Approximation Algorithms for Circle Packing

Flávio K. Miyazawa
UNICAMP

São Paulo School of Advanced Science on
Algorithms, Combinatorics and Optimization

São Paulo, SP

July, 2016

Contents

Colaborators and references

Main references

• F. K. Miyazawa, L. L. C. Pedrosa, R. C. S. Schouery, M. Sviridenko, Y. Wakabayashi. Polynomial-Time Approximation Schemes for Circle and Other Packing Problems. *Algorithmica*. To appear.

• P. H. Hokama, F. K. Miyazawa and R. C. S. Schouery. A bounded space algorithm for online circle packing. *Information Processing Letters*, 116, p. 337-342, 2016.

This work also obtained collaboration from Lehilton L. C. Pedrosa, Maxim Sviridenko, Rafael C. S. Schouery, Pedro H. Hokama and Yoshiko Wakabayashi.

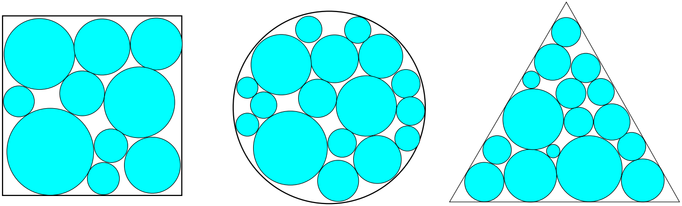
My thanks to Lehilton Pedrosa and Rafael Schouery that also contributed to part of these slides. This work obtained support from CNPq, FAPESP, UNICAMP, USP.

Circle Packing Problems

Packings

Given:

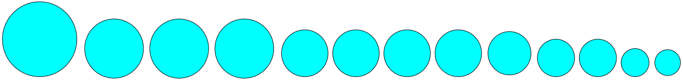
- A list of geometrical items L and bins $\mathcal{B}$
- Obtain a *good* packing of items in L into bin $B \in \mathcal{B}$
- The inner region of two packed items cannot overlap
- Each packed item must be totally contained in the bin



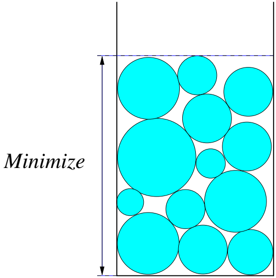
Problems

Circle Strip Packing

- Input: List of circles $L = (c_1, \ldots, c_n)$



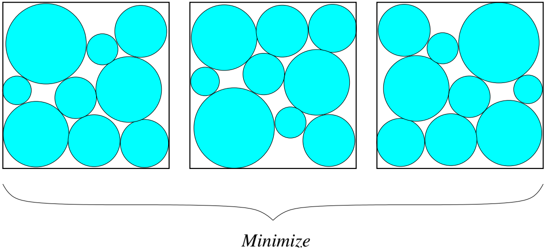
- Output: Packing of L into a rectangle of width 1 and minimum height.



Problems

Circle Bin Packing

- Input: List of circles $L = (c_1, \ldots, c_n)$
- Output: Packing of L into the minimum number of unit bins.



Some Applications

- Cutting and Packing of circular items
- Transportation of tubes, cilinders,...
- Cable assembly/allocations
- Tree plantation
- Origami design
- Marketing
- Cylinder pallet assembly
- ...

Marketing


&
Get Out There
DIVINE
DAIRY
snap
ANJOU
Boston and Albany


BURGER KING
NISSAN
LOTUS
STARBUCKS COFFEE
SAAB
FIAT
DELL


Web Images Maps News Shopping Gmail more ▾
Sign in
Google
california driving
Search
Advanced Search
Preferences
Web
Results 1 - 10 of about 1,680,000 for california driving. (0.06 seconds)
California Driving -- A Survivial Guide for Travellers and Tourists
An illustrated survival guide for foreign and domestic tourists and travellers, containing information, tips, warnings, etc, about driving in California.
www.caldrive.com/ - 5k - Cached - Similar pages
The California Driving Guide: Test Your California Driving IQ!
California Driving: Test Your California Driving IQ! So you think you're pretty au fait with California, its drivers, and its culture? ...
www.caldrive.com/iq.html - 24k - Cached - Similar pages
Samples of Driver Tests
... access are also encouraged to try our Driving Knowledge Tutorial. ... Review the California Driver Handbook. It contains a lot of information and it may ...
www.dmv.ca.gov/pubs/interactive/tdrive/exam.htm - 25k - Cached - Similar pages
California Department of Motor Vehicles
I'm New to California! Replace a Lost or Stolen Driver License? ... Request For Driving Record · Traffic Accident Report · Bill of Sale · More DMV Forms ...
www.dmv.ca.gov/ - 45k - Cached - Similar pages
Driving School Association of California
The Driving School Association of California website is dedicated to improving driver safety through educating and informing the public about it's related ...
www.dsac.com/ - 26k - Cached - Similar pages
Sponsored Links
California Road
Fast, easy to use maps & directions
Explore your area with Google Maps.
maps.google.com
DMV CA Online Guide
CA DMV Department of Motor Vehicles
Online Guide.
DMV.org/California
$24.95 Online Drivers ED
DMV Authorized Course & Certificate
Summer Special! Enroll now
www.DmvDriversEducation.org
California

Computational Complexity

Demaine, Fekete, Lang'10: To decide if a set of circles can be packed into a square is NP-hard.

Approximation Algorithms:

- Efficient Algorithms (polynomial time)
- Analysis: How far from the optimum solution value ?
- Compromise:

Computational Time × Solution Quality

Computational Complexity

Demaine, Fekete, Lang'10: To decide if a set of circles can be packed into a square is NP-hard.

Approximation Algorithms:

- Efficient Algorithms (polynomial time)
- Analysis: How far from the optimum solution value ?
- Compromise:

Computational Time $\times$ Solution Quality

Computational Complexity

Demaine, Fekete, Lang'10: To decide if a set of circles can be packed into a square is NP-hard.

Approximation Algorithms:

- Efficient Algorithms (polynomial time)
- Analysis: How far from the optimum solution value ?
- Compromise:

Computational Time × Solution Quality

Computational Complexity

Demaine, Fekete, Lang'10: To decide if a set of circles can be packed into a square is NP-hard.

Approximation Algorithms:

- Efficient Algorithms (polynomial time)
- Analysis: How far from the optimum solution value ?
- Compromise:

Computational Time $\times$ Solution Quality

Approximation Algorithms

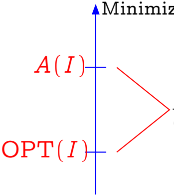
- $A(I)$ Value of the solution produced by A for instance I
- $\mathrm{OPT}(I)$ Value of an optimum solution of I

- A has approximation factor α if
- A has asymptotic approximation factor α if

for some constant β

Approximation Algorithms

- $A(I)$ Value of the solution produced by A for instance I
- $\mathrm{OPT}(I)$ Value of an optimum solution of I
- A has approximation factor α if



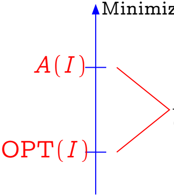
$$\frac{A(I)}{\mathrm{OPT}(I)} \leq \alpha \quad \text{for any instance } I$$

- A has asymptotic approximation factor α if

for some constant β

Approximation Algorithms

- $A(I)$ Value of the solution produced by A for instance I
- $\mathrm{OPT}(I)$ Value of an optimum solution of I
- A has approximation factor α if



$$\frac{A(I)}{\mathrm{OPT}(I)} \leq \alpha \quad \text{for any instance } I$$

- A has asymptotic approximation factor α if

$$A(I) \leq \alpha\,\mathrm{OPT}(I) + \beta \qquad \text{for any instance } I,$$

for some constant β

Approximation Algorithms

For Minimization Problems

PTAS: Polynomial Time Approximation Scheme

- A family of polynomial time algorithms A_ε, $\varepsilon > 0$, is a polynomial time approximation scheme if

$$A_\varepsilon(I) \leq (1+\varepsilon)\,\mathrm{OPT}(I), \qquad \text{for any instance } I$$

APTAS: Asymptotic Polynomial Time Approximation Scheme

- A family of algorithms A_ε, $\varepsilon > 0$ is an asymptotic polynomial time approximation scheme if

$$A_\varepsilon(I) \leq (1+\varepsilon)\,\mathrm{OPT}(I) + \beta_\varepsilon, \qquad \text{for any instance } I$$

where β_ε is a constant that depends only on ε

Approximation Algorithms

For Minimization Problems

PTAS: Polynomial Time Approximation Scheme

- A family of polynomial time algorithms A_ε, $\varepsilon > 0$, is a polynomial time approximation scheme if

$$A_\varepsilon(I) \leq (1+\varepsilon)\,\mathrm{OPT}(I), \quad \text{for any instance } I$$

APTAS: Asymptotic Polynomial Time Approximation Scheme

- A family of algorithms A_ε, $\varepsilon > 0$ is an asymptotic polynomial time approximation scheme if

$$A_\varepsilon(I) \leq (1+\varepsilon)\,\mathrm{OPT}(I) + \beta_\varepsilon, \quad \text{for any instance } I$$

where β_ε is a constant that depends only on ε

Online Packing Algorithms

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Online Packing Algorithms

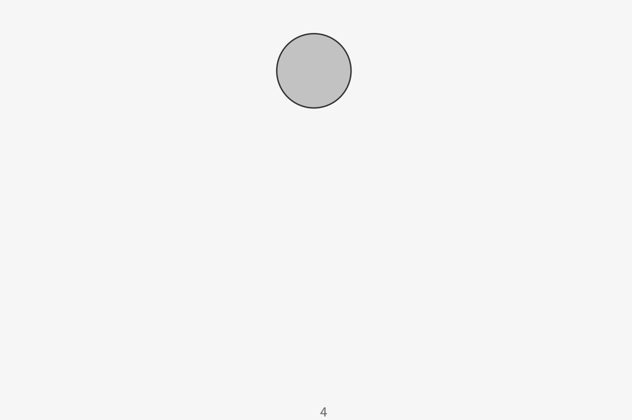
- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Online Packing Algorithms

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

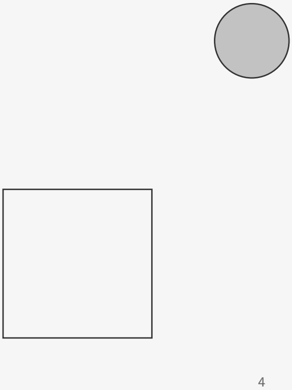
Online Circle Bin Packing

Example



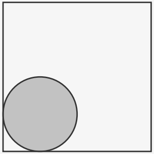
Online Circle Bin Packing

Example



Online Circle Bin Packing

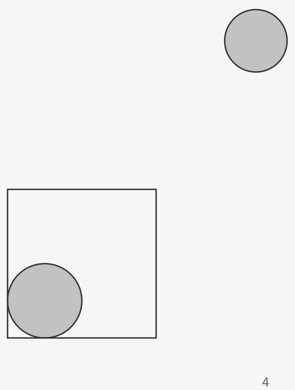
Example



4

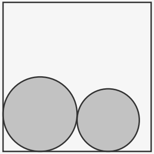
Online Circle Bin Packing

Example



Online Circle Bin Packing

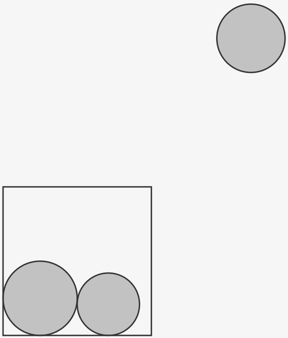
Example



4

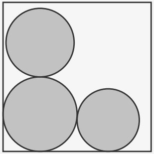
Online Circle Bin Packing

Example



Online Circle Bin Packing

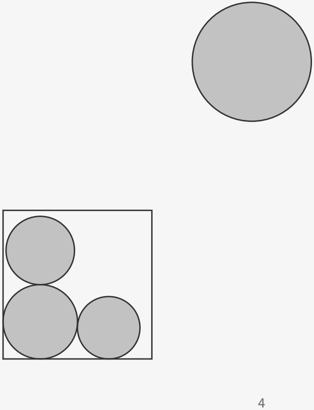
Example



4

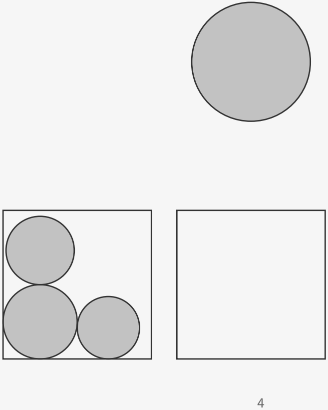
Online Circle Bin Packing

Example



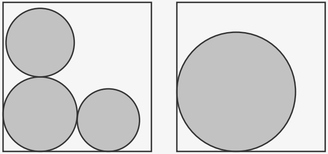
Online Circle Bin Packing

Example



Online Circle Bin Packing

Example



4

Preliminaries

Some Notation

- If $f : D \to \mathbb{R}$ is a numerical function, we may write
  - f_e and $f(e)$, indistictly
  - $f(S)$ as the value $\sum_{e \in D} f(e)$, there is no explicit definition
- If c is a circle and $L = (c_1, \ldots, c_n)$ a list of circles, then
  - c is also used to denote its radius
  - $\hat{c}$ is the square with side lengths $2c$
  - $\hat{L}$ is the list $\hat{L} = (\hat{c}_1, \ldots, \hat{c}_n)$
  - $\max(L)$ is the maximum radius of a circle in L
  - $\text{Area}(L)$ is the total area of the circles in L
  - $\overline{L}$ is the list with $|L|$ equal circles with radius $\max(L)$
  - $\mathcal{C}$ is the set containing all lists of circles for the input problem

Preliminaries

Some Notation

- If $f : D \to \mathbb{R}$ is a numerical function, we may write
  - f_e and $f(e)$, indistictly
  - $f(S)$ as the value $\sum_{e \in D} f(e)$, there is no explicit definition
- If c is a circle and $L = (c_1, \ldots, c_n)$ a list of circles, then
  - c is also used to denote its radius
  - $\hat{c}$ is the square with side lengths $2c$
  - $\hat{L}$ is the list $\hat{L} = (\hat{c}_1, \ldots, \hat{c}_n)$
  - $\max(L)$ is the maximum radius of a circle in L
  - $\mathrm{Area}(L)$ is the total area of the circles in L
  - $\overline{L}$ is the list with $|L|$ equal circles with radius $\max(L)$
  - $\mathcal{C}$ is the set containing all lists of circles for the input problem

Preliminaries

If $\mathcal{E}$ is a packing or other geometrical composition, $\mathcal{E}$ may be considered as the solid structure

- $\text{width}(\mathcal{E})$ is the width of $\mathcal{E}$
- $\text{height}(\mathcal{E})$ is the height of $\mathcal{E}$

First considerations

- We consider a more general computational model
- Possible to operate over polynomial solutions

Preliminaries

If $\mathcal{E}$ is a packing or other geometrical composition, $\mathcal{E}$ may be considered as the solid structure

- $\text{width}(\mathcal{E})$ is the width of $\mathcal{E}$
- $\text{height}(\mathcal{E})$ is the height of $\mathcal{E}$

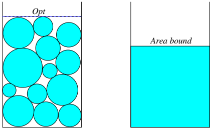
First considerations

- We consider a more general computational model
- Possible to operate over polynomial solutions

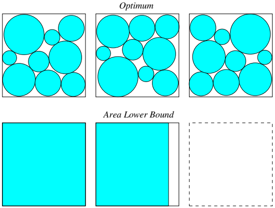
Basic Algorithms

Area Lower Bound

- Circle Strip Packing with bin width 1



- Circle Bin Packing with unit square bins



Area based algorithms

Let

$\mathcal{C}$ the set containing all lists of circles, and

$\mathcal{Q}$ the set containing all lists of squares

next algorithm is a circle version $\mathbb{C}\mathcal{A}$ from square packing algorithm $\mathcal{A}$

$\mathbb{C}\mathcal{A}(L)$

1. Let $\hat{\mathcal{P}} \leftarrow \mathcal{A}(\hat{L})$.
2. Let $\mathcal{P}$ the packing $\hat{\mathcal{P}}$ replacing $\hat{c}_i$ by c_i.
3. Return $\mathcal{P}$.

Lemma. If $\mathcal{A}$ is a square packing algorithm and α, β are constants, st. $\mathcal{A}(S) \leq \alpha \text{Area}(S) + \beta$, for any $S \in \mathcal{Q}$

$$\mathbb{C}\mathcal{A}(L) \leq \alpha \frac{4}{\pi} \text{Area}(L) + \beta, \text{ for any } L \in \mathcal{C}$$

Area based algorithms

Let

$\mathcal{C}$ the set containing all lists of circles, and

$\mathcal{Q}$ the set containing all lists of squares

next algorithm is a circle version $\mathbb{C}\mathcal{A}$ from square packing algorithm $\mathcal{A}$

$\mathbb{C}\mathcal{A}(L)$

1. Let $\widehat{\mathcal{P}} \leftarrow \mathcal{A}(\hat{L})$.
2. Let $\mathcal{P}$ the packing $\widehat{\mathcal{P}}$ replacing $\hat{c}_i$ by c_i.
3. Return $\mathcal{P}$.

Lemma. If $\mathcal{A}$ is a square packing algorithm and α, β are constants, st. $\mathcal{A}(S) \leq \alpha \text{Area}(S) + \beta$, for any $S \in \mathcal{Q}$

$$\mathbb{C}\mathcal{A}(L) \leq \alpha \frac{4}{\pi} \text{Area}(L) + \beta, \text{ for any } L \in \mathcal{C}$$

Area based algorithms

Let

$\mathcal{C}$ the set containing all lists of circles, and

$\mathcal{Q}$ the set containing all lists of squares

next algorithm is a circle version $\mathbb{C}\mathcal{A}$ from square packing algorithm $\mathcal{A}$

$\mathbb{C}\mathcal{A}(L)$

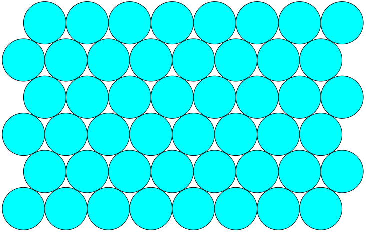
1. Let $\hat{\mathcal{P}} \leftarrow \mathcal{A}(\hat{L})$.
2. Let $\mathcal{P}$ the packing $\hat{\mathcal{P}}$ replacing $\hat{c}_i$ by c_i.
3. Return $\mathcal{P}$.

Lemma. If $\mathcal{A}$ is a square packing algorithm and α, β are constants, st. $\mathcal{A}(S) \leq \alpha \text{Area}(S) + \beta$, for any $S \in \mathcal{Q}$

$$\mathbb{C}\mathcal{A}(L) \leq \alpha \frac{4}{\pi} \text{Area}(L) + \beta, \text{ for any } L \in \mathcal{C}$$

Area based algorithms

Best possible density



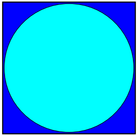
Hexagonal Packing

- Density $\frac{\pi}{\sqrt{12}} \approx 0.9069$

Lemma. There is no algorithm with approximation factor, based only on area arguments, better than $\frac{\sqrt{12}}{\pi} \approx 1.10266$

Using square packing algorithms

- Round each circle c_i to a square $\hat{c}_i$:

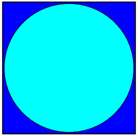


- Use square packing algorithms
- Area increasing: $\frac{Area(\hat{c}_i)}{Area(c_i)} = \frac{4}{\pi} \approx 1.27324$
- Let $\hat{L} = (\hat{c}_1, \ldots, \hat{c}_n)$ the list L rounding each circle to a square
- Bounding the optimum with the area:

$$\text{Area}(\hat{L}) = \frac{4}{\pi}\text{Area}(L) \leq \frac{4}{\pi}\text{OPT}(L)$$

Using square packing algorithms

- Round each circle c_i to a square $\widehat{c}_i$:

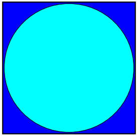


- Use square packing algorithms
- Area increasing: $\frac{Area(\widehat{c}_i)}{Area(c_i)} = \frac{4}{\pi} \approx 1.27324$
- Let $\widehat{L} = (\widehat{c}_1, \ldots, \widehat{c}_n)$ the list L rounding each circle to a square
- Bounding the optimum with the area:

$$\text{Area}(\widehat{L}) = \frac{4}{\pi}\text{Area}(L) \leq \frac{4}{\pi}\text{OPT}(L)$$

Using square packing algorithms

- Round each circle c_i to a square $\hat{c}_i$:

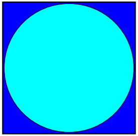


- Use square packing algorithms
- Area increasing: $\frac{Area(\hat{c}_i)}{Area(c_i)} = \frac{4}{\pi} \approx 1.27324$
- Let $\hat{L} = (\hat{c}_1, \ldots, \hat{c}_n)$ the list L rounding each circle to a square
- Bounding the optimum with the area:

$$\mathrm{Area}(\hat{L}) = \frac{4}{\pi}\mathrm{Area}(L) \leq \frac{4}{\pi}\mathrm{OPT}(L)$$

Using square packing algorithms

- Round each circle c_i to a square $\widehat{c}_i$:

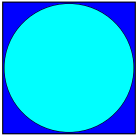


- Use square packing algorithms
- Area increasing: $\frac{Area(\widehat{c}_i)}{Area(c_i)} = \frac{4}{\pi} \approx 1.27324$
- Let $\widehat{L} = (\widehat{c}_1, \ldots, \widehat{c}_n)$ the list L rounding each circle to a square
- Bounding the optimum with the area:

$$\mathrm{Area}(\widehat{L}) = \frac{4}{\pi}\mathrm{Area}(L) \leq \frac{4}{\pi}\mathrm{OPT}(L)$$

Using square packing algorithms

- Round each circle c_i to a square $\hat{c}_i$:

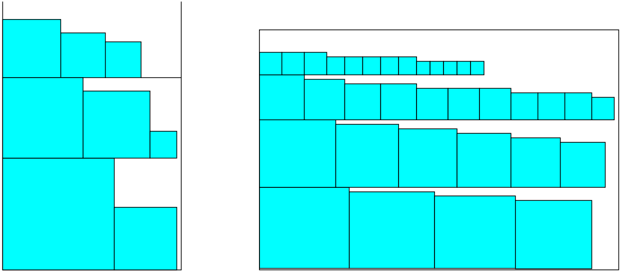


- Use square packing algorithms
- Area increasing: $\frac{Area(\hat{c}_i)}{Area(c_i)} = \frac{4}{\pi} \approx 1.27324$
- Let $\hat{L} = (\hat{c}_1, \ldots, \hat{c}_n)$ the list L rounding each circle to a square
- Bounding the optimum with the area:

$$\text{Area}(\hat{L}) = \frac{4}{\pi}\text{Area}(L) \leq \frac{4}{\pi}\text{OPT}(L)$$

Using square packing algorithms

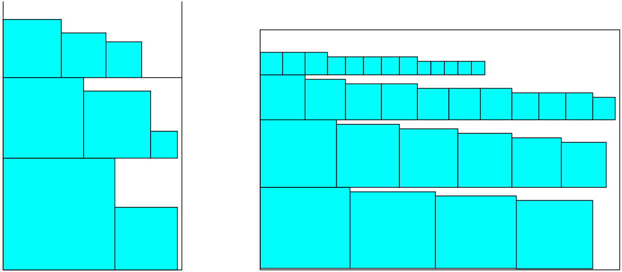
Shelf Packing:



- Items are packed over shelves (of zero thickness)
- side by side in a leftmost way
- Items in a same shelf are packed at the same height.
- Item s can be packed in a shelf S if $\text{width}(s) + \text{width}(S) \leq 1$

Using square packing algorithms

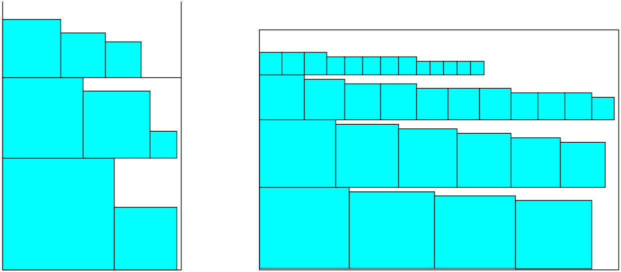
Shelf Packing:



- Items are packed over shelves (of zero thickness)
- side by side in a leftmost way
- Items in a same shelf are packed at the same height.
- Item s can be packed in a shelf S if $\text{width}(s) + \text{width}(S) \leq 1$

Using square packing algorithms

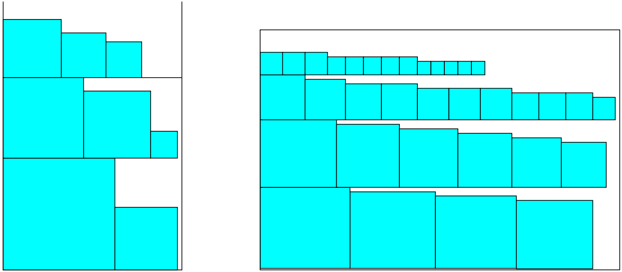
Shelf Packing:



- Items are packed over shelves (of zero thickness)
- side by side in a leftmost way
- Items in a same shelf are packed at the same height.
- Item s can be packed in a shelf S if $\text{width}(s) + \text{width}(S) \leq 1$

Using square packing algorithms

Shelf Packing:

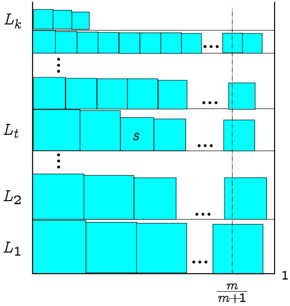


- Items are packed over shelves (of zero thickness)
- side by side in a leftmost way
- Items in a same shelf are packed at the same height.
- Item s can be packed in a shelf S if $\text{width}(s) + \text{width}(S) \leq 1$

Using square packing algorithms

NFDH^s(L) # for Strip Packing

1. Sort $L = (s_1, \ldots, s_n)$ st. $s_1 \geq \cdots \geq s_n$
2. For $i \leftarrow 1$ to n:
3. Pack s_i into the last shelf, if possible
4. otherwise, pack s_i in a new shelf on top of the previous shelf or bin's bottom (in case there is no previous shelf)



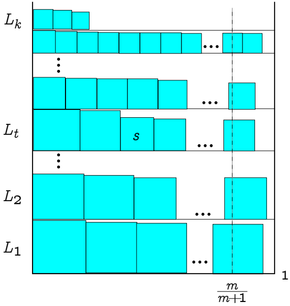
Lemma. If L has only squares with side lengths at most $1/m$

$$\text{NFDH}^s(L) \leq \frac{m+1}{m}\text{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

NFDH^s(L) # for Strip Packing

1. Sort $L = (s_1, \ldots, s_n)$ st. $s_1 \geq \cdots \geq s_n$
2. For $i \leftarrow 1$ to n:
3. Pack s_i into the last shelf, if possible
4. otherwise, pack s_i in a new shelf on top of the previous shelf or bin's bottom (in case there is no previous shelf)



Lemma. If L has only squares with side lengths at most $1/m$

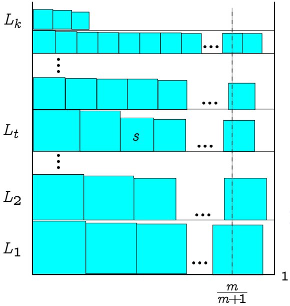
$$\text{NFDH}^s(L) \leq \frac{m+1}{m}\text{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Sketch.

Each of the first $k-1$ shelves have width filled by at least $\frac{m}{m+1}$

Let L_t the first shelf having square with side $\leq 1/(m+1)$



- $L_1, \ldots, L_{t-1}$: m squares of side $> \frac{1}{m+1}$, each.
- $L_t, \ldots, L_k$: width filled $> 1 - \frac{1}{m+1} = \frac{m}{m+1}$, otherwise receive another item.

Sliding up squares in L_i can cover all rectangular region of shelf L_{i+1}, up to width $\frac{m}{m+1}$.

$$(\text{NFDH}^s(L) - \mathit{height}(L_1))\frac{m}{m+1} \leq \text{Area}(L)$$

So,

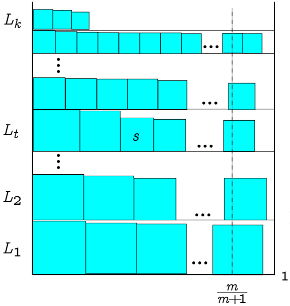
$$\text{NFDH}^s(L) \leq \frac{m+1}{m}\text{Area}(L) + \mathit{height}(L_1) \leq \frac{m+1}{m}\text{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Sketch.

Each of the first $k-1$ shelves have width filled by at least $\frac{m}{m+1}$

Let L_t the first shelf having square with side $\leq 1/(m+1)$



- $L_1, \ldots, L_{t-1}$: m squares of side $> \frac{1}{m+1}$, each.
- $L_t, \ldots, L_k$: width filled $> 1 - \frac{1}{m+1} = \frac{m}{m+1}$, otherwise receive another item.

Sliding up squares in L_i can cover all rectangular region of shelf L_{i+1}, up to width $\frac{m}{m+1}$.

$$(\text{NFDH}^s(L) - \mathit{height}(L_1))\frac{m}{m+1} \leq \text{Area}(L)$$

So,

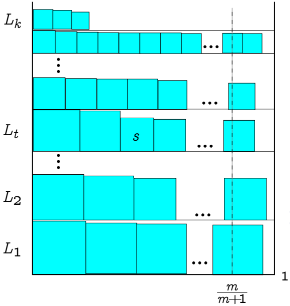
$$\text{NFDH}^s(L) \leq \frac{m+1}{m}\text{Area}(L) + \mathit{height}(L_1) \leq \frac{m+1}{m}\text{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Sketch.

Each of the first $k-1$ shelves have width filled by at least $\frac{m}{m+1}$

Let L_t the first shelf having square with side $\leq 1/(m+1)$



- $L_1,\ldots,L_{t-1}$: m squares of side $> \frac{1}{m+1}$, each.
- $L_t,\ldots,L_k$: width filled $> 1 - \frac{1}{m+1} = \frac{m}{m+1}$, otherwise receive another item.

Sliding up squares in L_i can cover all rectangular region of shelf L_{i+1}, up to width $\frac{m}{m+1}$.

$$(\mathrm{NFDH}^s(L) - \mathit{height}(L_1))\frac{m}{m+1} \leq \mathrm{Area}(L)$$

So,

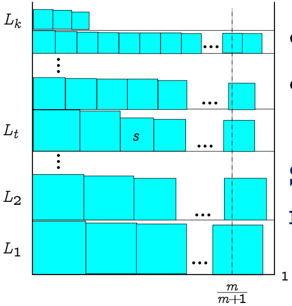
$$\mathrm{NFDH}^s(L) \leq \frac{m+1}{m}\mathrm{Area}(L) + \mathit{height}(L_1) \leq \frac{m+1}{m}\mathrm{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Sketch.

Each of the first $k-1$ shelves have width filled by at least $\frac{m}{m+1}$

Let L_t the first shelf having square with side $\leq 1/(m+1)$



- $L_1, \ldots, L_{t-1}$: m squares of side $> \frac{1}{m+1}$, each.
- $L_t, \ldots, L_k$: width filled $> 1 - \frac{1}{m+1} = \frac{m}{m+1}$, otherwise receive another item.

Sliding up squares in L_i can cover all rectangular region of shelf L_{i+1}, up to width $\frac{m}{m+1}$.

$$(\mathrm{NFDH}^{\mathrm{S}}(L) - height(L_1))\frac{m}{m+1} \leq \mathrm{Area}(L)$$

So,

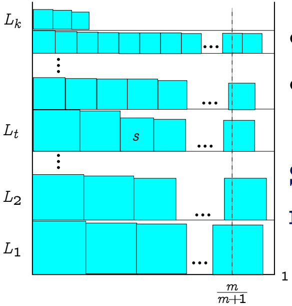
$$\mathrm{NFDH}^{\mathrm{S}}(L) \leq \frac{m+1}{m}\mathrm{Area}(L) + height(L_1) \leq \frac{m+1}{m}\mathrm{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Sketch.

Each of the first $k-1$ shelves have width filled by at least $\frac{m}{m+1}$

Let L_t the first shelf having square with side $\leq 1/(m+1)$



- $L_1, \ldots, L_{t-1}$: m squares of side $> \frac{1}{m+1}$, each.
- $L_t, \ldots, L_k$: width filled $> 1 - \frac{1}{m+1} = \frac{m}{m+1}$, otherwise receive another item.

Sliding up squares in L_i can cover all rectangular region of shelf L_{i+1}, up to width $\frac{m}{m+1}$.

$$(\mathrm{NFDH}^{\mathrm{S}}(L) - \mathit{height}(L_1))\frac{m}{m+1} \leq \mathrm{Area}(L)$$

So,

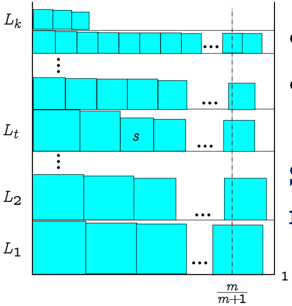
$$\mathrm{NFDH}^{\mathrm{S}}(L) \leq \frac{m+1}{m}\mathrm{Area}(L) + \mathit{height}(L_1) \leq \frac{m+1}{m}\mathrm{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Sketch.

Each of the first $k-1$ shelves have width filled by at least $\frac{m}{m+1}$

Let L_t the first shelf having square with side $\leq 1/(m+1)$



- $L_1,\ldots,L_{t-1}$: m squares of side $> \frac{1}{m+1}$, each.
- $L_t,\ldots,L_k$: width filled $> 1 - \frac{1}{m+1} = \frac{m}{m+1}$, otherwise receive another item.

Sliding up squares in L_i can cover all rectangular region of shelf L_{i+1}, up to width $\frac{m}{m+1}$.

$$(\mathrm{NFDH}^{\mathrm{S}}(L) - \mathit{height}(L_1))\frac{m}{m+1} \leq \mathrm{Area}(L)$$

So,

$$\mathrm{NFDH}^{\mathrm{S}}(L) \leq \frac{m+1}{m}\mathrm{Area}(L) + \mathit{height}(L_1) \leq \frac{m+1}{m}\mathrm{Area}(L) + \frac{1}{m}$$

Using square packing algorithms

Let

$\mathcal{C}_m$ the set of lists with small circles (diam. $\leq 1/m$, m integer)

Corollary. If $L \in \mathcal{C}_m$, then

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{s}}(L) \leq \frac{m+1}{m}\frac{4}{\pi}\mathrm{OPT}(L) + \frac{1}{m} \quad \forall L$$

Corollary. If L is a list of circles then

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{s}}(L) \leq 2.548\,\mathrm{OPT}(L) + 1$$

Using square packing algorithms

Let

$\mathcal{C}_m$ the set of lists with small circles (diam. $\leq 1/m$, m integer)

Corollary. If $L \in \mathcal{C}_m$, then

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{s}}(L) \leq \frac{m+1}{m}\frac{4}{\pi}\mathrm{OPT}(L) + \frac{1}{m} \quad \forall L$$

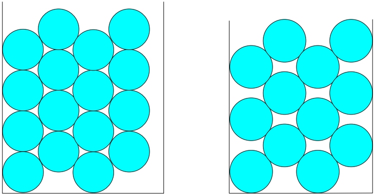
Corollary. If L is a list of circles then

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{s}}(L) \leq 2.548\,\mathrm{OPT}(L) + 1$$

Rounding circles to circles

Algorithm EqualCircles(L)   # all circles in L have a same size

1. Let $\mathcal{P}'$ and $\mathcal{P}''$ packings of L as below



2. Return packing $\mathcal{P} \in \{\mathcal{P}', \mathcal{P}''\}$ with minimum height.

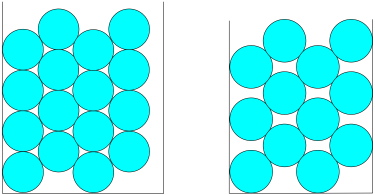
Lemma. If all circles of L have radius r, then

$$\text{EqualCircles}(L) \leq 1.654\text{Area}(L) + 2r.$$

Rounding circles to circles

Algorithm EqualCircles(L) # all circles in L have a same size

1. Let $\mathcal{P}'$ and $\mathcal{P}''$ packings of L as below



2. Return packing $\mathcal{P} \in \{\mathcal{P}', \mathcal{P}''\}$ with minimum height.

Lemma. If all circles of L have radius r, then

$$\text{EqualCircles}(L) \leq 1.654\text{Area}(L) + 2r.$$

Rounding circles to circles

Idea: Circles with close radius are rounded up to the same radius

Given $L \in \mathcal{C}$, $\overline{L}$ is the list with $|L|$ circles with radius $\max(L)$

Algorithm $\mathcal{A}_\varepsilon(L)$

1. $\delta \leftarrow \frac{\varepsilon}{6}$.
2. For $i \geq 0$ do
3. $\quad L_i \leftarrow \{r \in L : \frac{1/2}{(1+\delta)^{i+1}} < r \leq \frac{1/2}{(1+\delta)^{i}}\}$.
4. $\quad \mathcal{P}_i \leftarrow \text{EqualCircles}(\overline{L}_i)$.
5. $\mathcal{P} \leftarrow \mathcal{P}_0 \| \mathcal{P}_1 \| \mathcal{P}_2 \| \ldots$ # concatenation of packings
6. Return $\mathcal{P}$.

Theorem. Given $\varepsilon > 0$, we have

$$\mathcal{A}_\varepsilon(L) \leq (1.654 + \varepsilon)\,\text{Area}(L) + C_\varepsilon, \text{ for any } L \in \mathcal{C}$$

Rounding circles to circles

Idea: Circles with close radius are rounded up to the same radius

Given $L \in \mathcal{C}$, $\overline{L}$ is the list with $|L|$ circles with radius $\max(L)$

Algorithm $\mathcal{A}_\varepsilon(L)$

1. $\delta \leftarrow \frac{\varepsilon}{6}$.
2. For $i \geq 0$ do
3. $\quad L_i \leftarrow \{r \in L : \frac{1/2}{(1+\delta)^{i+1}} < r \leq \frac{1/2}{(1+\delta)^i}\}$.
4. $\quad \mathcal{P}_i \leftarrow \text{EqualCircles}(\overline{L_i})$.
5. $\mathcal{P} \leftarrow \mathcal{P}_0 \| \mathcal{P}_1 \| \mathcal{P}_2 \| \ldots$ # concatenation of packings
6. Return $\mathcal{P}$.

Theorem. Given $\varepsilon > 0$, we have

$$\mathcal{A}_\varepsilon(L) \leq (1.654 + \varepsilon)\,\text{Area}(L) + C_\varepsilon, \text{ for any } L \in \mathcal{C}$$

Online Circle Strip Packing

Online Packing

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Baker, Schwarz'83: For $0 < p < 1$, there exists online algorithm $\mathbb{C}\mathrm{NFS}_p$ s.t.,

$$\mathbb{C}\mathrm{NFS}_p(L) \leq \frac{2.548}{p}\mathrm{OPT}(L) + \frac{1}{p(1-p)}, \text{ for any } L \in C$$

Online Circle Strip Packing

Online Packing

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Baker, Schwarz'83: For $0 < p < 1$, there exists online algorithm $\mathbb{C}\mathrm{NFS}_p$ s.t.,

$$\mathbb{C}\mathrm{NFS}_p(L) \leq \frac{2.548}{p}\mathrm{OPT}(L) + \frac{1}{p(1-p)}, \text{ for any } L \in C$$

Online Circle Strip Packing

Online Packing

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Baker, Schwarz'83: For $0 < p < 1$, there exists online algorithm $\mathbb{C}\mathrm{NFS}_p$ s.t.,

$$\mathbb{C}\mathrm{NFS}_p(L) \leq \frac{2.548}{p}\mathrm{OPT}(L) + \frac{1}{p(1-p)}, \text{ for any } L \in C$$

Online Circle Strip Packing

Online Packing

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Baker, Schwarz'83: For $0 < p < 1$, there exists online algorithm $\mathbb{C}\text{NFS}_p$ s.t.,

$$\mathbb{C}\text{NFS}_p(L) \leq \frac{2.548}{p}\,\text{OPT}(L) + \frac{1}{p(1-p)}, \text{ for any } L \in C$$

Online Circle Strip Packing

Online Packing

- Incoming items appears one after the other, sequentially
- An incoming item must be packed when it arrives, without the knowledge of further items
- Once an item is packed, it cannot be repacked again.

Baker, Schwarz'83: For $0 < p < 1$, there exists online algorithm $\mathbb{C}\mathrm{NFS}_p$ s.t.,

$$\mathbb{C}\mathrm{NFS}_p(L) \leq \frac{2.548}{p}\,\mathrm{OPT}(L) + \frac{1}{p(1-p)}, \text{ for any } L \in \mathcal{C}$$

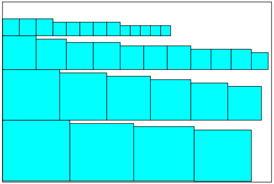
Circle Bin Packing

Rounding to squares

Adaptation of the strip packing version NFDH^s.

NFDH^b(L) # For the bin packing version

1. Sort $L = (s_1, \ldots, s_n)$ st. $s_1 \geq \cdots \geq s_n$
2. For $i \leftarrow 1$ to n:
3. Pack s_i in the last shelf (of the last bin), if possible
4. otherwise, pack s_i in a new shelf at the top of the previous shelf, if possible
5. otherwise, pack s_i in a new shelf of a new bin.



Rounding to squares

Meir, Moser'68. If all squares of L have side lengths at most $\frac{1}{m}$

$$\mathrm{NFDH}^{\mathrm{b}}(L) \leq \left(\frac{m+1}{m}\right)^2 \mathrm{Area}(L) + \frac{m+2}{m}$$

Proof: Exercise (analogous to the proof of $\mathrm{NFDH}^{\mathrm{s}}$)

Corollary. For any list $L \in C$ with diameters at most $\frac{1}{m}$

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}(L) \leq \frac{4}{\pi}\left(\frac{m+1}{m}\right)^2 \mathrm{Area}(L) + \frac{m+2}{m}$$

Corollary. For any list $L \in C$

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}(L) \leq 5.1\mathrm{OPT}(L) + 3$$

Corollary. As radius of circles decrease, the density of the packing is improved and $\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}$ goes to $4/\pi \approx 1.27324$.

Rounding to squares

Meir, Moser'68. If all squares of L have side lengths at most $\frac{1}{m}$

$$\mathrm{NFDH}^{\mathrm{b}}(L) \leq \left(\frac{m+1}{m}\right)^2 \mathrm{Area}(L) + \frac{m+2}{m}$$

Proof: Exercise (analogous to the proof of $\mathrm{NFDH}^{\mathrm{s}}$)

Corollary. For any list $L \in \mathcal{C}$ with diameters at most $\frac{1}{m}$

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}(L) \leq \frac{4}{\pi}\left(\frac{m+1}{m}\right)^2 \mathrm{Area}(L) + \frac{m+2}{m}$$

Corollary. For any list $L \in \mathcal{C}$

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}(L) \leq 5.1\mathrm{OPT}(L) + 3$$

Corollary. As radius of circles decrease, the density of the packing is improved and $\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}$ goes to $4/\pi \approx 1.27324$.

Rounding to squares

Meir, Moser'68. If all squares of L have side lengths at most $\frac{1}{m}$

$$\mathrm{NFDH}^{\mathrm{b}}(L) \leq \left(\frac{m+1}{m}\right)^2 \mathrm{Area}(L) + \frac{m+2}{m}$$

Proof: Exercise (analogous to the proof of $\mathrm{NFDH}^{\mathrm{s}}$)

Corollary. For any list $L \in \mathcal{C}$ with diameters at most $\frac{1}{m}$

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}(L) \leq \frac{4}{\pi}\left(\frac{m+1}{m}\right)^2 \mathrm{Area}(L) + \frac{m+2}{m}$$

Corollary. For any list $L \in \mathcal{C}$

$$\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}(L) \leq 5.1\mathrm{OPT}(L) + 3$$

Corollary. As radius of circles decrease, the density of the packing is improved and $\mathbb{C}\mathrm{NFDH}^{\mathrm{b}}$ goes to $4/\pi \approx 1.27324$.

Rounding to squares

Meir, Moser'68. If all squares of L have side lengths at most $\frac{1}{m}$

$$\text{NFDH}^{\text{b}}(L) \leq \left(\tfrac{m+1}{m}\right)^2 \text{Area}(L) + \tfrac{m+2}{m}$$

Proof: Exercise (analogous to the proof of NFDH^{s})

Corollary. For any list $L \in \mathcal{C}$ with diameters at most $\frac{1}{m}$

$$\mathbb{C}\text{NFDH}^{\text{b}}(L) \leq \frac{4}{\pi}\left(\frac{m+1}{m}\right)^2 \text{Area}(L) + \frac{m+2}{m}$$

Corollary. For any list $L \in \mathcal{C}$

$$\mathbb{C}\text{NFDH}^{\text{b}}(L) \leq 5.1\text{OPT}(L) + 3$$

Corollary. As radius of circles decrease, the density of the packing is improved and $\mathbb{C}\text{NFDH}^{\text{b}}$ goes to $4/\pi \approx 1.27324$.

Bounded Space Online Bin Packing

Bounded Space Online Bin Packing

- Algorithms must be online
- At any moment, bins are classified as *open* or *closed*
- Only open bins can receive new items
- A bin starts open and once it became closed, it cannot be open again.
- The number of open bins is bounded by a constant

Bounded Space Online Bin Packing

Related results with asymptotic approximation:

- Lee and Lee: Algorithm with factor
  1.69103 for 1-dimensional items and
  showed that no algorithm can have better performance
- Epstein, van Stee'07: Algorithms with factors
  2.3722 for packing squares and
  3.0672 for packing cubes.

Bounded Space Online Bin Packing

Related results with asymptotic approximation:

- Lee and Lee: Algorithm with factor
  1.69103 for 1-dimensional items and
  showed that no algorithm can have better performance

- Epstein, van Stee'07: Algorithms with factors
  2.3722 for packing squares and
  3.0672 for packing cubes.

Bounded Space Online Bin Packing

We will see

- Algorithm with asymptotic approximation factor 2.44
- Lower bound of 2.29

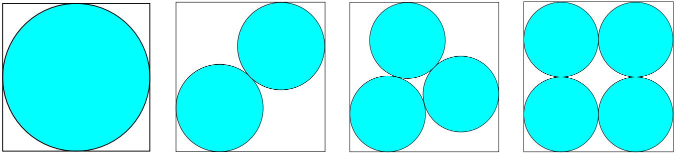
Techniques

- Weighting system to obtain approximation factors
- Specific algorithms to deal with big and small circles
- Grouping circles to consider as equal circles
- Geometric Partition to combine items of the same type

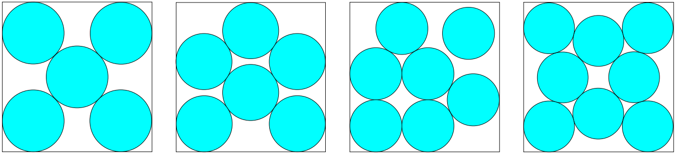
Bounded Space Online Bin Packing

Packing equal circles

Find the largest ρ^* st. k circles of radius ρ^* can be packed in a unit square



$\rho_1^* = 0.5$ $\rho_2^* = 0.2928$ $\rho_3^* = 0.2543$ $\rho_4^* = 0.1963$

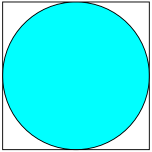


$\rho_5^* = 0.2071$ $\rho_6^* = 0.1876$ $\rho_7^* = 0.1744$ $\rho_8^* = 0.1705$

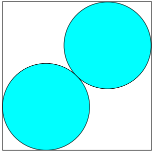
Previous results: It is known the exact values of ρ_n^*, for $n \leq 30$ and good lower bounds for many.

Packing equal circles

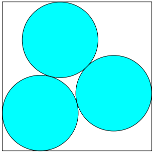
Find the largest ρ^* st. k circles of radius ρ^* can be packed in a unit square



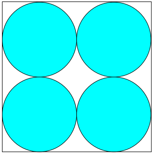
$\rho_1^* = 0.5$



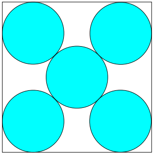
$\rho_2^* = 0.2928$



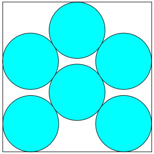
$\rho_3^* = 0.2543$



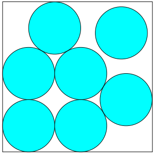
$\rho_4^* = 0.1963$



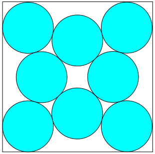
$\rho_5^* = 0.2071$



$\rho_6^* = 0.1876$



$\rho_7^* = 0.1744$



$\rho_8^* = 0.1705$

Previous results: It is known the exact values of ρ_n^*, for $n \leq 30$ and good lower bounds for many.

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} < 1/M \leq \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

Round up big circles to the nearest value of ρ
(to bound the number of different circles)

- A circle is big if its radius is larger than $1/M$
- Let ρ_i be the value of ρ_i^*, when it is known, otherwise, the best known lower bound.
- Let K be such that $\rho_{K+1} \leq 1/M < \rho_K$

A circle r is of type i if:

- $\rho_{i+1} < r \leq \rho_i$ (for $1 \leq i < K$)
- $1/M < r \leq \rho_K$ (for $i = K$)

Packing big circles

- For $1 \leq i \leq K$, a *c-bin* of type i is a circular bin of radius ρ_i
- Circles of type i are packed in a c-bin of type i
- Packing in a c-bins of type 2

Packing big circles

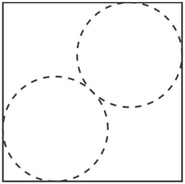
- For $1 \leq i \leq K$, a *c-bin* of type i is a circular bin of radius ρ_i
- Circles of type i are packed in a c-bin of type i
- Packing in a c-bins of type 2

Packing big circles

- For $1 \leq i \leq K$, a *c-bin* of type i is a circular bin of radius ρ_i
- Circles of type i are packed in a c-bin of type i
- Packing in a c-bins of type 2

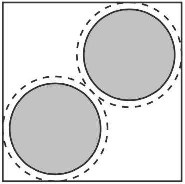
Packing big circles

- For $1 \leq i \leq K$, a *c-bin* of type i is a circular bin of radius ρ_i
- Circles of type i are packed in a c-bin of type i
- Packing in a c-bins of type 2



Packing big circles

- For $1 \leq i \leq K$, a *c-bin* of type i is a circular bin of radius ρ_i
- Circles of type i are packed in a c-bin of type i
- Packing in a c-bins of type 2



Algorithm - Part 1

To pack a big circle c of type i :

if there is no empty c-bin of type i

close the current bin of type i (if any)

open a new bin of type i containing i c-bins of type i

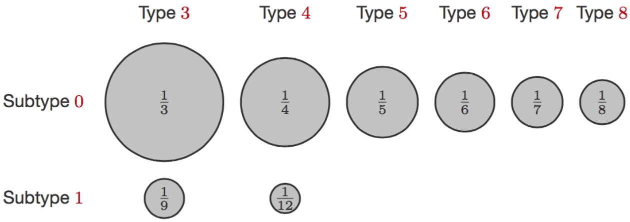
Pack c into a empty c-bin of type i

Small circles

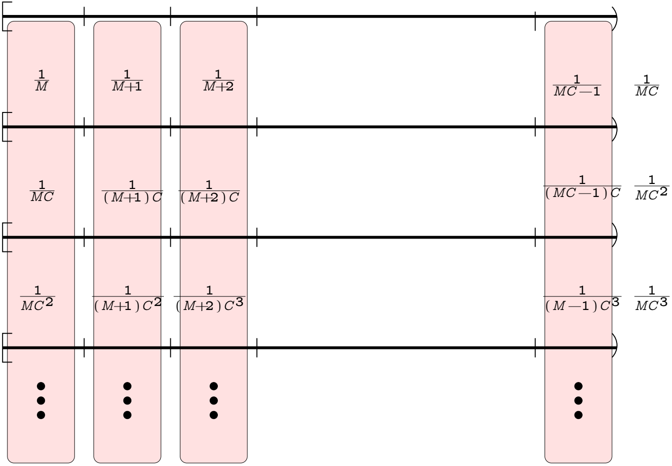
Let $C > 0$ be an integer multiple of 3

A small circle of radius r is of type i, subtype k if

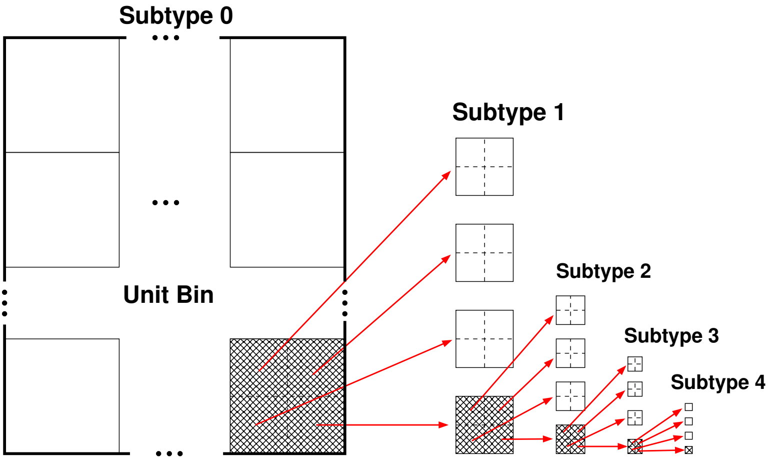
- $1/(i+1) < C^k r \leq 1/i$
- where k is the largest integer such that $C^k r \leq 1/M$
- and the circle is said to be of type (i, k)



Small circles



Small circles



Subdivisions within a same type

Sub-bins (h-bins and t-bins)

Idea: Round/Pack small circles into hexagonal bins

h-bin of type (i, k):

- hexagonal bin of side length $2/(\sqrt{3}\, C^k i)$
- Receives a small circle of type (i, k)

t-bin of type (i, k):

- trapezoidal bin obtained by the subdivision of a h-bin of type (i, k) in the center

Sub-bins (h-bins and t-bins)

Idea: Round/Pack small circles into hexagonal bins

h-bin of type (i, k):

- hexagonal bin of side length $2/(\sqrt{3}\, C^k i)$
- Receives a small circle of type (i, k)

t-bin of type (i, k):

- trapezoidal bin obtained by the subdivision of a h-bin of type (i, k) in the center

Sub-bins (h-bins and t-bins)

Idea: Round/Pack small circles into hexagonal bins

h-bin of type (i, k):

- hexagonal bin of side length $2/(\sqrt{3}\,C^k i)$
- Receives a small circle of type (i, k)

t-bin of type (i, k):

- trapezoidal bin obtained by the subdivision of a h-bin of type (i, k) in the center

Sub-bins (h-bins and t-bins)

Idea: Round/Pack small circles into hexagonal bins

h-bin of type (i, k):

- hexagonal bin of side length $2/(\sqrt{3}\, C^k i)$
- Receives a small circle of type (i, k)

t-bin of type (i, k):

- trapezoidal bin obtained by the subdivision of a h-bin of type (i, k) in the center

Sub-bins (h-bins and t-bins)

Idea: Round/Pack small circles into hexagonal bins

h-bin of type (i, k):

- hexagonal bin of side length $2/(\sqrt{3}\, C^k i)$
- Receives a small circle of type (i, k)

t-bin of type (i, k):

- trapezoidal bin obtained by the subdivision of a h-bin of type (i, k) in the center

Sub-bins (h-bins and t-bins)

Idea: Round/Pack small circles into hexagonal bins

h-bin of type (i, k):

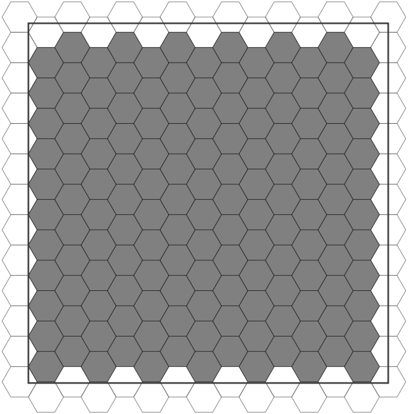
- hexagonal bin of side length $2/(\sqrt{3}\, C^k i)$
- Receives a small circle of type (i, k)

t-bin of type (i, k):

- trapezoidal bin obtained by the subdivision of a h-bin of type (i, k) in the center

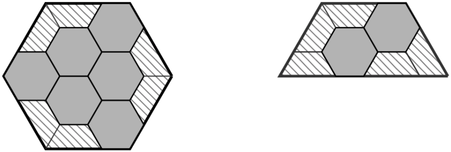
Subdividing a square into h-bins

Subdividing a square into h-Bins



Partitioning sub-bins

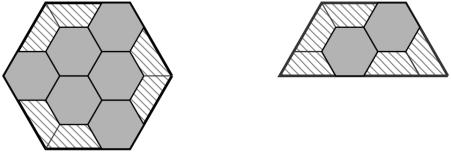
For all $M \leq i < CM$ and $k \geq 0$, if C is multiple of 3 then, it is possible to partition an h-bin or an t-bin) of type (i, k) into h-bins and t-bins of type $(i, k+1)$.



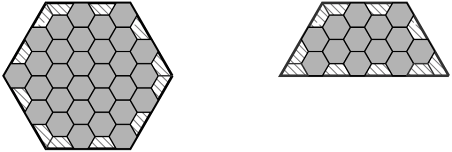
$C = 3$

Partitioning sub-bins

For all $M \leq i < CM$ and $k \geq 0$, if C is multiple of 3 then, it is possible to partition an h-bin or an t-bin) of type (i,k) into h-bins and t-bins of type $(i,k+1)$.



$C = 3$



$C = 6$

Algorithm - Part 2

When a small circle c of type (i, k) arrives:

if there is no empty h-bin of type (i, k) or an empty sub-bin of type (i, k') with $k' < k$

close the current bin of type i (if any)

open a bin of type i subdividing into h-bins of type $(i, 0)$

while there is no h-bin of type (i, k)

let k' the largest number such that $k' < k$ and there exists an en

of type (i, k')

if there exists an empty t-bin of type (i, k')

B tal t-bin

else

let B an h-bin of type (i, k')

particionate B in sub-bins of type $(i, k' + 1)$

packs c into a h-bin of type (i, k)

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function w: $L \rightarrow \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1
  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore
  $$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$
- Find $\alpha \geq$ *maximum bin weight*. I.e.,
  $$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$
  Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha \, \mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:
$$\mathcal{A}(L) \leq w(L) \leq \alpha \, \mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function w: $L \to \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1
  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore

$$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$

- Find $\alpha \geq$ *maximum bin weight*. I.e.,

$$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$

  Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha\,\mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:

$$\mathcal{A}(L) \leq w(L) \leq \alpha\,\mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function w: $L \rightarrow \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1
  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore

$$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$

- Find $\alpha \geq$ *maximum bin weight*. I.e.,

$$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$

Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha \,\mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:

$$\mathcal{A}(L) \leq w(L) \leq \alpha \,\mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function $w\colon L \to \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1
  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore
  $$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$
- Find $\alpha \geq$ *maximum bin weight*. I.e.,
  $$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$

Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha\,\mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:

$$\mathcal{A}(L) \leq w(L) \leq \alpha\,\mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function $w \colon L \to \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1
  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore

$$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$

- Find $\alpha \geq$ *maximum bin weight*. I.e.,

$$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$

Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha\, \mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:

$$\mathcal{A}(L) \leq w(L) \leq \alpha\, \mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function $w\colon L \to \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1
  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore

$$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$

- Find $\alpha \geq$ *maximum bin weight*. I.e.,

$$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$

Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha\,\mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:

$$\mathcal{A}(L) \leq w(L) \leq \alpha\,\mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Analysis by weighting function

Given algorithm $\mathcal{A}$ and weight function $w \colon L \to \mathbb{R}_{\geq 0}$ st.

- $\mathcal{A}$ produce bins with average weight at least 1

  I.e., $w(L)/\mathcal{A}(L) \geq 1$ and therefore

$$\mathcal{A}(L) \leq w(L), \quad \text{for any instance } L$$

- Find $\alpha \geq$ *maximum bin weight*. I.e.,

$$\alpha \geq \sup\{w(S) : S \subseteq L \text{ and } \exists \text{ packing of } S \text{ in one bin}\}$$

  Optimum uses at least $\frac{w(L)}{\alpha}$ bins: $w(L) \leq \alpha\,\mathrm{OPT}$

Algorithm $\mathcal{A}$ has approximation factor α:

$$\mathcal{A}(L) \leq w(L) \leq \alpha\,\mathrm{OPT}(L)$$

Removing *few* bins, before average leads to asymptotic factor

Circle weights

How to obtain average weight ≥ 1 ?

- Algorithm produce bins with weight ≥ 1

$$w(c) = \begin{cases} 1/i & \text{if } c \text{ is big and type } i \\ \text{Area}(c)/\gamma & \text{if } c \text{ is a small circle,} \end{cases}$$

where γ is area density or lower bound, for bins with small

- If B is closed type i bin (big items) then B has i circles of weight $1/i$ and $w(B) = 1$.
- If B is closed bin for small circles, then Area(B) is also its density and Area$(B) \geq \gamma$. So

$$w(B) = \sum_{c \in B} w(c) = \sum_{c \in B} \frac{\text{Area}(c)}{\gamma} \geq 1$$

Circle weights

How to obtain average weight ≥ 1 ?

- Algorithm produce bins with weight ≥ 1

$$w(c) = \begin{cases} 1/i & \text{if } c \text{ is big and type } i \\ \text{Area}(c)/\gamma & \text{if } c \text{ is a small circle,} \end{cases}$$

where γ is area density or lower bound, for bins with small

- If B is closed type i bin (big items) then B has i circles of weight $1/i$ and $w(B) = 1$.
- If B is closed bin for small circles, then $\text{Area}(B)$ is also its density and $\text{Area}(B) \geq \gamma$. So

$$w(B) = \sum_{c \in B} w(c) = \sum_{c \in B} \frac{\text{Area}(c)}{\gamma} \geq 1$$

Circle weights

How to obtain average weight ≥ 1 ?

- Algorithm produce bins with weight ≥ 1

$$w(c) = \begin{cases} 1/i & \text{if } c \text{ is big and type } i \\ \text{Area}(c)/\gamma & \text{if } c \text{ is a small circle,} \end{cases}$$

where γ is area density or lower bound, for bins with small

- If B is closed type i bin (big items) then B has i circles of weight $1/i$ and $w(B) = 1$.
- If B is closed bin for small circles, then $\text{Area}(B)$ is also its density and $\text{Area}(B) \geq \gamma$. So

$$w(B) = \sum_{c \in B} w(c) = \sum_{c \in B} \frac{\text{Area}(c)}{\gamma} \geq 1$$

Circle weights

How to obtain average weight ≥ 1 ?

- Algorithm produce bins with weight ≥ 1

$$w(c) = \begin{cases} 1/i & \text{if } c \text{ is big and type } i \\ \text{Area}(c)/\gamma & \text{if } c \text{ is a small circle,} \end{cases}$$

where γ is area density or lower bound, for bins with small

- If B is closed type i bin (big items) then
  B has i circles of weight $1/i$ and $w(B) = 1$.

- If B is closed bin for small circles, then
  $\text{Area}(B)$ is also its density and $\text{Area}(B) \geq \gamma$. So

$$w(B) = \sum_{c \in B} w(c) = \sum_{c \in B} \frac{\text{Area}(c)}{\gamma} \geq 1$$

Circle weights

How to obtain average weight ≥ 1 ?

- Algorithm produce bins with weight ≥ 1

$$w(c) = \begin{cases} 1/i & \text{if } c \text{ is big and type } i \\ \text{Area}(c)/\gamma & \text{if } c \text{ is a small circle,} \end{cases}$$

  where γ is area density or lower bound, for bins with small

- If B is closed type i bin (big items) then
  B has i circles of weight $1/i$ and $w(B) = 1$.
- If B is closed bin for small circles, then
  $\text{Area}(B)$ is also its density and $\text{Area}(B) \geq \gamma$. So

$$w(B) = \sum_{c \in B} w(c) = \sum_{c \in B} \frac{\text{Area}(c)}{\gamma} \geq 1$$

Circle weights

How to obtain average weight ≥ 1 ?

- Algorithm produce bins with weight ≥ 1

$$w(c) = \begin{cases} 1/i & \text{if } c \text{ is big and type } i \\ \text{Area}(c)/\gamma & \text{if } c \text{ is a small circle,} \end{cases}$$

where γ is area density or lower bound, for bins with small

- If B is closed type i bin (big items) then
  B has i circles of weight $1/i$ and $w(B) = 1$.
- If B is closed bin for small circles, then
  $\text{Area}(B)$ is also its density and $\text{Area}(B) \geq \gamma$. So

$$w(B) = \sum_{c \in B} w(c) = \sum_{c \in B} \frac{\text{Area}(c)}{\gamma} \geq 1$$

Circle weights

γ: lower bound for area covered in closed bins by small items

The non-covered regions are due to:

- $\mathcal{L}_B$: upper bound for the non-covered region due to the shape and partial intersection of hexagons with the border of the square bin, and is at most $5.89/M$
- $\mathcal{L}_F$: upper bound for to the set of non-covered hexagons when a bin is closed: $2\sqrt{3}C^2/(M^2(C^2-1))$
- $\mathcal{L}_H$: loss factor due to the *rounding* of circles into hexagons: $\frac{\pi}{\sqrt{12}}\frac{M^2}{(M+1)^2}$

That is

$$\gamma = (1 - \mathcal{L}_B - \mathcal{L}_F)\,\mathcal{L}_H$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$
\begin{aligned}
\text{maximize} \quad & \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i} \\
\text{subject to} \quad & y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \leq 1 \\
& x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \leq i \leq K \\
& y \geq 0
\end{aligned}
$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$\begin{aligned} \text{maximize} \quad & \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i} \\ \text{subject to} \quad & y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \leq 1 \\ & x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \leq i \leq K \\ & y \geq 0 \end{aligned}$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$
\begin{aligned}
\text{maximize} \quad & \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i} \\
\text{subject to} \quad & y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \leq 1 \\
& x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \leq i \leq K \\
& y \geq 0
\end{aligned}
$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$\text{maximize } \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i}$$

$$\text{subject to } \quad y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \leq 1$$

$$x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \leq i \leq K$$

$$y \geq 0$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$
\begin{aligned}
\text{maximize} \quad & \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i} \\
\text{subject to} \quad & y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \le 1 \\
& x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \le i \le K \\
& y \ge 0
\end{aligned}
$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$
\begin{aligned}
\text{maximize} \quad & \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i} \\
\text{subject to} \quad & y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \leq 1 \\
& x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \leq i \leq K \\
& y \geq 0
\end{aligned}
$$

Computing β

Value of β is obtained via Mixed Integer Programming:

- x_i: number of circles of type i
- y: area of small circles

$$
\begin{aligned}
\text{maximize } & \frac{y}{\alpha} + \sum_{i=1}^{K} \frac{x_i}{i} \\
\text{subject to } \quad & y + \sum_{i=1}^{K} \pi \rho_{i+1}^2 x_i \le 1 \\
& x_i \in \mathbb{Z}_+ \qquad \forall\, 1 \le i \le K \\
& y \ge 0
\end{aligned}
$$

Computing β

On the other hand, we do not know if a solution can indeed be packed in one bin

- Using Constraint Programing to verify if a solution can be packed in only one bin, with time limit
- If it is not possible, we add a constraint in the model to avoid such solution

For $M = 59$ and $K = 992$, the value of β is 2.4394

- But we do not know if the solution can in fact be packed in only one bin

Computing β

On the other hand, we do not know if a solution can indeed be packed in one bin

- Using Constraint Programing to verify if a solution can be packed in only one bin, with time limit
- If it is not possible, we add a constraint in the model to avoid such solution

For $M = 59$ and $K = 992$, the value of β is 2.4394

- But we do not know if the solution can in fact be packed in only one bin

Computing β

On the other hand, we do not know if a solution can indeed be packed in one bin

- Using Constraint Programing to verify if a solution can be packed in only one bin, with time limit
- If it is not possible, we add a constraint in the model to avoid such solution

For $M = 59$ and $K = 992$, the value of β is 2.4394

- But we do not know if the solution can in fact be packed in only one bin

Computing β

On the other hand, we do not know if a solution can indeed be packed in one bin

- Using Constraint Programing to verify if a solution can be packed in only one bin, with time limit
- If it is not possible, we add a constraint in the model to avoid such solution

For $M = 59$ and $K = 992$, the value of β is 2.4394

- But we do not know if the solution can in fact be packed in only one bin

Computing β

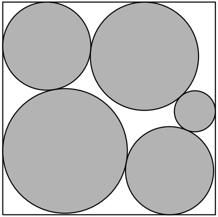
On the other hand, we do not know if a solution can indeed be packed in one bin

- Using Constraint Programing to verify if a solution can be packed in only one bin, with time limit
- If it is not possible, we add a constraint in the model to avoid such solution

For $M = 59$ and $K = 992$, the value of β is 2.4394

- But we do not know if the solution can in fact be packed in only one bin

Lower bound for any competitive factor



- 1 circle of type 1
- 1 circle of type 2
- 2 circle of type 4
- 1 circle of type 25

  (the area covered by the above circles: 0.77139)
- remaining space is completed with sand
  (very small circles, non-necessarily equal)

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower Bound

Consider N copies of the lower bound pattern with circles sorted by radius

An online bounded space algorithm B uses:

- at least $N - B$ bins for the circles of type 1
- at least $N/2 - 2B$ bins for the circles of type 2
- at least $2N/4 - 2B$ bins for the circles of type4
- at least $N/25 - 2B$ bins for the circles of type25

At least $2.04N - 7B$ for the circles

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with density $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Lower bound

Let $S = (1 - 0.77139 - \varepsilon)$ the remaining area used by sand

- The best way to obtain a dense packing of equal circles is the hexagonal packing
  - with densities $\pi/\sqrt{12}$
- Even for very small circles, the algorithm cannot do better than the hexagonal packing
- The algorithm uses at least $S\sqrt{12}/\pi - 2kB$ bins
  - k is the number of different radius

The algorithm uses at least $2.2920N - \delta N - O(1)$ bins, that tends to $2.2920 - \delta$ when N goes to infinity

An offline algorithm uses at most N bins

Exercises

- Obtain bounded online approximation algorithms to pack items into bins, each one could be one of the following: equilateral triangles, squares, circles, hexagons, etc.
- For the previous exercise, consider the three-dimensional or d-dimensional case.

THANKS!

QUESTIONS?