMeshIS variant matches the AFMeshOLS and with a ratio of 1:6, outperforms the mean coordination number. AFMeshOLSIS is the variant with the best geometric outcomes. On the other hand, the OLS yields the highest execution times. The OLSIS can produce up to 353K particles in 53 seconds for the 1:2 ratio, 192K particles in 32 seconds for the 1:3 ratio, 120K particles in 20 seconds for the 1:4 ratio, 82K particles in 15 seconds for the 1:5 ratio and 60K particles in 12 second for the 1:6 ratio. Each case presents



an average density of 0.825, 0.837, 0.848, 0.853 and 0.853, and average mean coordination of 4.343, 4.424, 4.535, 4.619 and 4.660, respectively.

## 6.6

### Analysis of the OLS

Right after creating a polygon that meets a certain condition, a local optimization is performed to relocate a subset of particles. In this section, we analyze how the trigger condition for the “outer loop” improvement strategy varies the pack outputs.

To be specific, we explore the influence of the polygon’s porosity  $\phi_p$  ( $\geq 0.214$ ,  $\geq 0.229$  and  $\geq 0.244$ ) and number of sides  $\psi_p$  ( $\geq 5$ ,  $\geq 6$  and  $\geq 7$ ) on the output to identify which performs better.

Table 6.4 summarizes all the combinations for the relocation trigger, including the “no relocation” which is equivalent to the AFMesh variant. In total, we have 16 scenarios, and for each one, we execute 50 simulations.

01: $\psi_p \geq 5$	07:( $\phi_p \geq 0.214$ or $\psi_p \geq 5$ )	13:( $\phi_p \geq 0.244$ or $\psi_p \geq 5$ )
02: $\psi_p \geq 6$	08:( $\phi_p \geq 0.214$ or $\psi_p \geq 6$ )	14:( $\phi_p \geq 0.244$ or $\psi_p \geq 6$ )
03: $\psi_p \geq 7$	09:( $\phi_p \geq 0.214$ or $\psi_p \geq 7$ )	15:( $\phi_p \geq 0.244$ or $\psi_p \geq 7$ )
04: $\phi_p \geq 0.214$	10:( $\phi_p \geq 0.229$ or $\psi_p \geq 5$ )	16: None
05: $\phi_p \geq 0.229$	11:( $\phi_p \geq 0.229$ or $\psi_p \geq 6$ )	
06: $\phi_p \geq 0.244$	12:( $\phi_p \geq 0.229$ or $\psi_p \geq 7$ )	

Table 6.4: Combination of relocation triggers in 16 scenarios.

The following experiment uses a fixed container of dimensions [100u, 100u] and a uniform distribution in the range of [0.2u-0.4u]. We measure the density, mean coordination number, execution time, polygon frequencies, and the number of relocations during the pack generation.

The 800 outputs are presented in Figure 6.14, Figure 6.15 and Figure 6.16 organized in 6 plots. For each plot, we cluster the outputs to identify the associated inputs scenarios.

The obtained arrangements had densities between the range of [0.813, 0.828], producing disks between [27,614 - 28,362].

The best results in density are achieved by scenarios with  $\psi_p \geq 5$ . On the other hand, the worst correspond to the scenarios with no relocation and the relocation only with  $\psi_p \geq 7$ . The probabilities of generating heptagons here are small; thus, the relocation strategy is not triggered as much as when the algorithm detects a pentagon.

Figure 6.14(a) shows that both the coordination number and density increases when the relocation condition is more strict. This is explained by the generation of more triangles and fewer polygons with a high number of sides.

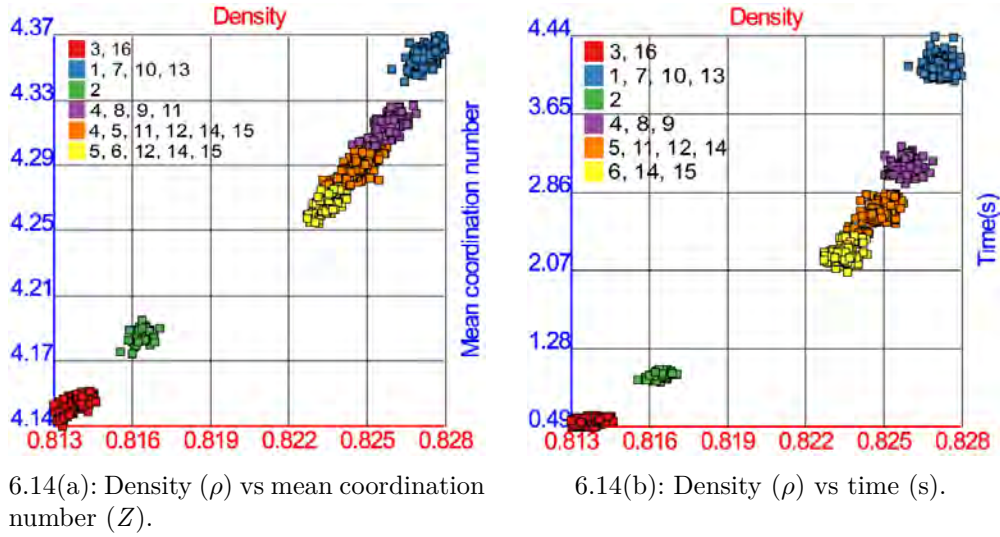


Figure 6.14: Clustering of algorithm's outputs.

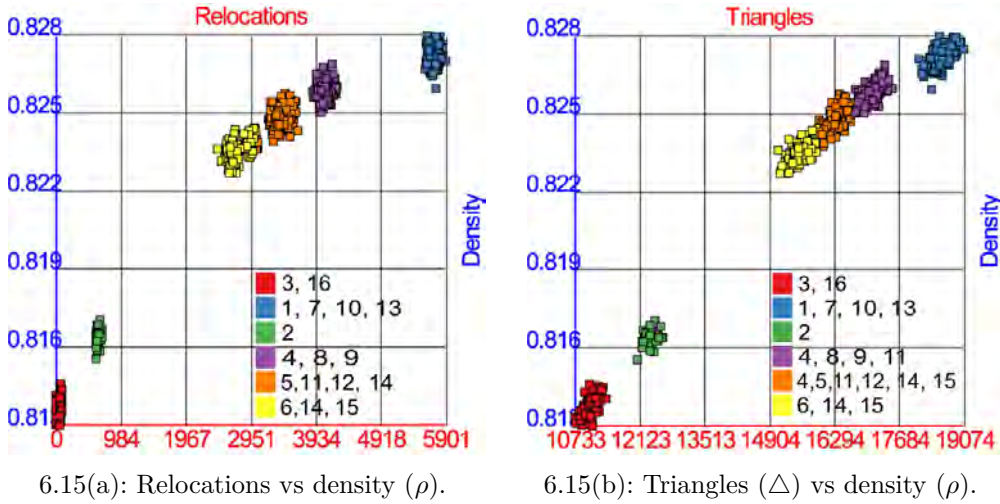


Figure 6.15: Clustering of algorithm's outputs (continuation).

Figure 6.15(b), Figure 6.16(a) and Figure 6.16(b) present the number of polygons in the scenarios. A high density is directly related to the high numbers of triangles and quads. Moreover, it is inversely related to the number of pentagons. The scenarios with  $\psi_p \geq 7$  and “no relocation” yield more pentagons than the others.

Regarding the packing density versus the time consumption in Figure 6.14(b), we can observe an exponential behavior to achieve higher densities. In conclusion, if the trigger to execute the local improvement has a strict condition regarding the number of sides or the porosity of a polygon, the algorithm will achieve better results at the cost of execution time.

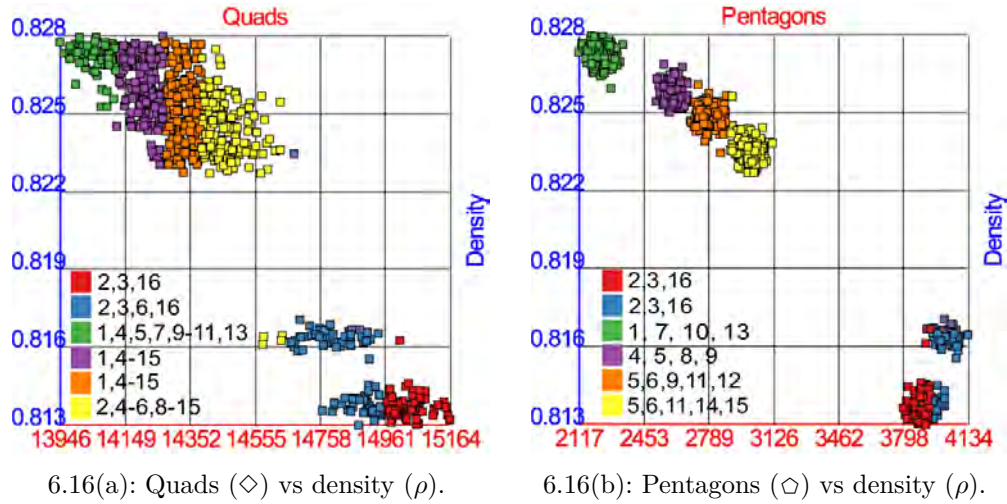


Figure 6.16: Clustering of algorithm's outputs (continuation).

## 6.7

### Analysis of the BS

To test the optional boundary refilling of the packs, we use the scenario proposed in Section 6.3. A rectangular box of dimensions  $50u \times 50u$  with particles in the range of  $[0.067-0.45]$  following a frequency histogram resembling a discrete truncated Gaussian distribution. After the arrangement generation, we apply the boundary strategy with different  $Br_{min}$  values.

$Br_{min}$	Density	Coord Number	Particles
-	0.824	4.352	11,090
0.20	0.824	4.352	11,092
0.10	0.830	4.349	11,367
0.08	0.832	4.346	11,505
0.06	0.833	4.347	11,740
0.04	0.834	4.341	12,181

Table 6.5: Boundary strategy results varying the  $Br_{min}$  value.

Table 6.5 presents the obtained number of particles, density and mean coordination number varying the value of the minimum allowed refilling particle size  $Br_{min}$ . Figure 6.17 illustrates the refilling.

It is logical to obtain an increase in the number of disks, and due to the insertion of new disks, mostly in contact with two other particles, the mean coordination number slightly decreases. Also, at inserting particles with small sizes, we are interested in the impact of the resulting frequency histograms.

Figure 6.18 presents two plots with the histograms with no refilling and the refilling with  $Br_{min} = 0.06$ . There is an increment of 650 particles ( $\approx 5\%$  of 11,000) that raises the bars at the left side of the histograms. Thus, the

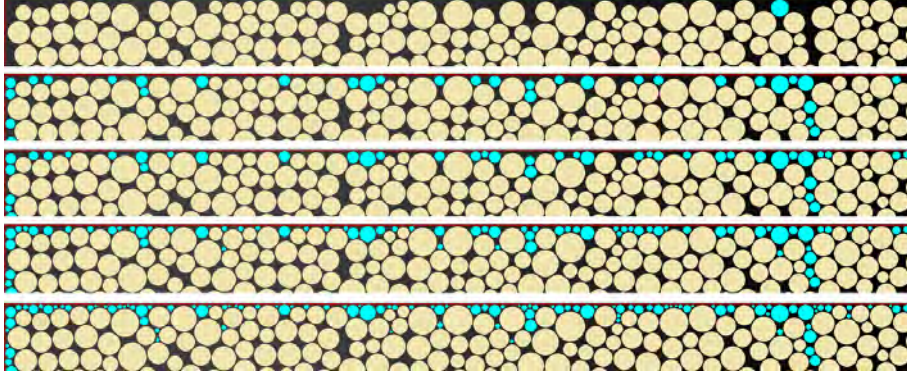


Figure 6.17: Border improvement. From top to bottom  $Br_{min} = 0.20, 0.10, 0.08, 0.06, 0.04$ . Added particles in cyan.

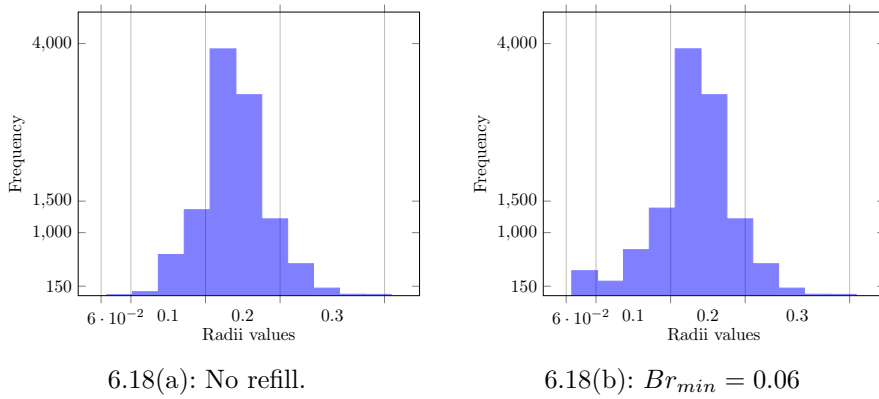


Figure 6.18: Experiment with  $Br_{min}$ .

application of this procedure will affect the desired PDF. On the other hand, it could improve the stability of the arrangement under physical forces.

## 6.8

### Stability tests

The new algorithm yields assemblies with more particles and higher contacts. To test the packs' stability under gravity, we employ Box2d<sup>1</sup>, a rigid collision simulator in 2D. The simulator uses the Verlet integration and an impulse-based contact model.

Our test scenario is the example of the circular container described in Section 6.5. We employ, as the initial position for the arrangements, three packs with different variants. The first uses the AF16 variant. The second employs the AFMeshOLSIS, and the third applies the boundary refilling procedure to the second pack, it uses the AFMeshOLSISBS variant with the minimum radius  $Br_{min} = 0.07$ .

We compare in Figure 6.19 and Figure 6.20 the initial and final positions

<sup>1</sup><https://box2d.org/>



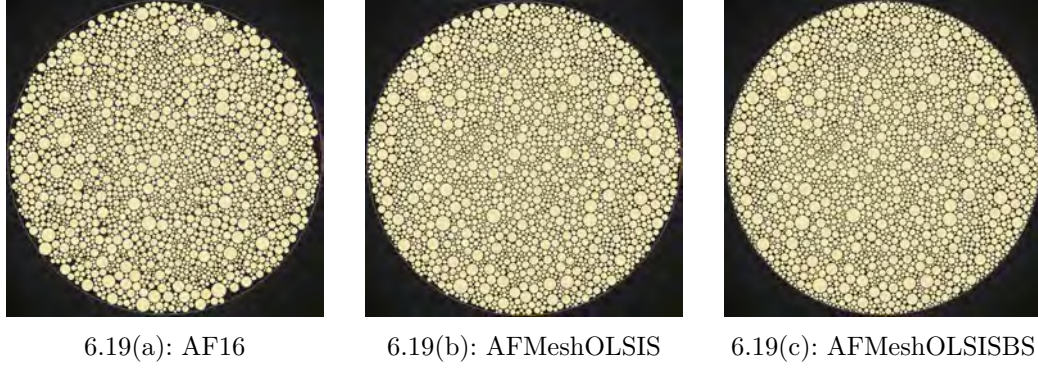


Figure 6.19: Initial packs for simulations

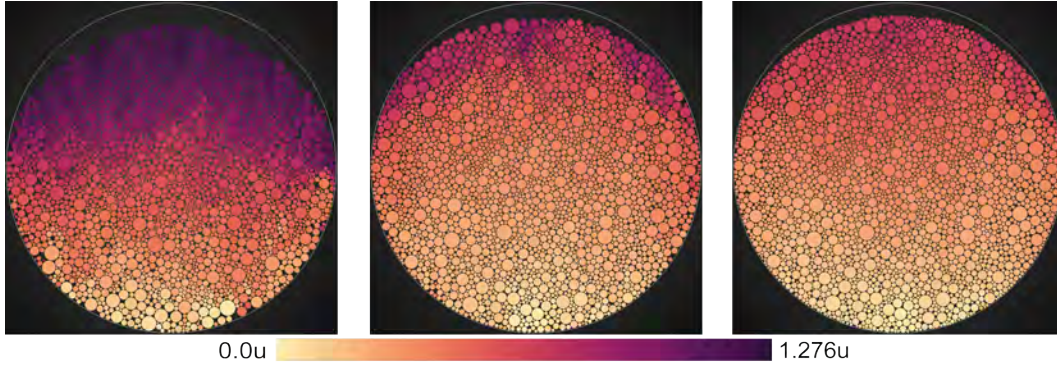


Figure 6.20: Final packs after simulations.

Left:  $\max(\Delta p_i) = 1.276u$  and  $\overline{\Delta p_i} = 0.673u$ ; Middle:  $\max(\Delta p_i) = 0.981u$  and  $\overline{\Delta p_i} = 0.383$ ; Right:  $\max(\Delta p_i) = 0.779u$  and  $\overline{\Delta p_i} = 0.365u$

before and after the simulations. More precisely, we measure the displacements of the particles. On the left side, we see that the AF16 variant creates fewer particles. The pack has a lower mean coordination number, so the displacements  $\Delta p$  are higher in comparison to the new variants. The AFMeshOLSIS creates more particles at the boundaries and at the interior of the arrangement. The simulation on the right side presents fewer displacements due to the refilling procedure at the pack's border.

## 6.9

### Benchmark with other approaches

In this section, we will compare the proposed method with other algorithms in the literature that define reproducible test scenarios. Despite having execution time in seconds from the other methods, it is somewhat unfair to compare our algorithm speed. The tests were carried out by different hardware and programming languages.

## 6.9.0.1

## Comparison 1 – Arbitrary PDF

[Benabbou *et al.* 2008] fills a square of dimensions 240x300 nm with a grain size distribution of radii ranging from 2 to 8 nm. Each range size has a specific frequency detailed in Table 6.6. It achieves 1,330 disks with a density of 0.80 and a computing time of less than 0.01s.

Radii (nm)	Frequencies (%)	Worst Freq (%)	Best Freq (%)
[2.0 - 2.5[	15.8	14.652	15.846
[2.5 - 3.0[	21.0	21.685	21.128
[3.0 - 3.5[	20.2	22.271	20.129
[3.5 - 4.0[	16.7	14.725	16.774
[4.0 - 4.5[	10.0	8.718	10.136
[4.5 - 5.0[	6.8	7.180	6.924
[5.0 - 5.5[	4.0	3.736	3.498
[5.5 - 6.0[	3.0	3.883	3.355
[6.0 - 6.5[	0.8	0.733	0.571
[6.5 - 7.0[	0.7	1.172	0.642
[7.0 - 7.5[	0.6	0.586	0.571
[7.5 - 8.0[	0.4	0.659	0.428

Table 6.6: Benabbou desired frequencies  
 Table 6.7: Our frequencies with the highest and lowest chi-square value.

We run 100 instances of this scenario with the relocation trigger ( $\phi_p \geq 0.222$  or  $\psi_p \geq 5$ ) for the OLS improvement. Our results are summarized in Table 6.8. On average, the new strategy produces a density of 0.82 with 1,406 particles in 0.22 seconds. Among all the instances, the smallest number of particles was 1,350, which is greater than the 1,330 obtained by [Benabbou *et al.* 2008]. Our density is always higher, and our lowest outcome was 0.816. Our execution time is slower.

Regarding the expected radius frequency, we summed the errors of each range per instance using the chi-square test. In the fourth column of Table 6.8, we can see that the chi-square value is small. The frequency distributions with the highest and lowest cumulative errors are shown in Table 6.7. An instance is illustrated in Figure 6.21.

	Density	Coord Number	F. Tensor ( $\beta_1, \beta_2$ )	Particles	Frequency errors	Time (s)
Min	0.8160	4.3098	0.49,0.49	1,350.000	0.002	0.1891
Max	0.8268	4.4397	0.50,0.50	1,453.000	0.017	0.2605
Mean	0.8210	4.3790	0.49,0.50	1,406.640	0.008	0.2228
Variance	4.e-6	0.0006	7.e-6,7.e-6	702.410	9.e-6	0.0001
S. deviation	2.e-3	0.0253	0.002,0.002	26.503	0.003	0.0117

Table 6.8: Comparison 1 – Summary of 100 instances.

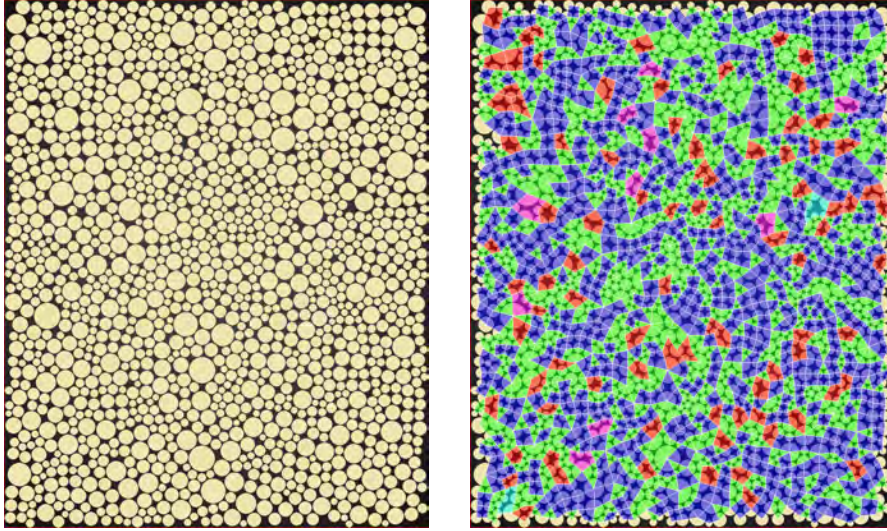


Figure 6.21: Comparison 1 – Pack and mesh  
( $\triangle$ ,  $\diamond$ ,  $\hexagon$ ,  $\square$ , He).

### 6.9.0.2

#### Comparison 2 – Ratio 1:7

[Bagi 2005] defines a rectangular container of 100x100 cm and fill it with radii of uniform distribution in the interval of [0.06 cm, 0.42 cm], a 1:7 ratio. We list in Table 6.9 some packing algorithms in the literature that test this scenario.

[Benabbou *et al.* 2009] achieves a density of 0.84 with around 39,000 particles in 0.9 seconds. [Liu *et al.* 2012] achieved 0.87 density in two phases. During the first phase, their algorithm reaches 0.84 in 18 seconds, and after a refilling phase, the density reaches 0.87 in 9 seconds. A total of 27 seconds. Their refilling does not strictly respect the given distribution.

	Machine	Particles	$\rho$	$Z$	Time(s)
Lubachevsky <i>et al.</i> 1991	IBM PC, 3GHz, Pentium 4	-	0.840	-	Hours
Bagi 2005	IBM PC, 3GHz, Pentium 4	39,219	0.824	3.98	196.0
Benabbou <i>et al.</i> 2009	Intel Core2, 2.8GHz	$\approx$ 39,000	0.840	-	0.9
Liu <i>et al.</i> 2012	Intel Core2, 2.53GHz	-	0.840	-	18.0
		-	0.870	-	27.0

Table 6.9: Comparison 2 – Literature results.

We run, in a first experiment, 100 instances for this scenario only with the relocation trigger ( $\phi_p \geq 0.222$  or  $\psi_p \geq 5$ ). Table 6.10 presents a summary of the results. The minimum achieved density is 0.850, while the maximum is 0.851. Among all the samples, 16 are below the range of [0.8512-s.deviation, 0.8512+s.deviation], 69 are inside that range, and 15 are above. The mean coordination number is, on average, 4.51. On the downside, the algorithm

took 7.5 seconds on average, [Benabbou *et al.* 2009] obtains a better time performance.

	Density	Coord Number	F. Tensor $(\beta_1, \beta_2)$	Particles	Frequency errors (%)	Time (s)
<b>Min</b>	0.8502	4.5046	0.49,0.49	39,116.000	0.0007	7.07
<b>Max</b>	0.8522	4.5283	0.50,0.50	40,040.000	0.0018	7.53
<b>Mean</b>	0.8512	4.5157	0.50,0.49	39,634.050	0.0012	7.21
<b>Variance</b>	1.e-7	2.e-5	2.e-7,2.e-7	24,050.767	5.e-8	0.006
<b>S. deviation</b>	0.0003	0.0048	5.e-4,5.e-4	155.083	0.0002	0.079

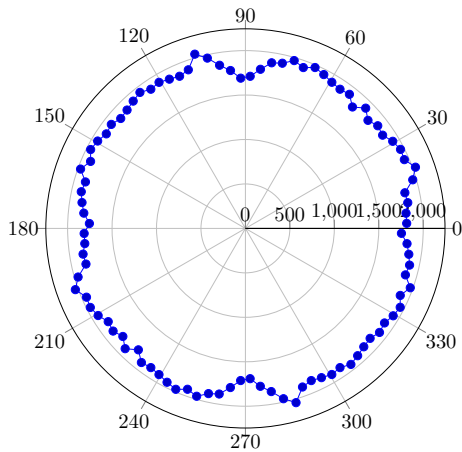
Table 6.10: Comparison 2 – AFMeshOLS – Summary of 100 instances.

In a second experiment, we maintain the OLS improvement with the same condition, and we also use the IS improvement to insert disks inside the polygons along with the generation. Table 6.11 shows that with the AFMeshOLSIS variant, the algorithm manages to insert  $\approx 600$  more particles, increasing the density and the coordination number.

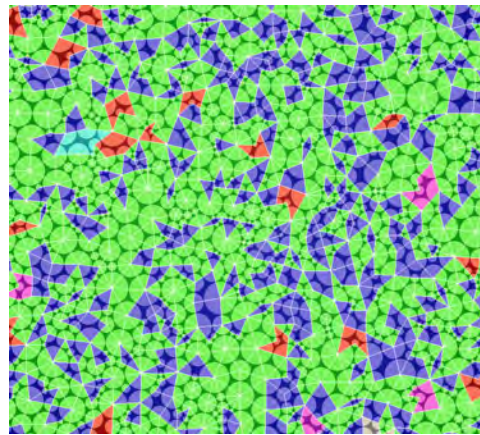
	Density	Coord Number	F. Tensor $(\beta_1, \beta_2)$	Particles	Frequency errors (%)	Time (s)
<b>Min</b>	0.8633	4.7738	0.49,0.49	39,850.000	0.0006	11.86
<b>Max</b>	0.8654	4.8055	0.50,0.50	40,638.000	0.0018	13.31
<b>Mean</b>	0.8645	4.7921	0.49,0.49	40,243.279	0.0012	12.28
<b>Variance</b>	1.e-7	3.e-5	1.e-7,1.e-7	23,674.701	5.e-8	0.03
<b>S. deviation</b>	0.0004	0.0054	4.e-4,4.e-4	153.865	0.0002	0.19

Table 6.11: Comparison 2 – AFMeshOLSIS – Summary of 100 instances.

We also compute the eigenvalues of the tensor. The mean eigenvalues in the variants are  $(\beta_1 = 0.50, \beta_2 = 0.49)$  and  $(\beta_1 = 0.49, \beta_2 = 0.49)$ . Figure 6.22(a) presents, for a single instance, the frequency distribution of contact orientations. The structures do not have a preferred contact direction and are isotropic.



6.22(a): Contact orientation.



6.22(b): Small pack region ( $\Delta$  ■,  $\Diamond$  ■,  $\square$  ■,  $\circ$  ■, He ■).

Figure 6.22: Comparison 2 – AFMeshOLSIS instance N°4



Table 6.12 shows a counter of occurrences per polygon for the fourth instance in Table 6.11. Note the high number of triangles and quads.

Polygon Size	3	4	5	6	7	8	9
Count	39,950	16,410	1,502	417	87	18	4

Table 6.12: Comparison 2 – Polygon count of AFMeshOLSIS instance N°4.

We also collect additional data regarding the front removal logic and the OLS improvement in Table 6.13. With only the OLS, the method executes approximately 7,800 relocations per run, and approximately 2,600 are rejected, returning the removed particles to their original configuration. With the IS, the number of relocations is reduced by  $\approx 1,000$ . The reduction is expected because the inner insertion is tested before adding new particles to the “outer loop”. With fewer number of “bad polygons” at the mesh borders, the relocation trigger declines.

	AFMeshOLS				AFMeshOLSIS			
	Relocations	Rollbacks	qnr	qpr	Relocations	Rollbacks	qnr	qpr
<b>Min</b>	7,706.0	2,367.0	0.0	0.0	6,590.0	1,938.0	0.00	0.00
<b>Max</b>	8,011.0	2,588.0	10.0	11.0	6,909.0	2,155.0	10.00	11.00
<b>Mean</b>	7,828.9	2,484.9	6.0	4.9	6,752.9	2,047.8	5.71	5.21

Table 6.13: Comparison 2 – Relocations and front logic output data.

In both experiments, in  $\approx 70\%$  of the cases, the algorithm improves the density. On the other hand,  $\approx 30\%$  of the cases, the algorithm wastes time trying to find new locations for the particles

Additionally, we count the remaining elements in the queues of newly rejected radii (qnr) and previously rejected radii (qpr) at the end of the runs. Those queues are minimal compared to the total inserted particles; thus, the algorithm followed the specified PDF.

### 6.9.0.3

#### Comparison 3 - Densification

[Bagi 2005] employs the output of the previous experiment and use it to test its densification strategy. It consists of detecting pairs of contacts between disks and trying to insert a third particle in contact, without generating collisions. The new particle size must be within a given range size defined by the user. Bagi reused the [0.06 cm, 0.42 cm] range. The final PDF of the pack will change. Filling the voids will favor the generation of smaller particles. Some methods in the literature also use this scenario. Since all of them employ different particle generation approaches, the test computes the average radius in the pack  $r_{avg}$  as a comparison property.

	Machine	Particles	$r_{avg}$	$\rho$	$Z$	Time(s)
<b>PFC2D*</b>	-	-	0.229	0.857	4.29	64h
<b>Bagi 2005</b>	IBM PC, 3GHz, Pentium 4	56,213	0.221	0.858	3.98	388
<b>Labra et al. 2009</b>	Pentium 4, 3GHz	56,084	0.223	0.903	5.97	118
<b>Cui et al. 2003**#</b>	Intel Core i7-8850H	56,241	0.214	0.768	1.01	52
<b>Zhang et al. 2020<sup>#</sup> Alg<sub>1</sub></b>	Intel Core i7-8850H	56,753	0.221	0.849	3.99	80
<b>Zhang et al. 2020<sup>#</sup> Alg<sub>2</sub></b>	Intel Core i7-8850H	55,614	0.219	0.803	4.27	181

\* Computed by Labra. \*\*Computed by Zhang. # Unoptimized MATLAB implementation

Table 6.14: Comparison 3 – Literature results.

We see that the final density of Bagi in Table 6.14 is comparable to our density using the AFMeshOLS variant. For that reason, we will generate a pack to be densified without any improvement procedure to start with a lower density, which is the AFMesh variant.

We perform a similar densification procedure as in [Bagi 2005]. In our case, we repopulate the domain in two steps. The first step performs a particle insertion similar to the IS strategy. It creates a set of holes using the final polygons at the mesh and then places particles, between the radii range, inside them. The insertions trigger the split of holes, and new polygons will be registered in the set. The procedure ends as soon as the largest hole is smaller than the smallest allowed particle. The second step performs the BS improvement that inserts new disks in contact with the border particles and the container.

For our runs, we explore the  $Br_{min}$  minimum radius effect on the densification outputs.

$Br_{min}$	Density	Coord Number	F. Tensor $(\beta_1, \beta_2)$	Particles	$r_{avg}$	Time(s)
-	0.828	4.194	(0.492, 0.507)	38,610	0.239	1.06
0.080	0.875	4.691	(0.497, 0.502)	52,186	0.204	1.03
0.075	0.881	4.761	(0.496, 0.503)	55,083	0.197	1.05
0.070	0.887	4.837	(0.498, 0.501)	58,492	0.190	1.06
0.065	0.892	4.912	(0.498, 0.501)	62,466	0.182	1.07
0.060	0.898	4.991	(0.498, 0.501)	67,167	0.174	1.07

Table 6.15: Comparison 3 – Densification results.

The first line in Table 6.15 corresponds to the initial pack. Note that the initial  $r_{avg}$  is close to the Bagi's average radii after the densification. This gives us the idea that our densification will lead to the insertions of a huge number of small particles. Table 6.15 presents the outputs for five values of  $Br_{min}$ . We see that reaching a particle size of  $\approx 56,000$  will yield an  $r_{avg}$  of  $\approx 0.19$ . Figure 6.23 shows the left corner of one pack and the corresponding polygonal mesh.

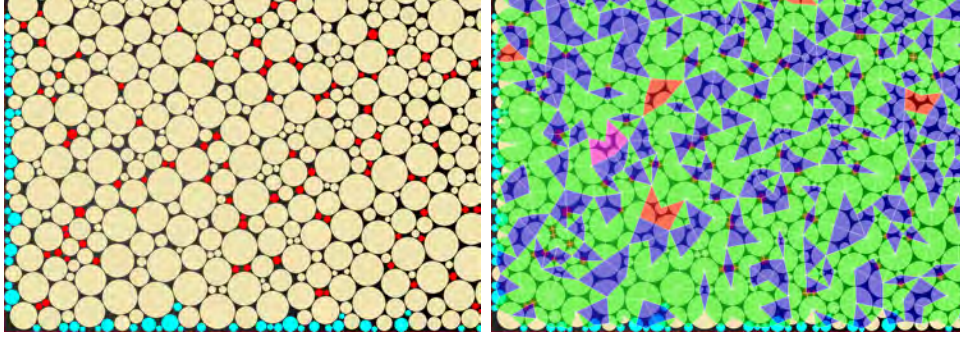


Figure 6.23: AFMesh pack densification with IS (post process) + BS using  $Br_{min} = 0.065$ . IS and BS particles are in red and cyan respectively.

( $\triangle$ ,  $\diamond$ ,  $\square$ ,  $\square$ ).

## 6.10

### Complex geometries

This section shows that our method is not restricted to simple containers but can handle complex geometries.

#### 6.10.1

##### Slope

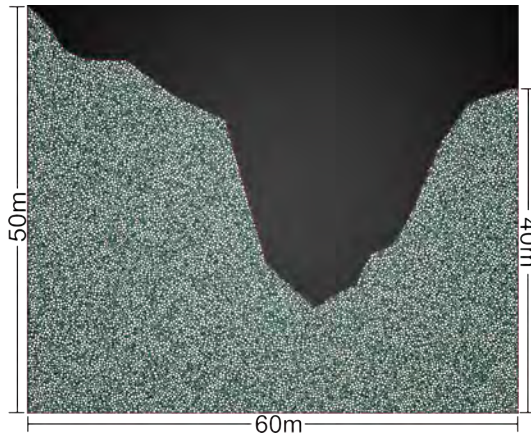
As a first example, we use a model defined by [Zhang *et al.* 2020]. It represents a slope, a common problem in civil engineering. Figure 6.24(a) displays the model and its dimensions. The work does not prescribe a PDF because Zhang used a mesh-based method to generate particles on triangular meshes representing the container. The PDF on this kind of approach depends on the quality of the given triangulation. Nevertheless, it describes the resulting particle range  $[0.04, 0.20]$ .

Here, we perform a packing using this range size with a uniform distribution. The packing will use the AFMeshOLSISBS variant. We apply the BS improvement to increase contacts with the boundary with  $Br_{min} = 0.04$ .

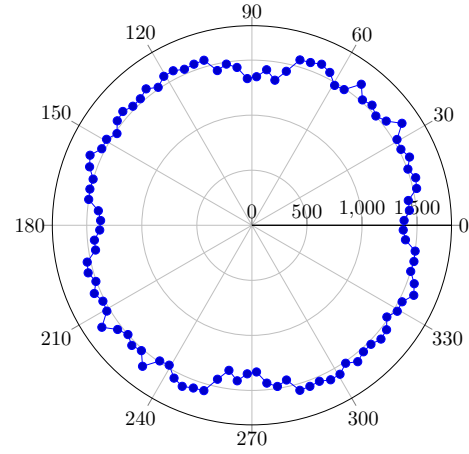
Figure 6.24(b) plots the polar frequencies of the contacts between the particles. This pack is composed of 32,867 disks, reaching a density of 0.86 with a mean coordination of 4.67 in 6.7 seconds.

Figure 6.25 depicts a small portion of the model and the respective polygonal mesh. Note the particles generated by the BS improvement in cyan. Also, to have a closer look at the radius distribution, we use in Figure 6.26(a) a color palette according to the radii size of the particles.

Finally, we plot the radius frequencies of the packs before and after the BS improvement. Note the small increase of the smaller particles in Figure 6.26(b).



6.24(a): Model and pack.



6.24(b): Contact orientations.

Figure 6.24: Zhang's slope model.

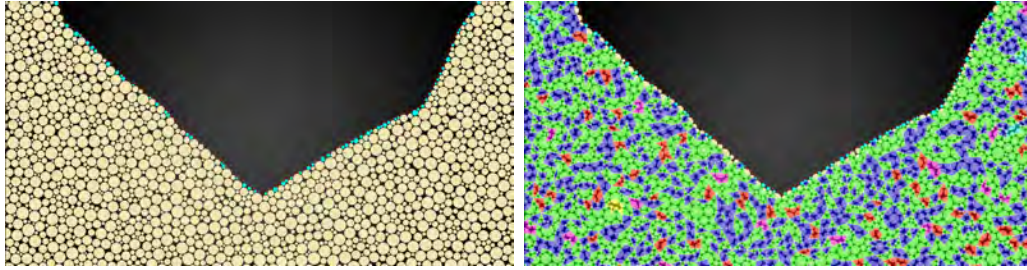
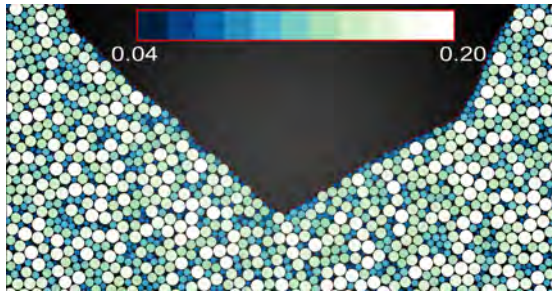
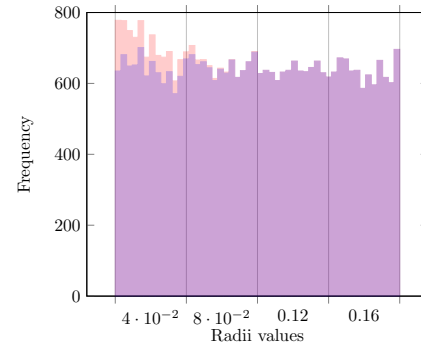


Figure 6.25: Zoom at the slope model. Pack and mesh.

( $\triangle$  ■,  $\diamond$  ■,  $\pentagon$  ■,  $\hexagon$  ■, He ■,  $\circ$  ■).



6.26(a): Colors by radius size.



6.26(b): Radius frequencies.

Figure 6.26: Slope model – Radius frequencies.

## 6.10.2

### Handling domains with holes

We explore in this section, the generation of packs for domains with holes. Figure 6.27 presents a complex test model, the dragon silhouette. The model exhibits a hole in the region A2.

When packing a domain with holes, there is a moment when two sets



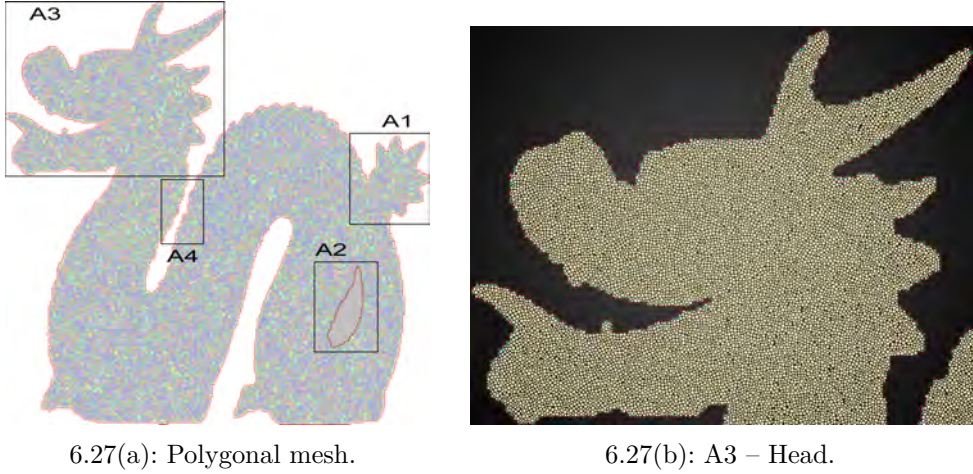


Figure 6.27: Dragon model.

of advancing fronts merge. In Figure 6.28, we present a small example where two separate front lines merge after a particle insertion. The “outer loop” now connects the two front lines. Figure 6.28(c) shows the resulting polygon surrounding the hole. These polygons will not be considered for the polygonal mesh frequency or for the interior strategy (IS); it would be too expensive to compute the Soddy circles of the triplets. Empty spaces of the polygon will be filled by the fronts nearby in future iterations. Optionally, we could use the boundary strategy to increase the density around the hole.

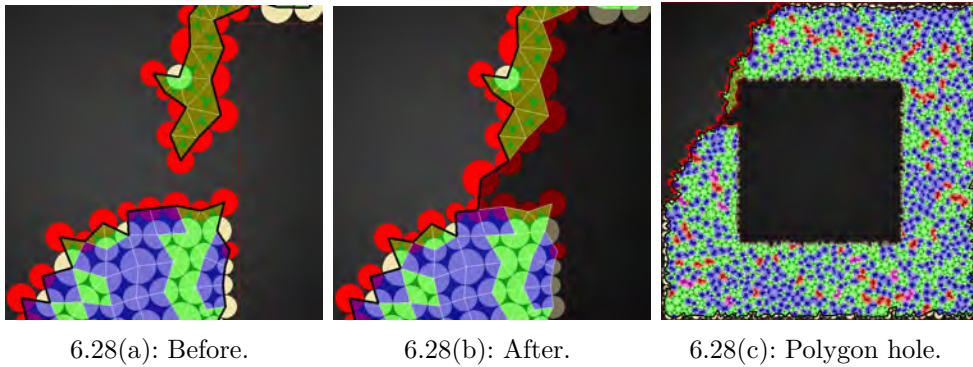


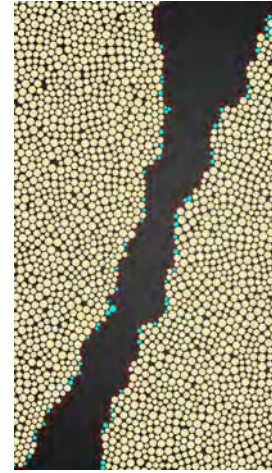
Figure 6.28: Merging two fronts. Active fronts in red.  
 $(\triangle \blacksquare, \diamond \blacksquare, \triangleleft \blacksquare, \triangleleft \blacksquare)$ .

Testing if every polygon is related to a domain hole may be expensive, so we only perform that verification for the polygons created by new particles placed within a  $r_{max}$  distance from the boundary. Figure 6.29 focuses on three areas of the dragon. Note the small particles in cyan added with the BS procedure.

Figure 6.30 introduces a more complex model, the octopus. We consider this model a good test because of two points:



6.29(a): A1 – Tail.

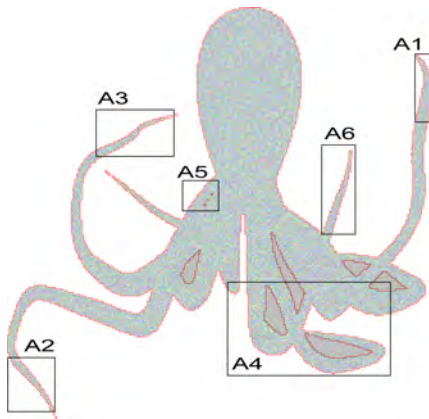
6.29(b): A2 – Hole do-  
main.

6.29(c): A4 – Neck.

Figure 6.29: Dragon detail. Added particle with the BS improvement in cyan.

- Allow us to observe the algorithm's behavior handling very narrow regions. Especially for the tentacles.
- It has several holes of different dimensions in the middle of the model. It allows us to verify our ADF implementation. The tiny holes in the region A5 are the product of a discretization error during the extraction of the model contour from the triangle mesh. Still, our method respects the holes.

Figure 6.31 and Figure 6.32 show that the method can spread through narrow regions.



6.30(a): Polygonal mesh.



6.30(b): A4 – Holes.

Figure 6.30: Octopus model

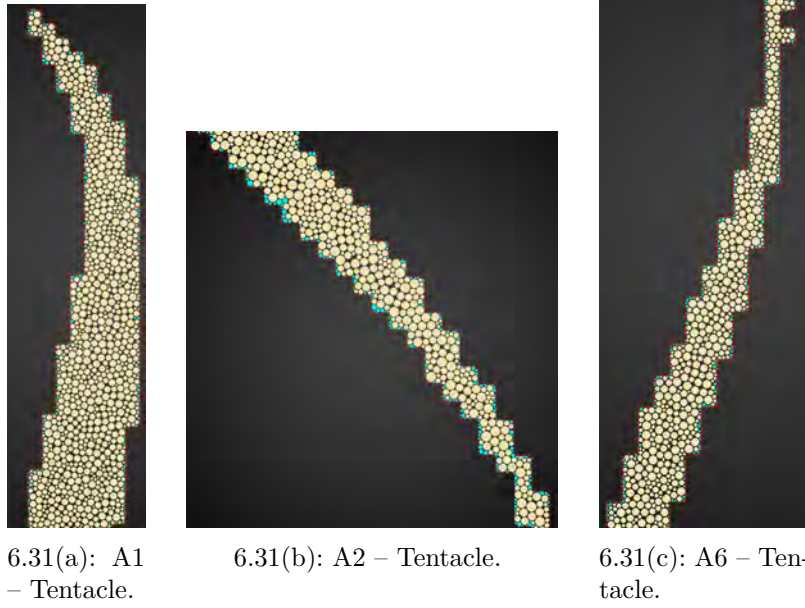


Figure 6.31: Octopus detail.

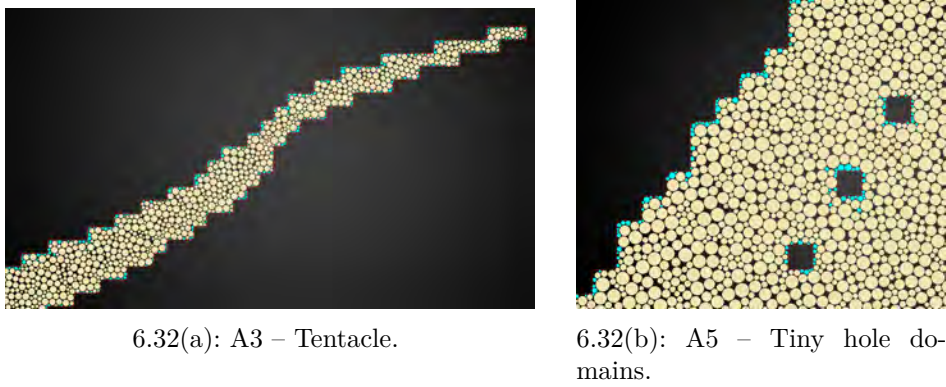


Figure 6.32: Octopus detail (Continuation).

### 6.10.3 Tomography slices

Among the literature, [Skarżyński *et al.* 2015], [Nitka *et al.* 2018] and [Nitka *et al.* 2020] combine DEM simulations with X-ray micro-CT images to model concrete fractures in 4-phase material images. All the phases, except for the void, are populated with particles.

Here, we will treat 2-phase micro-CT images, and create packs in the solid phase. Our test models were created with an iso-contour extraction algorithm based on a dual contouring approach [Lobello *et al.* 2014] on sections of volumetric images available at the web site of the Department of Earth Science

and Engineering of the Imperial College London<sup>2</sup>.

Table 6.16 enumerates the test image names, resolutions and porosity. It also details the corresponding slice of the image and the number of segments of the resulting contour mesh.

Name	Volume	Image data		Extraction data	
		Resolution	Porosity	XY-Section N°	N°Segments
Berea sandstone	400 <sup>3</sup>	5.3450 $\mu m$	19.6%	200	7,123
Carbonate C1	400 <sup>3</sup>	2.8500 $\mu m$	23.3%	200	5,051
Ketton carbonate	1000 <sup>3</sup>	3.0035 $\mu m$	13.9%	500	6,820
Sand pack (LV60C)	450 <sup>3</sup>	10.0020 $\mu m$	37.2%	100	21,029

Table 6.16: Dual contouring inputs and outputs.

The models of microstructures are complex and heterogeneous. Our method identifies the connected components of segments surrounded by void regions, and for each one, we perform the following:

- Compute the bounding box of the segments.
- Compute the signed distance field within the bounding box.
- Define the particle seeds of the component. A random procedure determines three valid positions and radii for the disks. Only if the components' area is too small to fit three particles or the random procedure reaches a certain counter, the seed generation fails.
- Generate the pack of the component if the seed generation is successful.

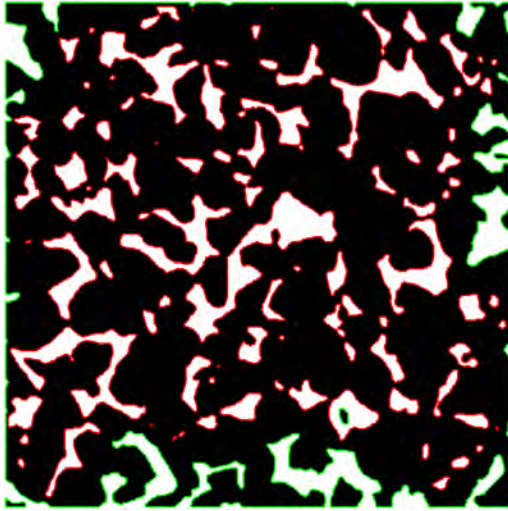
Each component will have a pack, a polygonal mesh, and an “outer loop”. The ADF structure and the grid are local to each component and can be deallocated from memory after the generation.

Name	Components	Uniform Dist.	Particles	Density	Z	Time(s)
Berea sandstone	6	[0.10-0.30]	771,970	0.83	4.38	206.0
Carbonate C1	15	[0.10-0.30]	750,320	0.84	4.40	169.0
Ketton carbonate	8	[0.20-0.50]	1,702,998	0.83	4.39	399.0
Sand pack (LV60C)	152	[0.10-0.20]	1,439,529	0.81	4.32	290.0

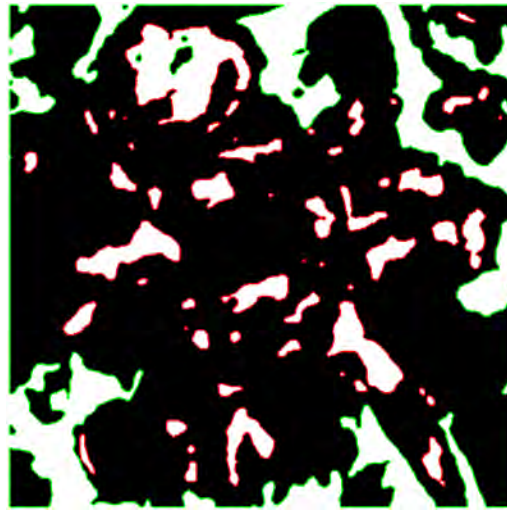
Table 6.17: Summary of rock image packing.

<sup>2</sup><https://www.imperial.ac.uk/earth-science/research/research-groups/perm/research/pore-scale-modelling/micro-ct-images-and-networks/>



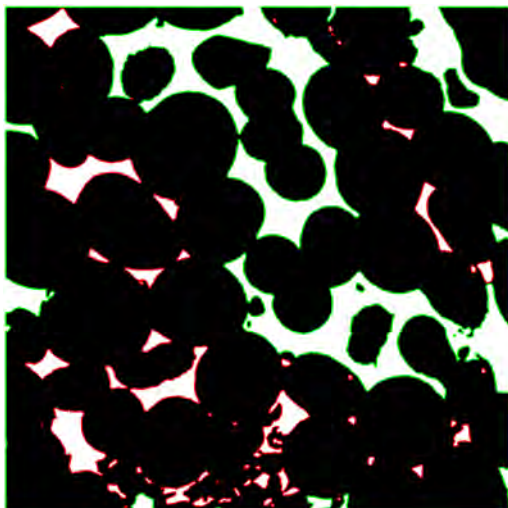


6.33(a): Berea sandstone.

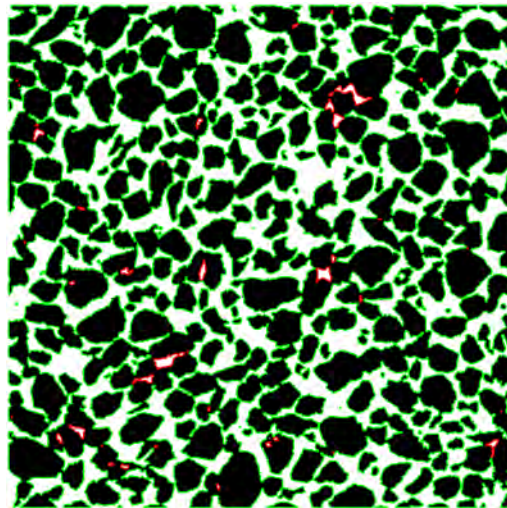


6.33(b): Carbonate C1.

Figure 6.33: Extracted models.

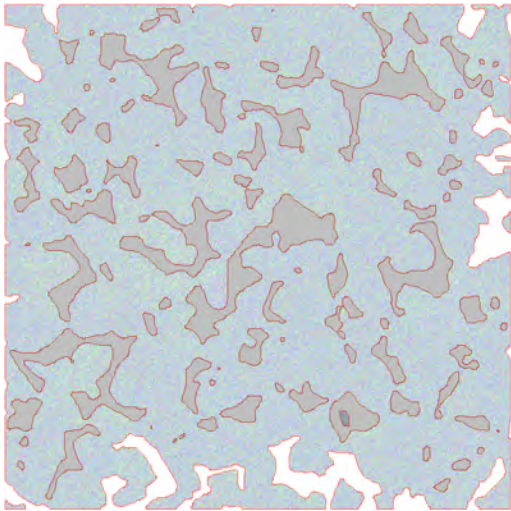


6.34(a): Ketton carbonate.

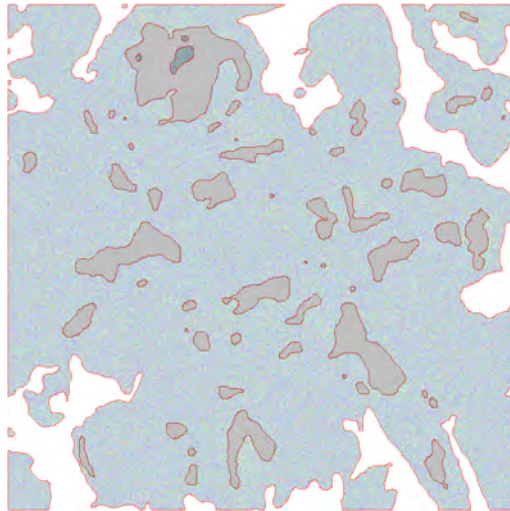


6.34(b): Sand pack (LV60C).

Figure 6.34: Extracted models (Continuation).

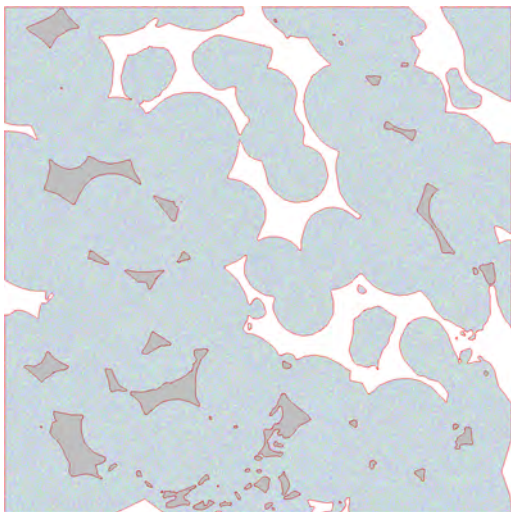


6.35(a): Berea sandstone.

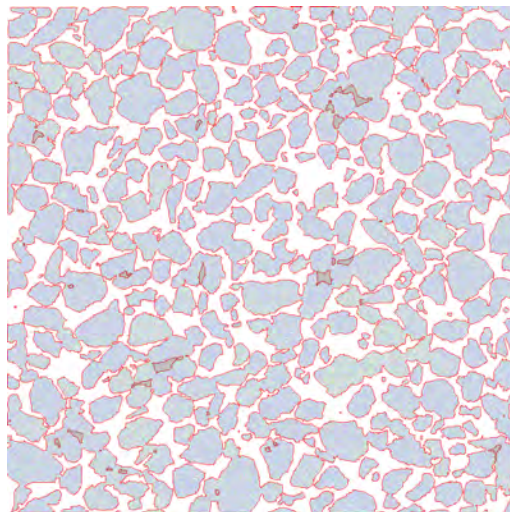


6.35(b): Carbonate C1.

Figure 6.35: Pack on images.



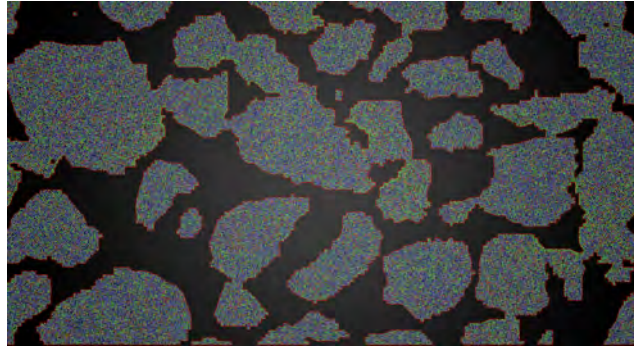
6.36(a): Ketton carbonate.



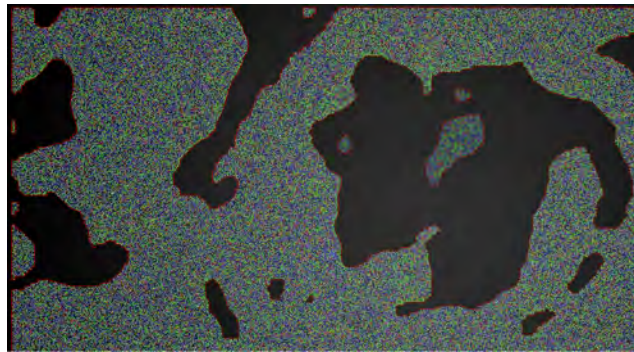
6.36(b): Sand pack (LV60C).

Figure 6.36: Pack on images (Continuation).



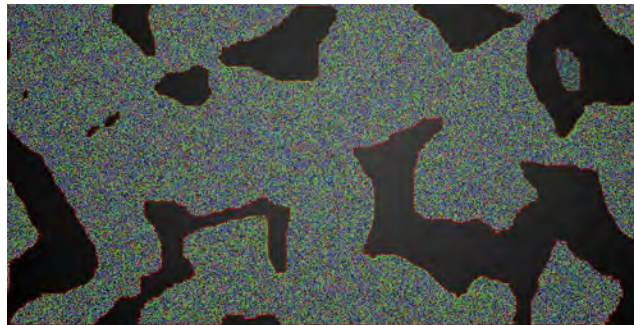


6.37(a): Sand pack (LV60C) right bottom region.

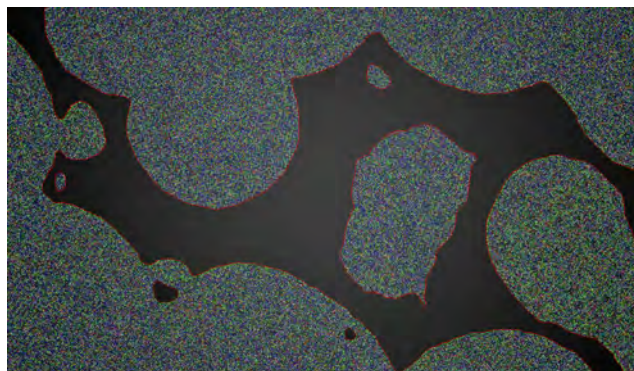


6.37(b): Carbonate left upper corner.

Figure 6.37: LV60C and carbonate pack zooms.



6.38(a): Berea sandstone bottom corner.



6.38(b): Ketton carbonate middle region.

Figure 6.38: Berea and ketton pack zooms.

## 7

## Conclusions

This work presents a new packing algorithm to create dense random arrangements of disk particles as an initial step for the discrete element method (DEM). The algorithm yielded excellent geometric properties while maintaining good computational time.

The optimal density achieved by hexagonal packings ( $\approx 0.9069$ ) is theoretically obtained for infinite planes. Using a domain restriction adds an edge effect that is minimized when the container is scaled up to a certain size or when the particle size is reduced. The proposed algorithm creates regular arrangements for the monodisperse case. Given a circular container 40 times the size of the disk radius our method reaches a density of 0.87.

For a mixture of disks with different sizes, our method yields random arrangements following a given particle distribution function (PDF) or an arbitrary radius frequency histogram.

For the bidisperse case, the algorithm obtained a density of 0.822 for the 1:1.4 ratio between the smallest and the biggest radius. According to physical simulations in [O'Hern *et al.* 2002, Donev *et al.* 2004, Henkes *et al.* 2007, Meyer *et al.* 2010] such packings report a maximum density of  $\rho \approx 0.84$ .

For uniform distributions we reached densities of 0.825, 0.837, 0.848, 0.853 and 0.853 for the ratios of 1:2, 1:3, 1:4, 1:5 and 1:6 respectively. As a reference for comparison with other geometric algorithms, [Bagi 2005] obtained a density of  $\approx 0.816$  for uniform packs with a 1:2 ratio. [Liu *et al.* 2012] obtained density values of  $\approx 0.805$ ,  $\approx 0.818$ ,  $\approx 0.826$  and  $\approx 0.834$  for uniform packs for the ratios of 1:2, 1:3, 1:4 and 1:5.

The construction of a polygonal mesh on top of the pack allowed the definition of heuristics to determine the next positions of the incoming radius. The heuristics favored the insertion of particles inside adjacent polygons to the current front, enforced the contacts between the new particle and particles in the neighborhood and reinforced the generation of compact arrangements. The new heuristics proved to be more effective than the heuristic we proposed in previous work. Additionally, two new parameters were introduced to enhance the generation of particles near the domain boundaries.

A new strategy that aims to locally rearrange a small subset of particles

on the pack's borders is proposed. The maximum porosity and the maximum number of sides among the new polygons are employed as conditions to trigger the relocations after a disk insertion. This mechanism improved the density and coordination number of the arrangements. However, the local packing algorithm is not 100% effective. Experiments reveal that  $\approx 30\%$  of the relocations are reverted because they cannot find a better configuration for the removed disks. From a combinatorial viewpoint, the destruction and reconstruction of portions of the pack classify this as a non-constructive strategy. In consequence, the variants AFMeshOLS and AFMeshOLSIS are semi-constructive algorithms.

Another strategy that permits the insertion of particles in the void space all over the pack is introduced. We developed a greedy algorithm that uses the mesh to identify the particles that can be placed inside the polygons during the packing generation. This mechanism does not affect the PDF of the radii pack and increases the density and coordination number, mainly in heterogeneous arrangements.

A refilling procedure was offered to improve the packing near the boundaries as a post-packing optional step. It was used a minimum radius as a stop condition to add more particles at empty regions near the borders of the domain. This refilling will increase the frequencies of the smaller particles. Nevertheless, the increment is minimal in comparison to the total number of particles in the arrangements.

The polygonal mesh structure granted complete control of the void regions of the packs. The set of holes, created in the IS strategy, supported further densification by inserting more particles inside the polygons.

Through several experiments and runs of the algorithm and its variants for the proposed scenarios, our method's time complexity presented a linear behavior on the number of particles. The AFMesh and AFMeshIS variants were the fastest while the AFMeshOLS and AFMeshOLSIS took more time to produce the arrangements. The AFMeshOLSIS, among the variants, yielded the highest density and mean coordination number.

In comparison to the AF16, the new variants created more triangles in the mesh. As stated in [Papadopoulos *et al.* 2018], triangles tend to be stabilizing structures than can maintain rigidity under applied forces. The new variants produced more stable arrangements than the AF16 according to rigid body simulations only with a gravitational force. This was further enhanced by employing the boundary strategy to increment the contacts with the boundary domain. Still, our solution is not completely physically stable.

Finally, our method is capable of handling complex containers such as

rock structures from micro-CT images. Our adaptive distance field implementation handled domain holes of different dimensions, even tiny holes generated by discretization errors during the contour extraction of triangle meshes. Polygons enclosing holes were identified to apply the boundary improvement on them to increase contacts with the frontiers. Each component of the domain was packed independently for the other components. Our method managed to expand the particle generation even through narrow regions of the containers.

## 7.1

### Future work

It is missing a smarter trigger for the particles in the “outer loop” improvement. The relocation algorithm can improve the density in 70% of the cases. The remaining 30% had to restore the original configuration of the removed particles. Another alternative is to explore new ways to relocate the particles. [Akeb 2014] explores other search space algorithms to pack spheres. More advanced strategies will have an impact on time consumption.

The refilling problem at the domain boundaries is not trivial. A more sophisticated and global process needs to be implemented to achieve better results. This includes not only the radii’s adjustment but also the modification of the locations of particles near the frontiers [Labra *et al.* 2009].

Further study is needed to adapt the algorithm to handle the porosity as a user parameter. Having the polygonal mesh provides awareness of the whole network connections in the pack. So, it is possible to remove some particles, starting with the ones with fewer contacts, without affecting the stability, until reaching the desired porosity. The removal strategy is challenging for highly heterogeneous arrangements because removing big particles will significantly impact the local contact network.

For complex geometries with a set of connected components, the execution time could be improved with a parallel execution of the pack generation.

An adaptation of the generation algorithm could be explored to obtain anisotropic packs following a given tensor field.

A variation of the current algorithm is the sphere packing on top of planes as an application to create the initial pack for 3D methods that use an inward packing generation for known containers such as parallelepipeds.

A natural evolution of the packing method is the implementation of the 3D version. Instead of a polygonal mesh, it will be necessary to study the construction of a polyhedral mesh. The biggest challenge will be to define how to close polyhedrons upon spherical particle insertions. Note that in 2D, disk particles’ insertion always closes polygons, while in 3D, this will not hold.

Besides, in 3D, all the void spaces are interconnected, there is no closed void region. The method must be able to handle a network of interconnected void spaces. A crucial step in solving this problem is defining the difference between a void body and a void throat. This is a problem that is still subject of discussion [Van der Linden *et al.* 2016].

## Bibliography

- [Akeb 2014] AKEB, H.. A look-forward heuristic for packing spheres into a three-dimensional bin. In: 2014 FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS, p. 397–404. IEEE, 2014.
- [Ardanza *et al.* 2014] ARDANZA-TREVIJANO, S.; ZURIGUEL, I.; ARÉVALO, R. ; MAZA, D.. Topological analysis of tapped granular media using persistent homology. *Physical Review E*, 89(5):052212, 2014.
- [Aste *et al.* 1992] NOLAN, G.; KAVANAGH, P.. Computer simulation of random packing of hard spheres. *Powder technology*, 72(2):149–155, 1992.
- [Aste *et al.* 2005] ASTE, T.. Variations around disordered close packing. *Journal of Physics: Condensed Matter*, 17(24):S2361, 2005.
- [Atkinson *et al.* 2014] ATKINSON, S.; STILLINGER, F. H. ; TORQUATO, S.. Existence of isostatic, maximally random jammed monodisperse hard-disk packings. *Proceedings of the National Academy of Sciences*, 111(52):18436–18441, 2014.
- [Azevedo *et al.* 2013] AZEVEDO, N. M.; LEMOS, J. V.. A 3d generalized rigid particle contact model for rock fracture. *Engineering Computations: Int J for Computer-Aided Engineering*, 30(2):277–300, 2013.
- [Bagi 1993] BAGI, K.. A quasi-static numerical model for micro-level analysis of granular assemblies. *Mechanics of materials*, 16(1-2):101–110, 1993.
- [Bagi 2005] BAGI, K.. An algorithm to generate random dense arrangements for discrete element simulations of granular assemblies. *Granular Matter*, 7(1):31–43, 2005.
- [Barber *et al.* 1996] BARBER, C. B.; DOBKIN, D. P. ; HUHDANPAA, H.. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.



- [Benabbou *et al.* 2008] BENABBOU, A.; BOROUCHAKI, H.; LAUG, P. ; LU, J.. **Sphere packing and applications to granular structure modeling.** In: PROCEEDINGS OF THE 17TH INTERNATIONAL MESHING ROUNDTABLE, p. 1–18. Springer, 2008.
- [Benabbou *et al.* 2009] BENABBOU, A.; BOROUCHAKI, H.; LAUG, P. ; LU, J.. **Geometrical modeling of granular structures in two and three dimensions. application to nanostructures.** International Journal for Numerical Methods in Engineering, 80(4):425–454, 2009.
- [Berryman 1983] BERRYMAN, J. G.. **Random close packing of hard spheres and disks.** Phys. Rev. A, 27:1053–1061, Feb 1983.
- [Bithell *et al.* 2014] BITHELL, M.; RICHARDS, K. S. ; BITHELL, E. G.. **Simulation of scree-slope dynamics: investigating the distribution of debris avalanche events in an idealized two-dimensional model.** Earth Surface Processes and Landforms, 39(12):1601–1610, 2014.
- [Bonneau *et al.* 2020] BONNEAU, F.; SCHOLTES, L. ; RAMBURE, H.. **An algorithm for generating mechanically sound sphere packings in geological models.** Computational Particle Mechanics, p. 1–14, 2020.
- [Borkovec *et al.* 1994] BORKOVEC, M.; DE PARIS, W. ; PEIKERT, R.. **The fractal dimension of the apollonian sphere packing.** Fractals, 2(04):521–526, 1994.
- [Campello *et al.* 2016] CAMPELLO, E.; CASSARES, K. R.. **Rapid generation of particle packs at high packing ratios for dem simulations of granular compacts.** Latin American Journal of Solids and Structures, 13(1):23–50, 2016.
- [Chang *et al.* 2010] CHANG, H.-C.; WANG, L.-C.. **A simple proof of thue’s theorem on circle packing.** arXiv preprint arXiv:1009.4322, 2010.
- [Cohen *et al.* 1995] COHEN, J. D.; LIN, M. C.; MANOCHA, D. ; PONAMGI, M.. **I-collide: An interactive and exact collision detection system for large-scale environments.** In: PROCEEDINGS OF THE 1995 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS, p. 189–ff, 1995.
- [Cui *et al.* 2003] CUI, L.; O’SULLIVAN, C.. **Analysis of a triangulation based approach for specimen generation for discrete element simulations.** Granular Matter, 5(3):135–145, 2003.

- [Cundall *et al.* 1979] CUNDALL, P. A.; STRACK, O. D.. **A discrete numerical model for granular assemblies.** *geotechnique*, 29(1):47–65, 1979.
- [Devroye *et al.* 1986] DEVROYE, L.. **Sample-based non-uniform random variate generation.** In: PROCEEDINGS OF THE 18TH CONFERENCE ON WINTER SIMULATION, p. 260–265. ACM, 1986.
- [Donev *et al.* 2004] DONEV, A.; TORQUATO, S.; STILLINGER, F. H. ; CONNELLY, R.. **Jamming in hard sphere and disk packings.** *Journal of applied physics*, 95(3):989–999, 2004.
- [Dong *et al.* 2020] DONG, Q.; WANG, Y. ; FENG, D.. **An algorithm to generate dense and stable particle assemblies for 2d dem simulation.** *Engineering Analysis with Boundary Elements*, 114:127–135, 2020.
- [Ebeida *et al.* 2016] EBEIDA, M. S.; RUSHDI, A. A.; AWAD, M. A.; MAHMOUD, A. H.; YAN, D.-M.; ENGLISH, S. A.; OWENS, J. D.; BAJAJ, C. L. ; MITCHELL, S. A.. **Disk density tuning of a maximal random packing.** In: COMPUTER GRAPHICS FORUM, volumen 35, p. 259–269. Wiley Online Library, 2016.
- [Eitz *et al.* 2007] EITZ, M.; LIXU, G.. **Hierarchical spatial hashing for real-time collision detection.** In: IEEE INTERNATIONAL CONFERENCE ON SHAPE MODELING AND APPLICATIONS 2007 (SMI'07), p. 61–70. IEEE, 2007.
- [Ericson 2004] ERICSON, C.. **Real-Time Collision Detection.** The Morgan Kaufmann Series in Interactive 3D Technology. Elsevier Science, 2004.
- [Feng *et al.* 2003] FENG, Y.; HAN, K. ; OWEN, D.. **Filling domains with disks: an advancing front approach.** *International journal for numerical methods in engineering*, 56(5):699–713, 2003.
- [Ferreira 2009] PINTO, A. L. F.. **Algoritmo para geração de arranjos de partículas para utilização no método dos elementos discretos.** Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2009.
- [Frery *et al.* 2012] FRERY, A. C.; RIVAROLA-DUARTE, L.; RAMOS, V. C. L.; RAMOS, A. S. ; LIRA, W. W. M.. **Stochastic particle packing with specified granulometry and porosity.** *Granular Matter*, 14(1):27–36, 2012.
- [Friskén *et al.* 2000] FRISKÉN, S. F.; PERRY, R. N.; ROCKWOOD, A. P. ; JONES, T. R.. **Adaptively sampled distance fields: A general**

- representation of shape for computer graphics. In: PROCEEDINGS OF THE 27TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, p. 249–254, 2000.
- [Gao *et al.* 2018] GAO, K.; EUSER, B. J.; ROUGIER, E.; GUYER, R. A.; LEI, Z.; KNIGHT, E. E.; CARMELIET, J. ; JOHNSON, P. A.. **Modeling of stick-slip behavior in sheared granular fault gouge using the combined finite-discrete element method.** Journal of Geophysical Research: Solid Earth, 123(7):5774–5792, 2018.
- [Garcia 2018] GARCIA, F. E. T.. **Discrete element analysis of surface fault rupture through granular media.** PhD thesis, UC Berkeley, 2018.
- [Gardiner *et al.* 2015] GARDINER, B. S.; WONG, K. K.; JOLDES, G. R.; RICH, A. J.; TAN, C. W.; BURGESS, A. W. ; SMITH, D. W.. **Discrete element framework for modelling extracellular matrix, deformable cells and subcellular components.** PLoS computational biology, 11(10), 2015.
- [Ghasemi *et al.* 2020] GHASEMI, M.; FALAHATGAR, S.. **Discrete element simulation of damage evolution in coatings.** Granular Matter, 22(2):1–16, 2020.
- [Han *et al.* 2005] HAN, K.; FENG, Y. ; OWEN, D.. **Sphere packing with a geometric based compression algorithm.** Powder Technology, 155(1):33–41, 2005.
- [Henkes *et al.* 2007] HENKES, S.; O'HERN, C. S. ; CHAKRABORTY, B.. **Entropy and temperature of a static granular assembly: An ab initio approach.** Physical review letters, 99(3):038002, 2007.
- [Jerier *et al.* 2009] JERIER, J.-F.; IMBAULT, D.; DONZE, F.-V. ; DOREMUS, P.. **A geometric algorithm based on tetrahedral meshes to generate a dense polydisperse sphere packing.** Granular Matter, 11(1):43–52, 2009.
- [Jerier *et al.* 2010] JERIER, J.-F.; RICHEFEU, V.; IMBAULT, D. ; DONZÉ, F.-V.. **Packing spherical discrete elements for large scale simulations.** Computer Methods in Applied Mechanics and Engineering, 199(25):1668–1676, 2010.
- [Jian *et al.* 2003] JIANG, M.; KONRAD, J. ; LEROUÉIL, S.. **An efficient technique for generating homogeneous specimens for dem studies.** Computers and geotechnics, 30(7):579–597, 2003.

- [Labra *et al.* 2009] LABRA, C.; ONATE, E.. **High-density sphere packing for discrete element method simulations.** Communications in Numerical Methods in Engineering, 25(7):837–849, 2009.
- [Lagarias *et al.* 2002] LAGARIAS, J. C.; MALLOWS, C. L. ; WILKS, A. R.. **Beyond the descartes circle theorem.** The American mathematical monthly, 109(4):338–361, 2002.
- [Li *et al.* 2018] LI, Y.; JI, S.. **A geometric algorithm based on the advancing front approach for sequential sphere packing.** Granular Matter, 20(4):59, 2018.
- [Liu 2008] LIU, J.; LI, S. ; CHEN, Y.. **A fast and practical method to pack spheres for mesh generation.** Acta Mechanica Sinica, 24(4):439–447, 2008.
- [Liu *et al.* 2012] LIU, J.; YUN, B. ; ZHAO, C.. **An improved specimen generation method for dem based on local delaunay tessellation and distance function.** International Journal for Numerical and Analytical Methods in Geomechanics, 36(5):653–674, 2012.
- [Liu *et al.* 2018] LIU, L.; JI, S.. **Ice load on floating structure simulated with dilated polyhedral discrete element method in broken ice field.** Applied Ocean Research, 75:53–65, 2018.
- [Lo 1985] LO, S.. **A new mesh generation scheme for arbitrary planar domains.** International journal for numerical methods in engineering, 21(8):1403–1426, 1985.
- [Lobello *et al.* 2014] LOBELLO, R. U.; DUPONT, F. ; DENIS, F.. **Out-of-core adaptive iso-surface extraction from binary volume data.** Graphical models, 76(6):593–608, 2014.
- [Lopes *et al.* 2020] LOPES, L. G.; CINTRA, D. T. ; LIRA, W. W.. **A geometric separation method for non-uniform disk packing with prescribed filling ratio and size distribution.** Computational Particle Mechanics, p. 1–14, 2020.
- [Lozano *et al.* 2016] LOZANO, E.; ROEHL, D.; CELES, W. ; GATTASS, M.. **An efficient algorithm to generate random sphere packs in arbitrary domains.** Computers & Mathematics with Applications, 71(8):1586–1601, 2016.

- [Lubachevsky *et al.* 1990] LUBACHEVSKY, B. D.; STILLINGER, F. H.. **Geometric properties of random disk packings.** Journal of statistical Physics, 60(5-6):561–583, 1990.
- [Marchi *et al.* 2019] MARCHI, B. C.; KETEN, S.. **Microstructure and size effects on the mechanics of two dimensional, high aspect ratio nanoparticle assemblies.** Frontiers in Materials, 6:174, 2019.
- [Meidani *et al.* 2018] MEIDANI, M.; MEGUID, M. A. ; CHOUINARD, L. E.. **A finite-discrete element approach for modelling polyethylene pipes subjected to axial ground movement.** International Journal of Geotechnical Engineering, p. 1–13, 2018.
- [Meyer *et al.* 2010] MEYER, S.; SONG, C.; JIN, Y.; WANG, K. ; MAKSE, H. A.. **Jamming in two-dimensional packings.** Physica A: Statistical Mechanics and its Applications, 389(22):5137–5144, 2010.
- [Miao *et al.* 2014] MIAO, Q.; HUANG, M.; XUE, J. ; BEN, Y.. **Spatial hashing based contact detection for numerical manifold method.** Geomechanics and Geoengineering, 9(2):153–159, 2014.
- [Morfa *et al.* 2018] MORFA, C. A. R.; MORALES, I. P. P.; DE FARIAS, M. M.; DE NAVARRA, E. O. I.; VALERA, R. R. ; CASAÑAS, H. D.-G.. **General advancing front packing algorithm for the discrete element method.** Computational Particle Mechanics, 5(1):13–33, 2018.
- [Munjiza *et al.* 1998] MUNJIZA, A.; ANDREWS, K.. **Nbs contact detection algorithm for bodies of similar size.** International Journal for Numerical Methods in Engineering, 43(1):131–149, 1998.
- [Nguyen *et al.* 2019] NGUYEN, T.-T.; ANDRÉ, D. ; HUGER, M.. **Analytic laws for direct calibration of discrete element modeling of brittle elastic media using cohesive beam model.** Computational Particle Mechanics, 6(3):393–409, 2019.
- [Nitka *et al.* 2018] NITKA, M.; TEJCHMAN, J.. **A three-dimensional meso-scale approach to concrete fracture based on combined dem with x-ray  $\mu$ ct images.** Cement and Concrete Research, 107:11–29, 2018.
- [Nitka *et al.* 2020] NITKA, M.; TEJCHMAN, J.. **Meso-mechanical modelling of damage in concrete using discrete element method with porous itzs of defined width around aggregates.** Engineering Fracture Mechanics, p. 107029, 2020.

- [O'Hern *et al.* 2002] O'HERN, C. S.; LANGER, S. A.; LIU, A. J. ; NAGEL, S. R.. **Random packings of frictionless particles.** Physical Review Letters, 88(7):075507, 2002.
- [Osa *et al.* 2018] OSA, J. L.; ORTEGA, N.; VIDAL, G.; FERNANDEZ-GAUNA, B.; CARBALLO, A. ; TOLOSA, I.. **Future of the discrete element method in the modelling of grinding wheels.** Engineering Computations, 2018.
- [Panigraphy *et al.* 2008] PANIGRAHY, R.; TALWAR, K. ; WIEDER, U.. **A geometric approach to lower bounds for approximate near-neighbor search and partial match.** In: 2008 49TH ANNUAL IEEE SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, p. 414–423. IEEE, 2008.
- [Papadopoulos *et al.* 2018] PAPADOPOULOS, L.; PORTER, M. A.; DANIELS, K. E. ; BASSETT, D. S.. **Network analysis of particles and grains.** Journal of Complex Networks, 6(4):485–565, 2018.
- [Scrimieri *et al.* 2014] SCRIMIERY, D.; AFAZOV, S. M.; BECKER, A. A. ; RATCHEV, S. M.. **Fast mapping of finite element field variables between meshes with different densities and element types.** Advances in Engineering Software, 67:90–98, 2014.
- [Skarżyński *et al.* 2015] SKARŻYŃSKI, Ł.; NITKA, M. ; TEJCHMAN, J.. **Modelling of concrete fracture at aggregate level using fem and dem based on x-ray  $\mu$ ct images of internal structure.** Engineering Fracture Mechanics, 147:13 – 35, 2015.
- [Specht 2015] SPECHT, E.. **A precise algorithm to detect voids in poly-disperse circle packings.** Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471(2182):20150421, 2015.
- [Stoyan98] STOYAN, D.. **Random sets: Models and statistics.** International Statistical Review, 66(1):1–27, 1998.
- [Taniyama 2017] TANIYAMA, H.. **Distinct element analysis of overburden subjected to reverse oblique-slip fault.** Journal of Structural Geology, 96:90–101, 2017.
- [Teuber *et al.* 2013] TEUBER, J.; WELLER, R.; ZACHMANN, G. ; GUTHE, S.. **Fast sphere packings with adaptive grids on the gpu.** GI AR/VRWorkshop (Würzburg, Germany, vol. 4, 2013.

- [Torquato *et al.* 2000] TORQUATO, S.; TRUSKETT, T. M. ; DEBENEDETTI, P. G.. Is random close packing of spheres well defined? Physical review letters, 84(10):2064, 2000.
- [Tulluri 2003] TULLURI, S. S.. Analysis of random packing of uniform spheres using the Monte Carlo simulation method. PhD thesis, New Jersey Institute of Technology, Newark, New Jersey, 2003.
- [Vallejos *et al.* 2016] VALLEJOS, J. A.; SUZUKI, K.; BRZOVIC, A. ; IVARS, D. M.. Application of synthetic rock mass modeling to veined core-size samples. International Journal of Rock Mechanics and Mining Sciences, 81:47–61, 2016.
- [Van der Linden *et al.* 2016] VAN DER LINDEN, J. H.; NARSILIO, G. A. ; TORDESILLAS, A.. Machine learning framework for analysis of transport through complex networks in porous, granular media: a focus on permeability. Physical Review E, 94(2):022904, 2016.
- [Wang *et al.* 2007] WANG, W.; MING, C. ; LO, S.. Generation of triangular mesh with specified size by circle packing. Advances in Engineering Software, 38(2):133–142, 2007.
- [Weaire *et al.* 2008] WEAIRE, D.; ASTE, T.. The pursuit of perfect packing. CRC Press, 2008.
- [Weller *et al.* 2010] WELLER, R.; ZACHMANN, G.. Protosphere: A gpu-assisted prototype guided sphere packing algorithm for arbitrary objects. In: ACM SIGGRAPH ASIA 2010 SKETCHES, SA '10, New York, NY, USA, 2010. Association for Computing Machinery.
- [Yeom *et al.* 2019] YEOM, S. B.; HA, E.-S.; KIM, M.-S.; JEONG, S. H.; HWANG, S.-J. ; CHOI, D. H.. Application of the discrete element method for manufacturing process simulation in the pharmaceutical industry. Pharmaceutics, 11(8):414, 2019.
- [Zhang *et al.* 2020] ZHANG, K.; LIU, F.; ZHAO, G. ; XIA, K.. Fast and efficient particle packing algorithms based on triangular mesh. Powder Technology, 2020.
- [Zsaki 2009a] ZSAKI, A.. An efficient method for packing polygonal domains with disks for 2d discrete element simulation. Computers and Geotechnics, 36(4):568–576, 2009.

- [Zsaki 2009b] ZSAKI, A.. Parallel generation of initial element assemblies for two-dimensional discrete element simulations. International journal for numerical and analytical methods in geomechanics, 33(3):377–389, 2009.



## A

### Mesh operations

To perform the necessary procedures on the mesh, we have implemented the following basic operations:

- **ps=PolygonStar(v)**: Given a vertex  $v$ , obtains the list of polygons  $ps$  containing  $v$ .
- **vs=VertexStar(v)**: Given a vertex  $v$ , obtains the list of vertices sharing an edge.
- **IsBoundary(e)**: Determines if an edge  $e$  is boundary.
- **e=NextBoundary()**: Loop over the boundary edges in the “outer loop”.
- **AddVertex(v)**: Adds a vertex  $v$  to the mesh PM.
- **AddPolygon(po)**: Adds a polygon  $po$  to the mesh PM.
- **IsBoundary(po)**: Determines if the polygon  $po$  is at the border of the mesh.
- **IsInside(pt, po)**: Determines if the point  $pt$  is inside the polygon  $po$ .
- **ps=SplitPolygon(v, po)**: Inserts a new vertex  $v$  inside the polygon  $po$ . The polygon  $po$  is removed from the mesh and new polygons  $po_i$  are added. To create new polygons, it is important to verify the connectivity of the particles in the polygon  $po$ .
- **RemovePolygon(po)** and **RemoveVertex(v)**: Removes polygons and vertices respectively. We must verify that  $po$  and  $v$  are boundary, except for the case of **SplitPolygon(v, po)**. The removal of vertex  $v$  must remove all the associated polygons as well. Removal operations must maintain the 2-connectivity condition of the “outer loop”. The graph also must have only one connected component. Sometimes additional vertices and polygons should be removed to meet these conditions. It is important to identify the vertices and polygons removed at the end of these operations because additional modifications on the mesh should be replicated on the grid index and pack assembly. And, in some cases, the operations must be reverted.

## B

### Distance field for simple geometries

For known geometries, such as circles and rectangles, it is straightforward to define a continuous signed distance field as described in Algorithm 8 and Algorithm 9. Note that we are just interested in distances inside the container; for those points beyond the frontiers of the region of interest, we just assign them invalid distances with the value of `-MAX_DISTANCE`.

---

**Algorithm 8:** Circle distance field

---

**Input** : Circle center and radius  $(p, r)$ . The query point  $x$ .

**Output:** The signed distance

```
1 sqDist = sqDist(p, x) ;           // square distance from  $p$  to  $x$ 
2 if ( $sqDist > r * r$ ) then
3   | return -MAX_DISTANCE;
4 else
5   | return r - sqrt(sqDist);
```

---

---

**Algorithm 9:** Rectangle distance field

---

**Input** : Rectangle's corners  $bmin$  and  $bmax$ . The query point  $x$ .

**Output:** The signed distance

```
1 distance = MAX_DISTANCE;
2 for  $i = 1$  to 2 do
3   |  $v = x[i]$ ;
4   | if ( $v > bmax[i]$ ) return -MAX_DISTANCE;
5   | if ( $v < bmin[i]$ ) return -MAX_DISTANCE;
6   | distance = min(distance,  $bmax[i] - v$ );
7   | distance = min(distance,  $v - bmin[i]$ );
8 return dist;
```

---

## C Flowcharts

Figure C.1 gives a broad view of the packing algorithm. The flowchart includes the use of the outer loop and internal strategies. Figure C.2 illustrates in detail the logic for the removal of the fronts and radius rejection.

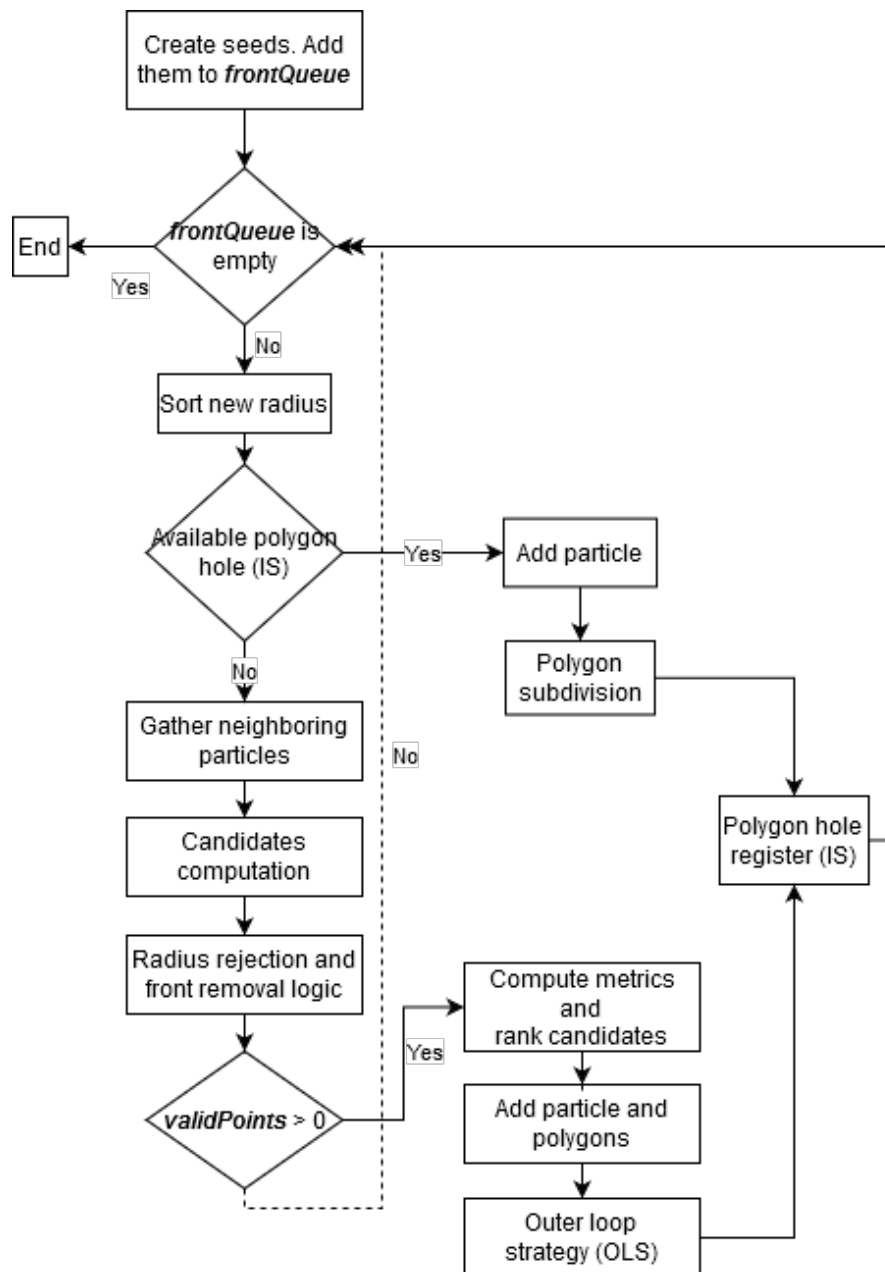


Figure C.1: Packing algorithm flowchart.

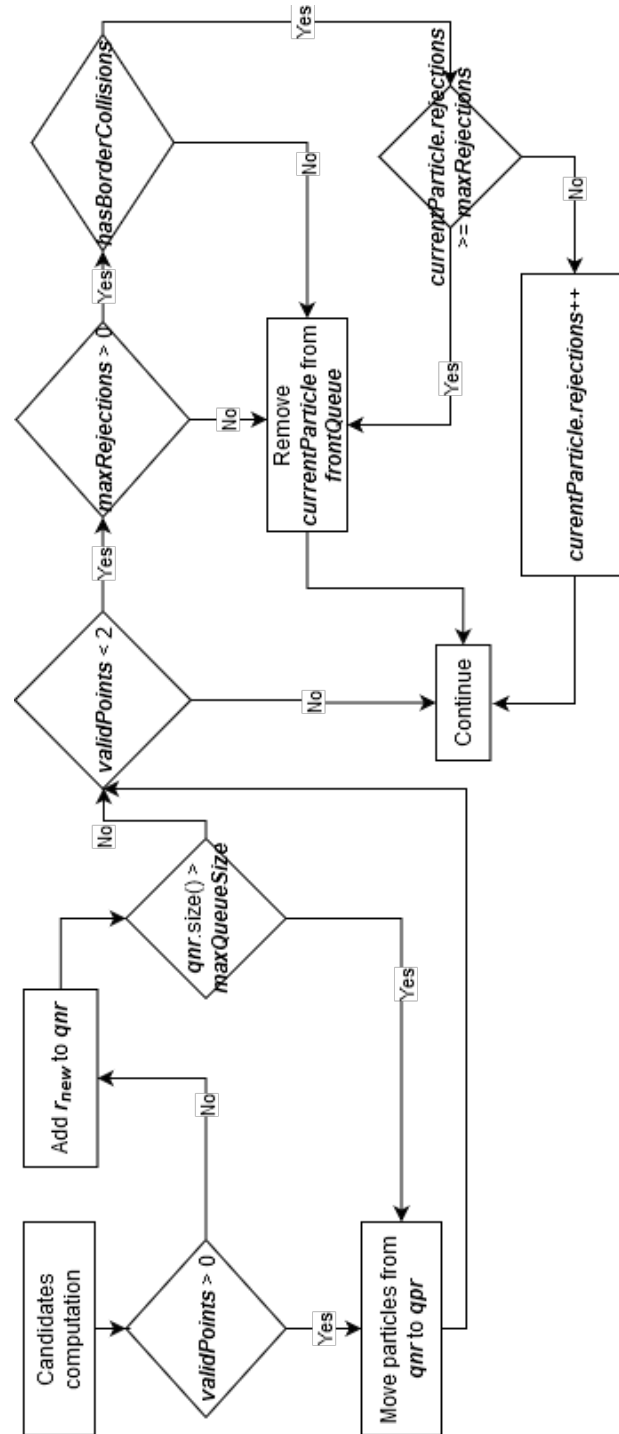


Figure C.2: Radius rejection and front removal logic.