Universidad de Oviedo

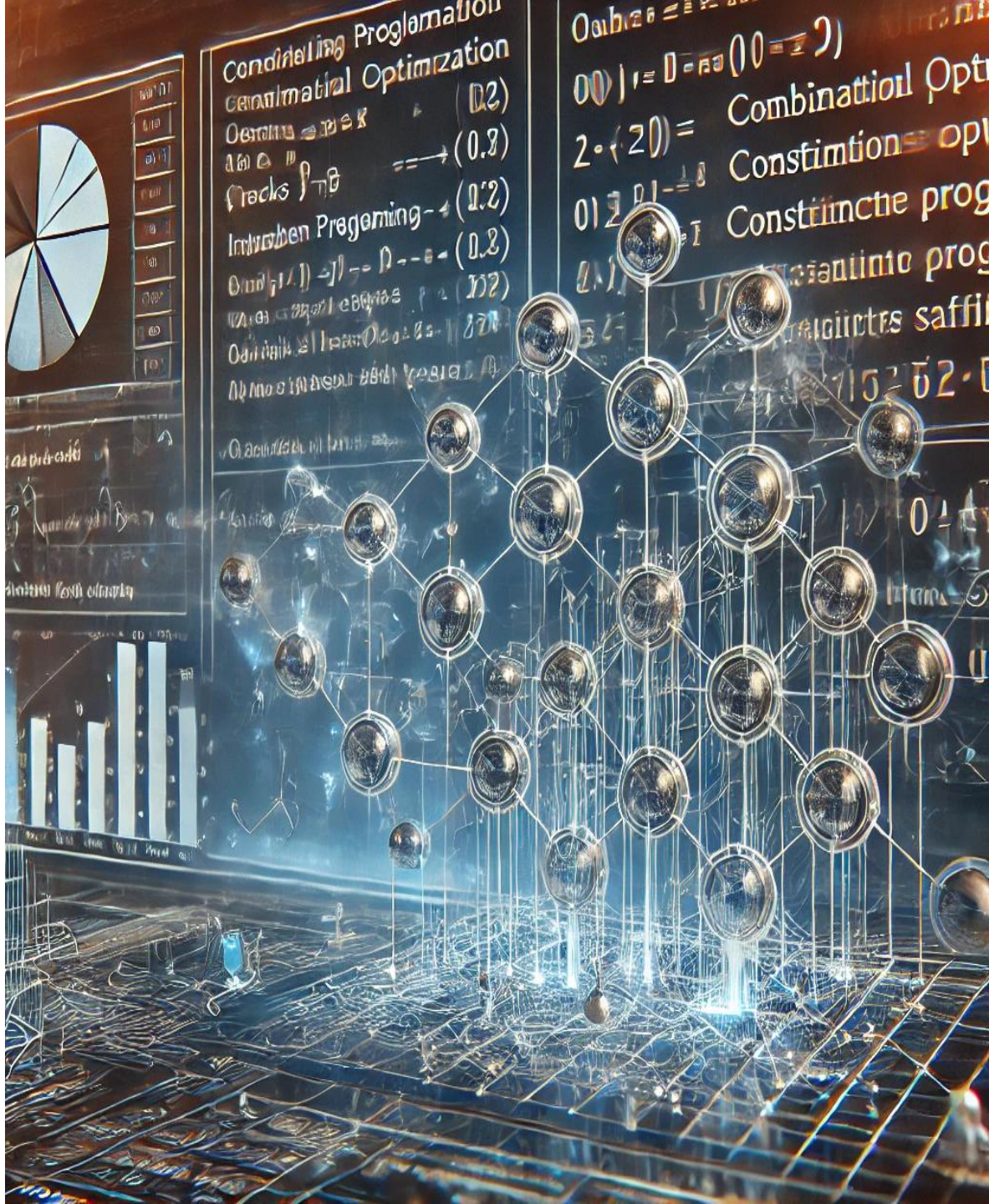
Intelligent Scheduling and Optimization

Técnicas de Inteligencia Artificial para la Optimización y Programación de Recursos

Aplicaciones de Scheduling en la vida real I

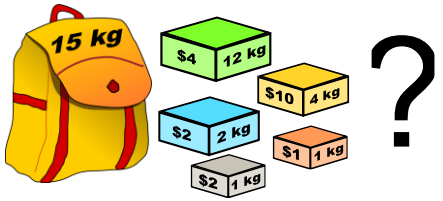
Jesús Quesada Matilla
{quesadajesus}@uniovi.es

Ciencia de la Computación e Inteligencia Artificial
Departamento de Informática



Aplicaciones de Scheduling en la vida real

- Problema de la mochila (knapsack problem)



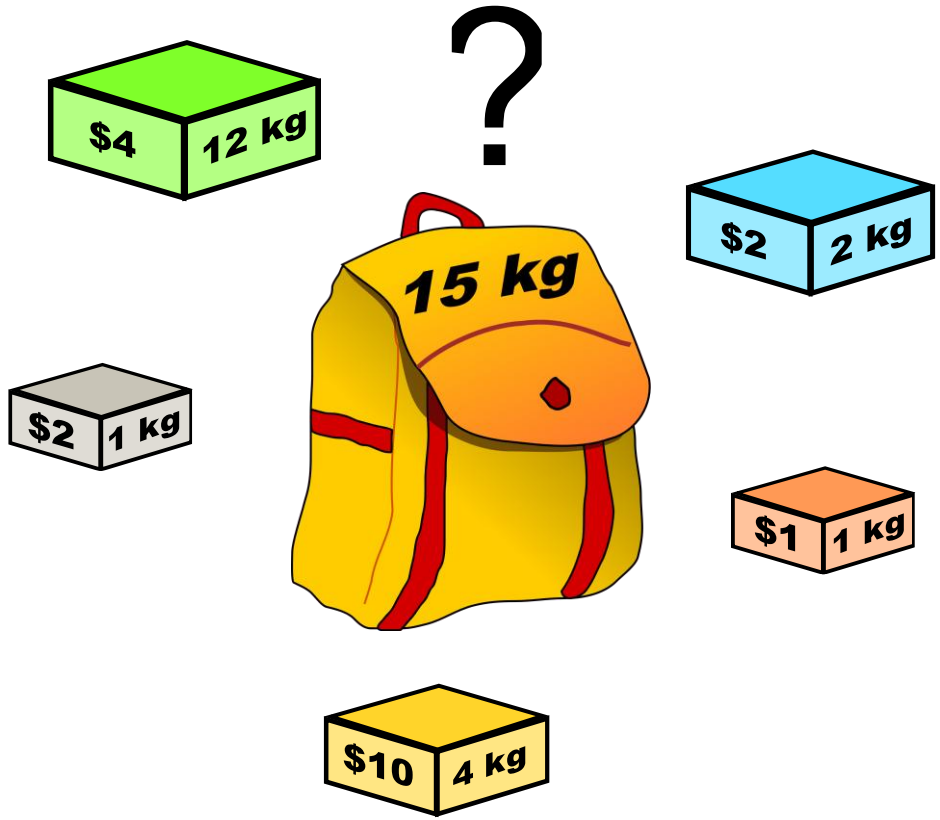
?

- Problema de empaquetado (Bin Packing Problem)

- Problema de corte de valores (Cutting Stock Problem)

El problema de la mochila

- Introducir la mayor cantidad de valor en una mochila sin exceder el peso máximo que se puede llevar
- Objetivo: maximizar el valor de la mochila
- Restricción: no superar el peso máximo



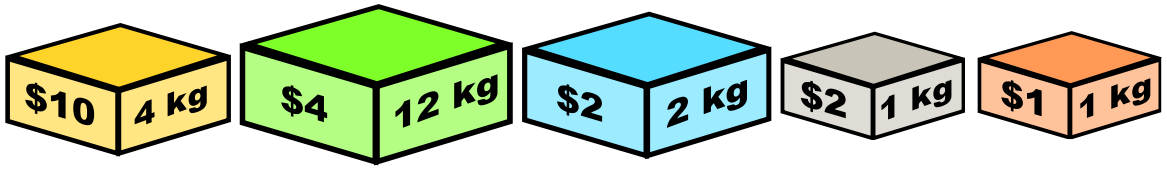
"Illustration of the knapsack problem" de Dake, disponible en https://es.wikipedia.org/wiki/Archivo:Knapsack.svg, bajo licencia Creative Commons Atribución/Compartir-Igual 2.5 (ver licencia en https://creativecommons.org/licenses/by-sa/2.5/deed.es). Se realizaron cambios.

El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $0

PesoTotal: 0kg

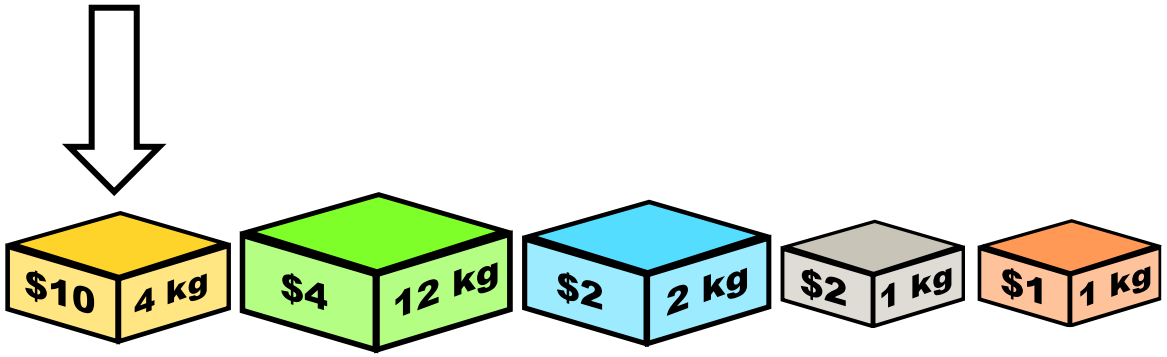


El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $0

PesoTotal: 0kg

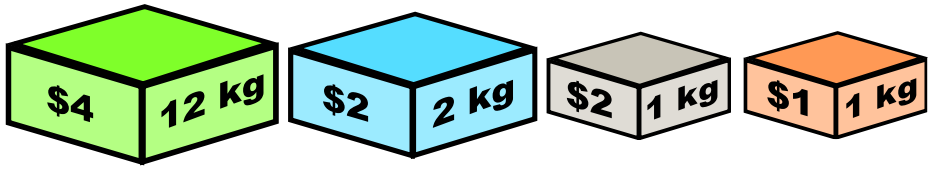


El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $10

PesoTotal: 4kg

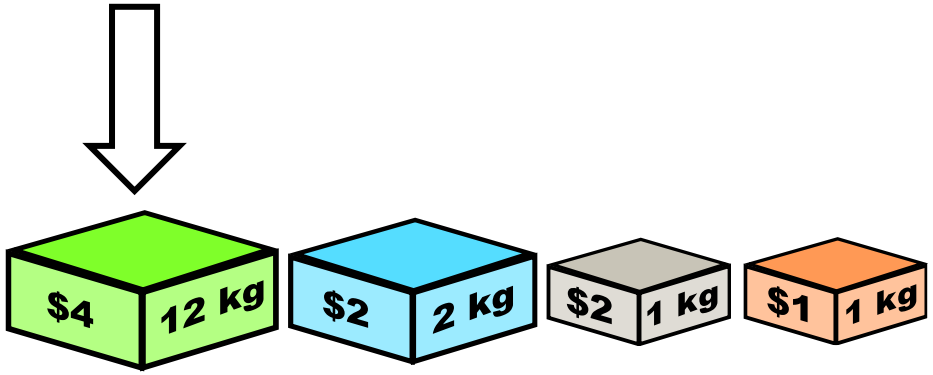


El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $10

PesoTotal: 4kg



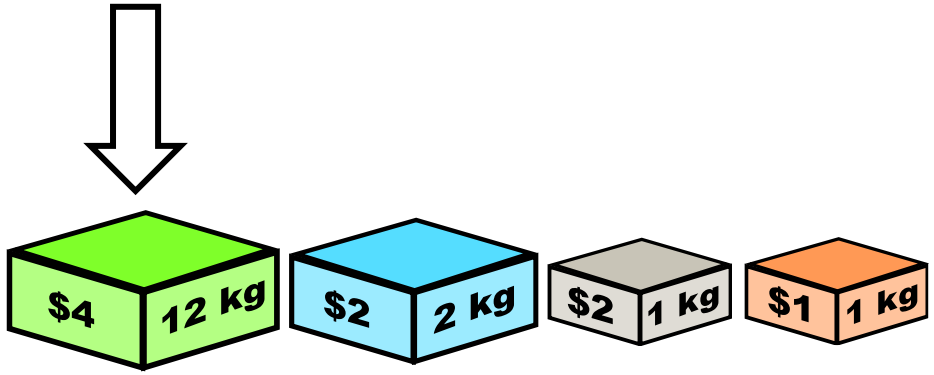
El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $10

PesoTotal: 4kg

¡No cabe! Lo omitimos

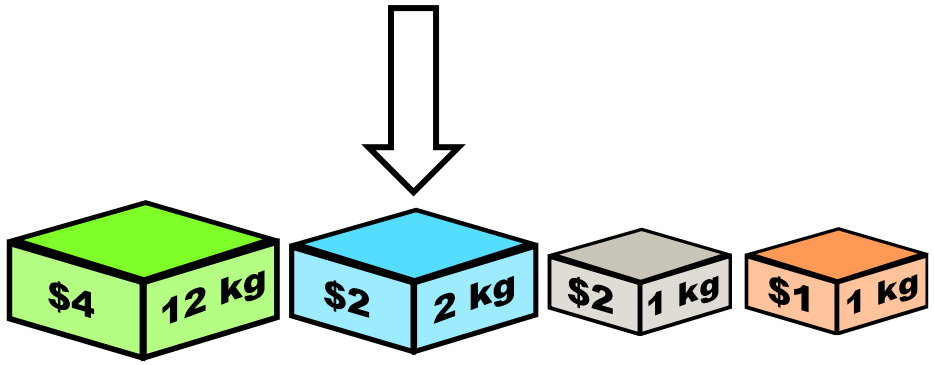


El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $10

PesoTotal: 4kg

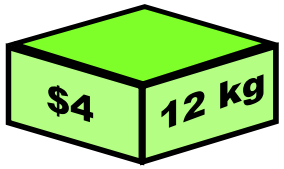


El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $12

PesoTotal: 6kg

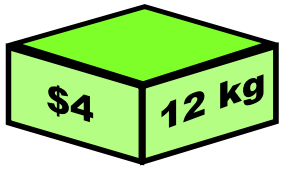


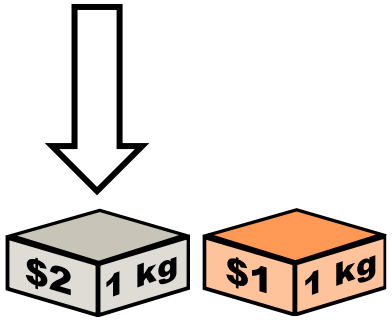
El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $12

PesoTotal: 6kg



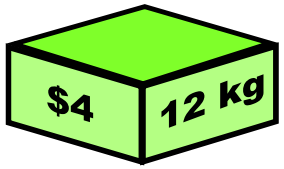


El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $14

PesoTotal: 7kg

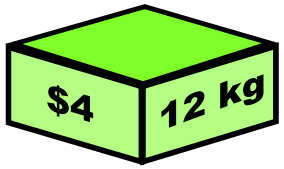


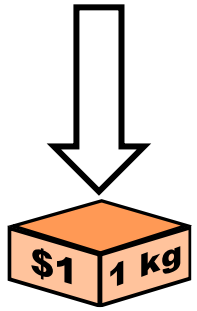
El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila

ValorTotal: $14

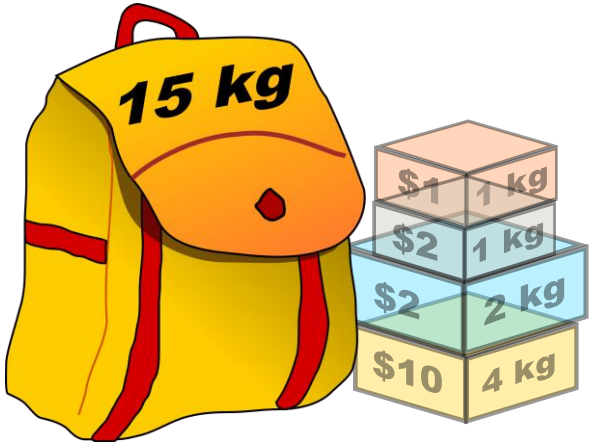
PesoTotal: 7kg





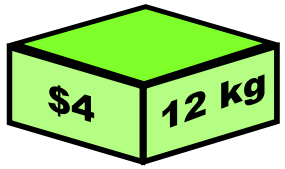
El problema de la mochila

- Vamos a generar una solución
- Empaquetamos el objeto de mayor **valor** que quepa en la mochila



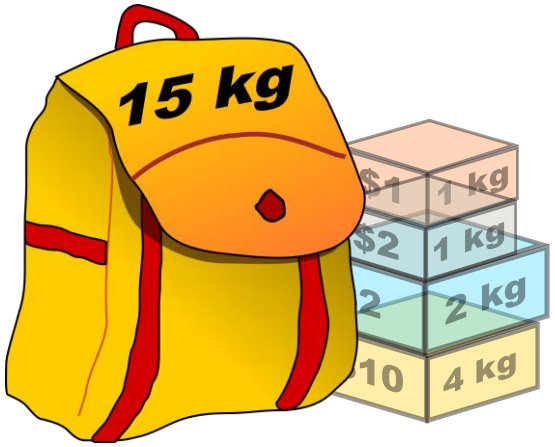
ValorTotal: $15

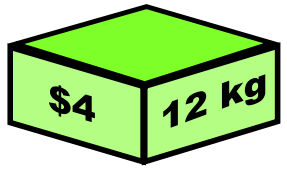
PesoTotal: 8kg



El problema de la mochila

- Solución #1 Valor: $15 Peso: 8kg





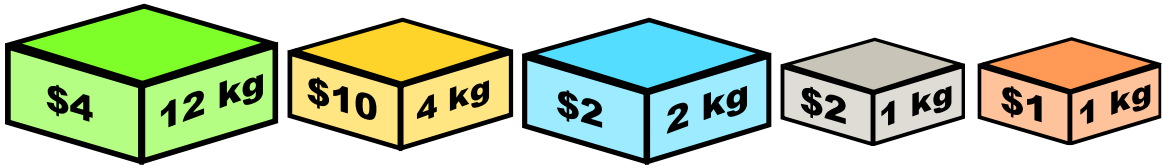
Empaquetamos el objeto de mayor **valor** que quepa en la mochila

El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila

ValorTotal: $0

PesoTotal: 0kg

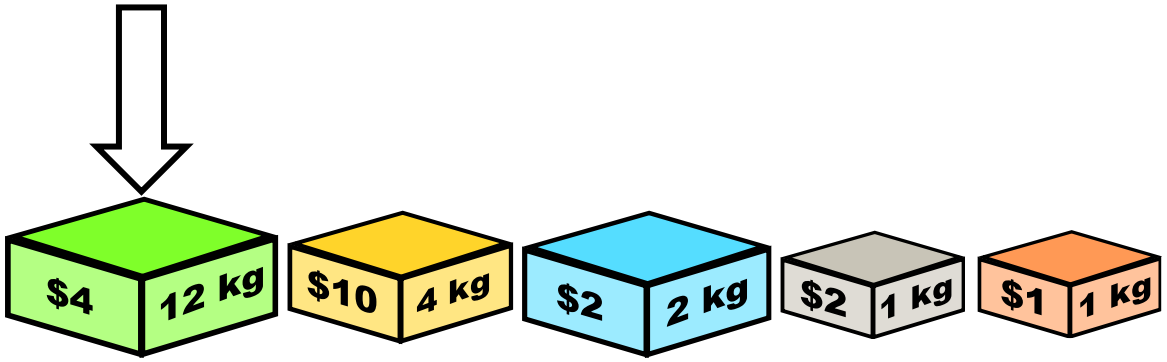


El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila

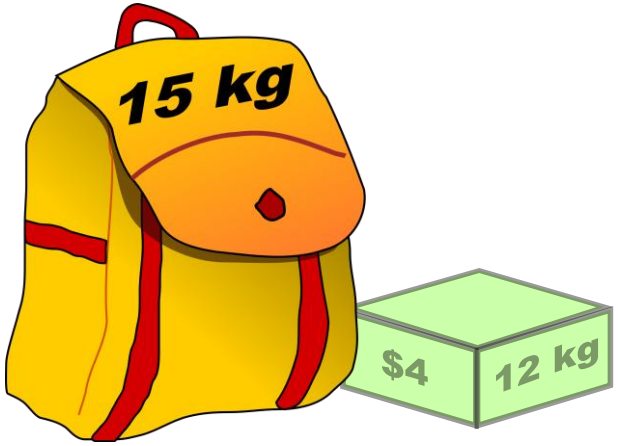
ValorTotal: $0

PesoTotal: 0kg



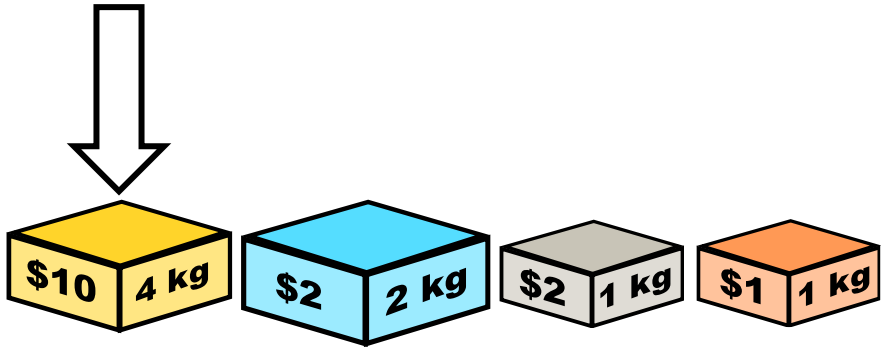
El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila



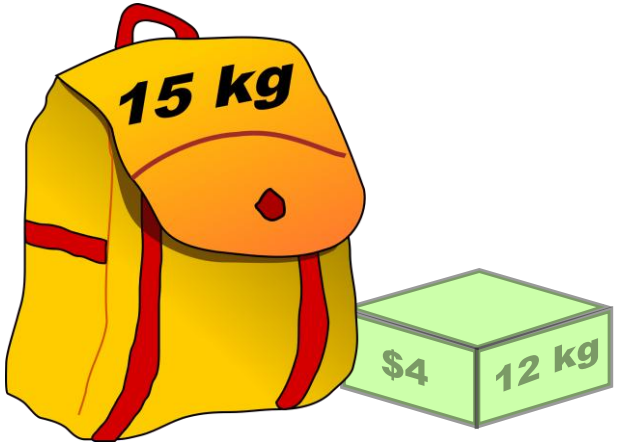
ValorTotal: $4

PesoTotal: 12kg



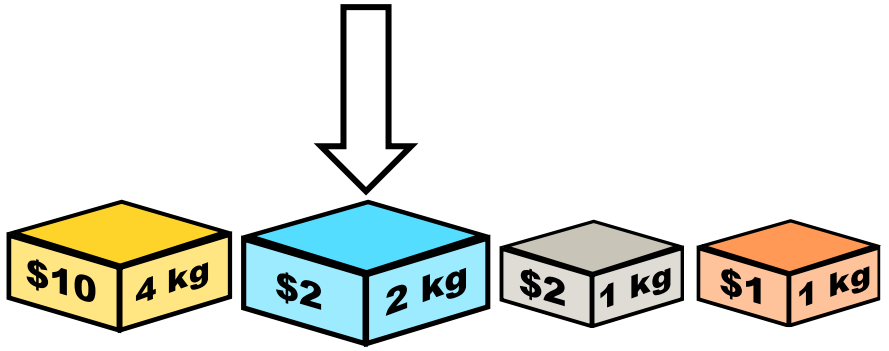
El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila



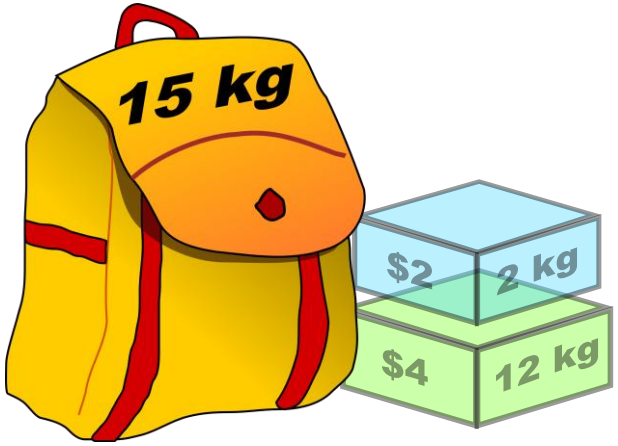
ValorTotal: $4

PesoTotal: 12kg



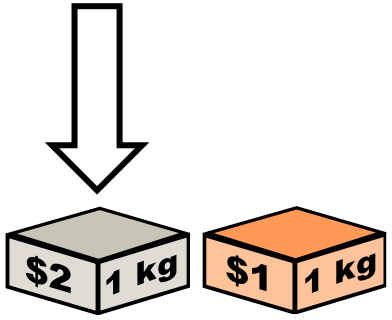
El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila



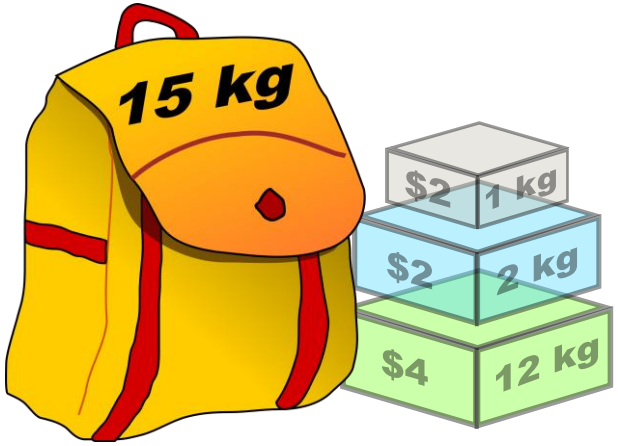
ValorTotal: $6

PesoTotal: 14kg



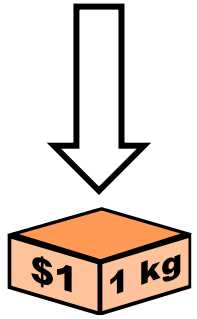
El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila



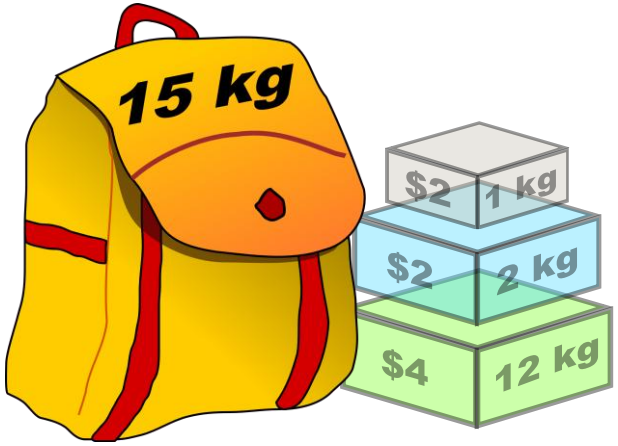
ValorTotal: $8

PesoTotal: 15kg



El problema de la mochila

- Vamos a generar otra solución
- Empaquetamos el objeto de mayor **peso** que quepa en la mochila

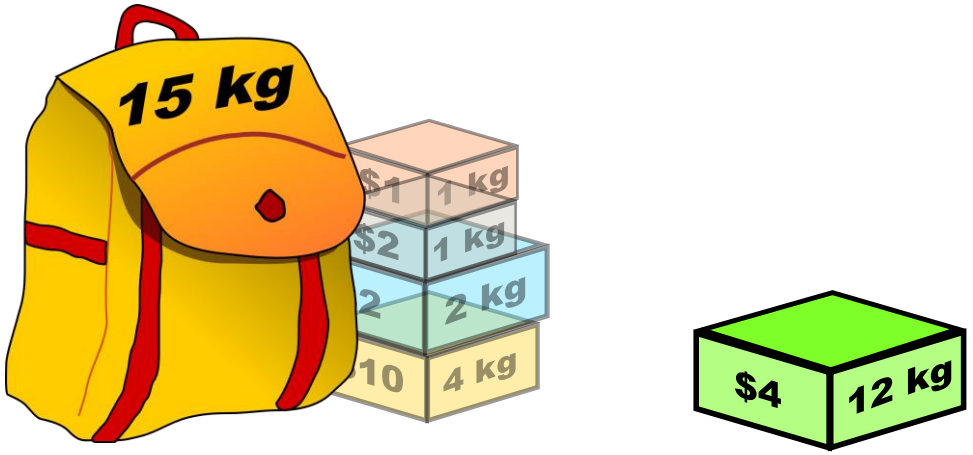


ValorTotal: $8

PesoTotal: 15kg

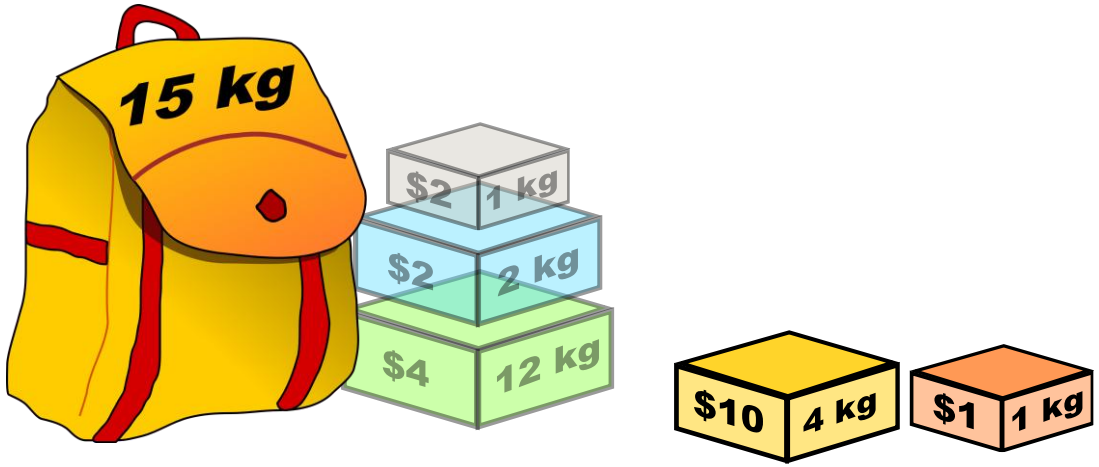
El problema de la mochila

- Solución #1 Valor: $15 Peso: 8kg



Empaquetamos el objeto de mayor **valor** que quepa en la mochila

- Solución #2 Valor: $8 Peso: 15kg

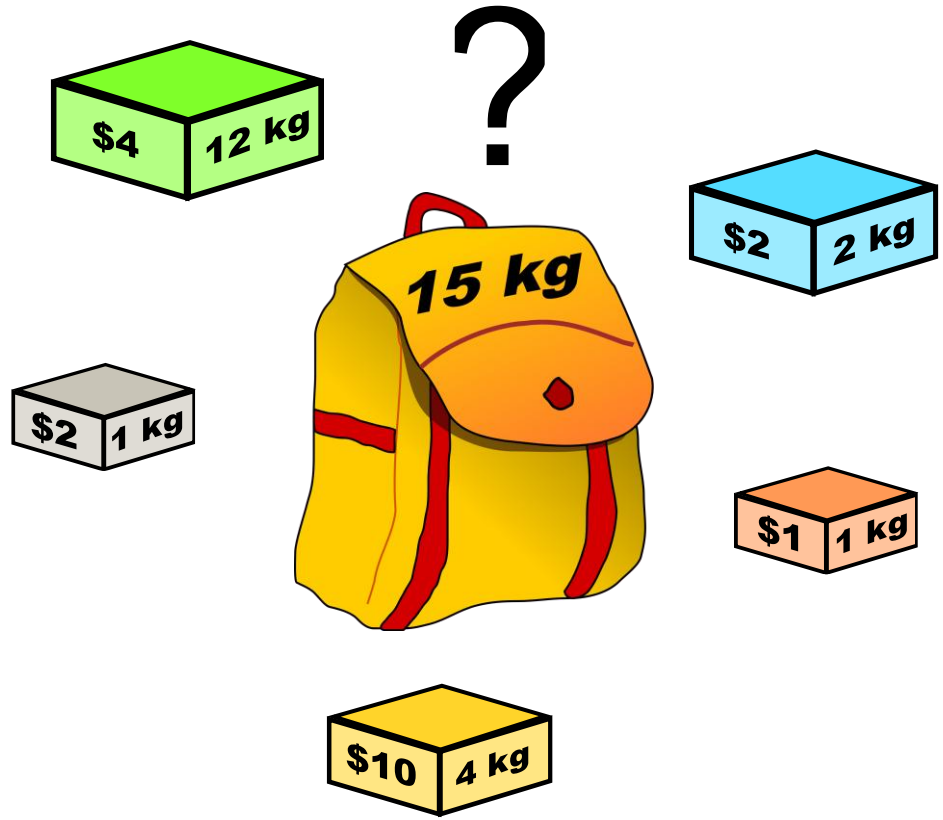


Empaquetamos el objeto de mayor **peso** que quepa en la mochila

El problema de la mochila

- ¿Qué pasaría si el objetivo fuese empaquetar todos los objetos?

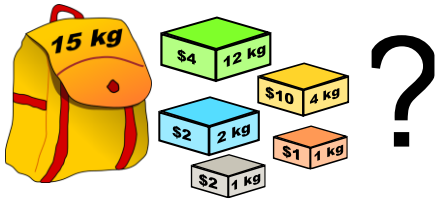
- Omitimos el valor de los objetos (todos van a ser empaquetados)
- Tenemos infinitas mochilas
- Objetivo: usar la menor cantidad de mochilas



"Illustration of the knapsack problem" de Dake, disponible en https://es.wikipedia.org/wiki/Archivo:Knapsack.svg, bajo licencia Creative Commons Atribución/Compartir-Igual 2.5 (ver licencia en https://creativecommons.org/licenses/by-sa/2.5/deed.es). Se realizaron cambios.

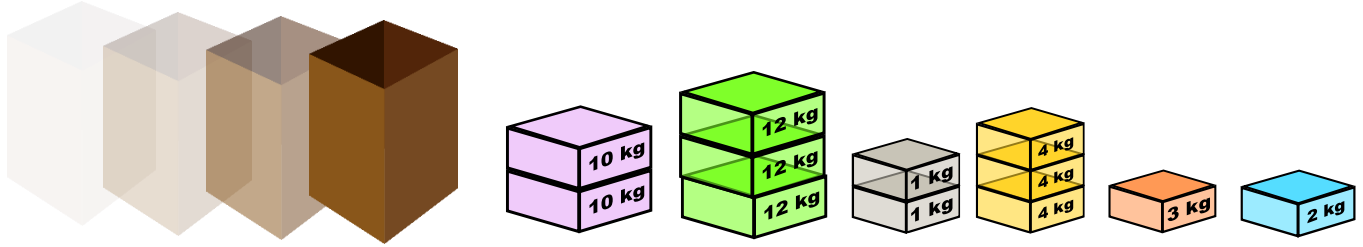
Aplicaciones de Scheduling en la vida real

- Problema de la mochila (knapsack problem)



?

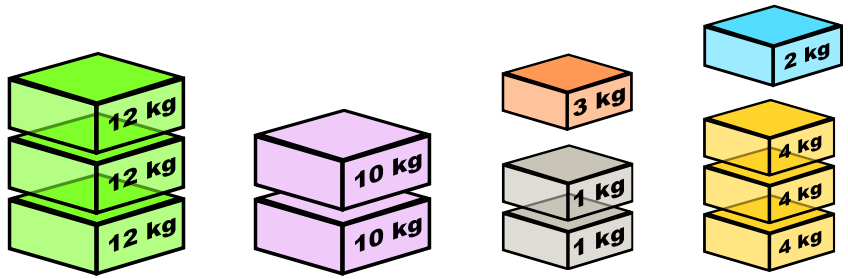
- Problema de empaquetado (Bin Packing Problem)



- Problema de corte de valores (Cutting Stock Problem)

Bin Packing Problem

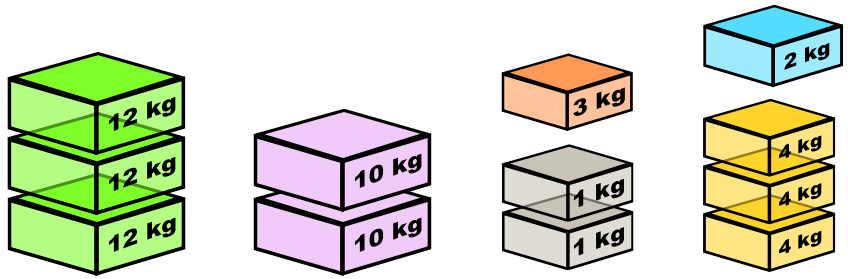
- Empaquetar un conjunto de ítems en la menor cantidad de contenedores posible



- Objetivo: usar la menor cantidad de contenedores
- Restricción: capacidad de los contenedores

Bin Packing Problem

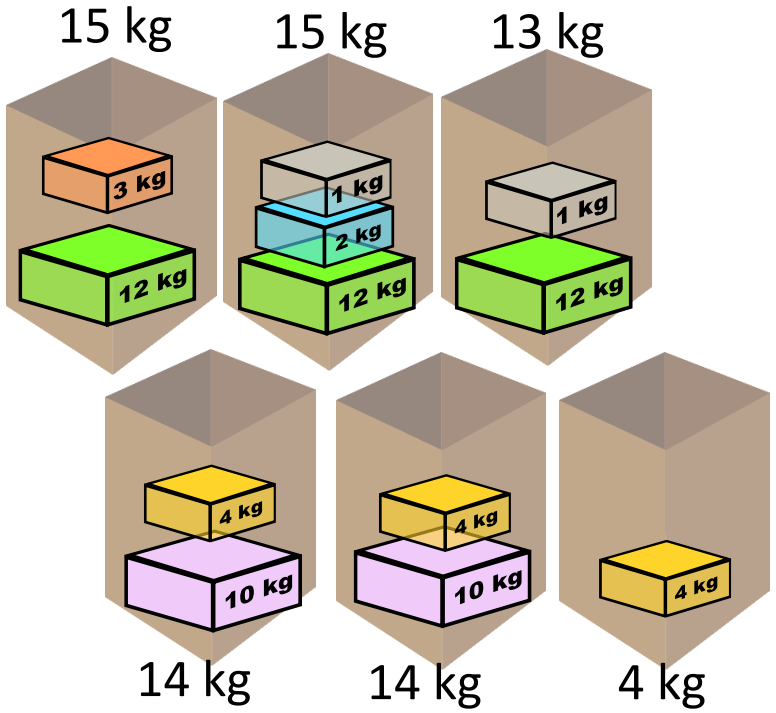
- Empaquetar un conjunto de ítems en la menor cantidad de contenedores posible



- Objetivo: usar la menor cantidad de contenedores
- Restricción: capacidad de los contenedores

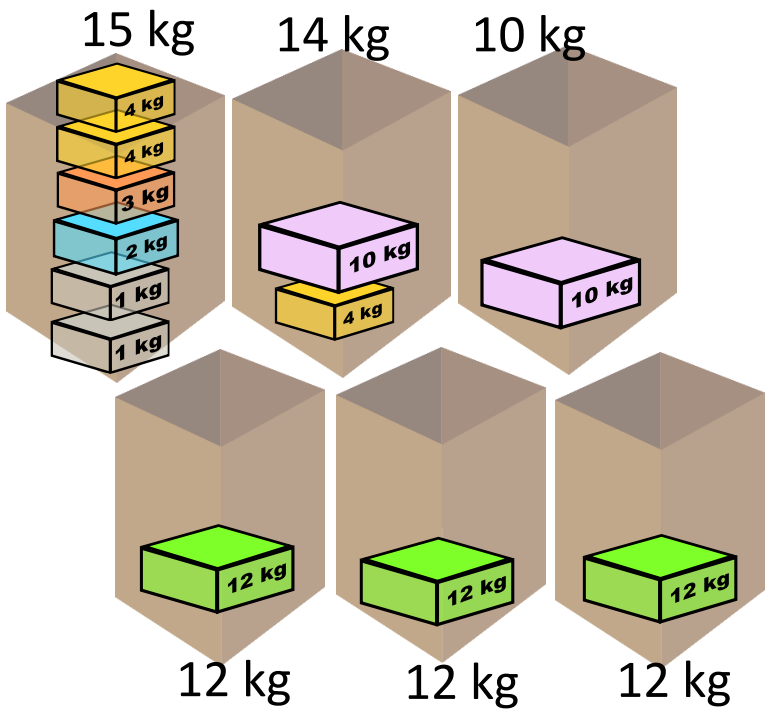
Bin Packing Problem

- Solución #1 N° Contenedores: 6



Empaquetamos el objeto de **mayor peso** que quepa en el contenedor

- Solución #2 N° Contenedores: 6



Empaquetamos el objeto de **menor peso** que quepa en el contenedor

Bin Packing Problem

- ¿Con qué métodos podemos resolver instancias?
- Algoritmo voraz guiado por heurístico
- Algoritmos genéticos
- Programación genética

```
// Instancia
15, 6  // Contenedores de tamaño 15, 6 pesos diferentes
12, 3  // De peso 12 hay 3 ítems
10, 2  // De peso 10 hay 2 ítems
4, 3   // ...
3, 1
2, 1
1, 2
```

Bin Packing Problem

- ¿Con qué métodos podemos resolver instancias?
- Algoritmo voraz guiado por heurístico
- **Algoritmos genéticos**
- Programación genética

```
// Instancia
15, 6  // Contenedores de tamaño 15, 6 pesos diferentes
12, 3  // De peso 12 hay 3 ítems
10, 2  // De peso 10 hay 2 ítems
4, 3   // ...
3, 1
2, 1
1, 2
```

Bin Packing Problem

- Genotipo
  - Un vector de ints
- Fenotipo
  - Un vector de ints
- Evaluador
  - Itera el vector como el orden en el que empaquetar los ítems

```
// Instancia
15, 6  // Contenedores de tamaño 15, 6 pesos diferentes
12, 3  // De peso 12 hay 3 ítems
10, 2  // De peso 10 hay 2 ítems
4, 3   // …
3, 1
2, 1
1, 2
```

Bin Packing Problem

- Un vector de ints
  - Puede ser los pesos en el orden de la instancia:

[12, 12, 12, 10, 10, 4, 4, 4, 3, 2, 1, 1]

  - Pueden ser identificadores de cada ítem:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

```
// Instancia
15, 6  // Contenedores de tamaño 15, 6 pesos diferentes
12, 3  // De peso 12 hay 3 ítems
10, 2  // De peso 10 hay 2 ítems
4, 3   // ...
3, 1
2, 1
1, 2
```

Bin Packing Problem

Identificadores de cada ítem:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] // Instancia

Datos del problema -> 15, 6 // Contenedores de tamaño 15, 6 pesos diferentes

0, 1, 2 -> 12, 3 // De peso 12 hay 3 ítems

3, 4 -> 10, 2 // De peso 10 hay 2 ítems

5, 6, 7 -> 4, 3 // ...

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

Identificadores de cada ítem:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

// Instancia

Datos del problema -> 15, 6 // Contenedores de tamaño 15, 6 pesos diferentes

0, 1, 2 -> 12, 3 // De peso 12 hay 3 ítems

3, 4 -> 10, 2 // De peso 10 hay 2 ítems

5, 6, 7 -> 4, 3 // ...

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Cada individuo será un orden de estos IDs

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

[2, 0, 1, 3, 4, 7, 8, 9, 5, 6, 10, 11]

[9, 5, 6, 1, 3, 4, 7, 8, 10, 11, 2, 0]

Bin Packing Problem

Cada individuo será un orden de estos IDs

[2, 0, 1, 3, 4, 7, 8, 9, 5, 6, 10, 11]

// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

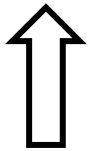
9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

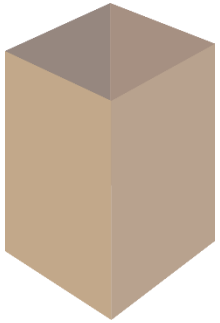
Cada individuo será un orden de estos IDs

[**2**, 0, 1, 3, 4, 7, 8, 9, 5, 6, 10, 11]



Empaquetamos por orden del vector

0 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

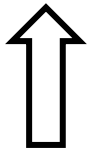
9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

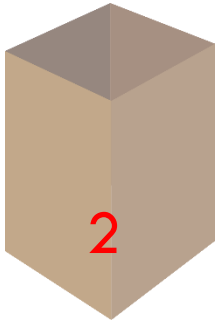
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, 8, 9, 5, 6, 10, 11]



Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

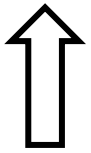
9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

Cada individuo será un orden de estos IDs

[~~2~~, **0**, 1, 3, 4, 7, 8, 9, 5, 6, 10, 11]



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

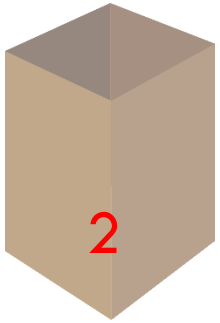
8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Empaquetamos por orden del vector

12 kg



Bin Packing Problem

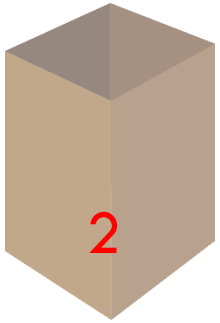
Cada individuo será un orden de estos IDs

[~~2~~, **0**, 1, 3, 4, 7, 8, 9, 5, 6, 10, 11]

Id 0 pesa demasiado

Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

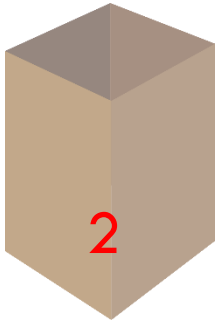
Cada individuo será un orden de estos IDs

[~~2~~, 0, **1**, 3, 4, 7, 8, 9, 5, 6, 10, 11]

Id 1 pesa demasiado

Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

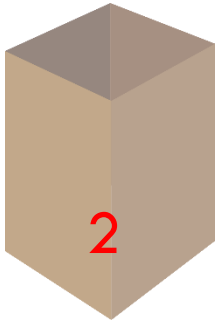
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, **3**, 4, 7, 8, 9, 5, 6, 10, 11]

Id 3 pesa demasiado

Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

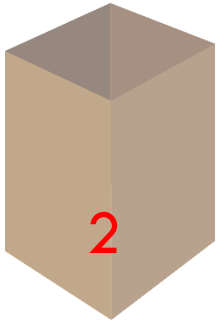
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, **4**, 7, 8, 9, 5, 6, 10, 11]

Id 4 pesa demasiado

Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

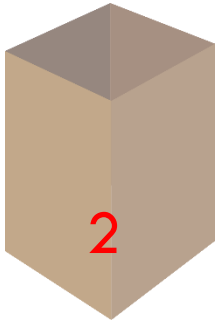
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, **7**, 8, 9, 5, 6, 10, 11]

Id 7 pesa demasiado

Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

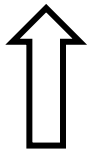
9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

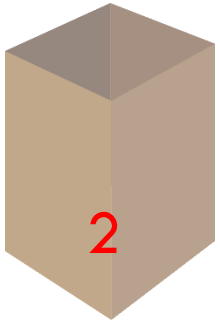
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, **8**, 9, 5, 6, 10, 11]



Empaquetamos por orden del vector

12 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

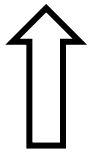
9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

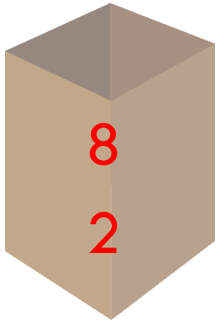
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, 11]



Empaquetamos por orden del vector

15 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

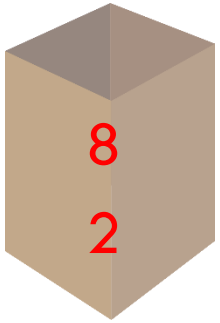
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, ~~8~~, **9**, 5, 6, 10, 11]

Id 9 pesa demasiado

Empaquetamos por orden del vector

15 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

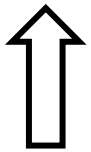
10, 11 -> 1, 2

Bin Packing Problem

Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, **11**]

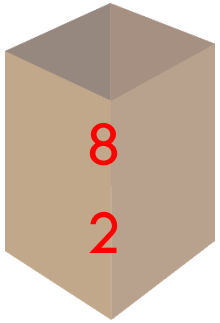
…



Id 11 pesa demasiado

Empaquetamos por orden del vector

15 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

Cada individuo será un orden de estos IDs

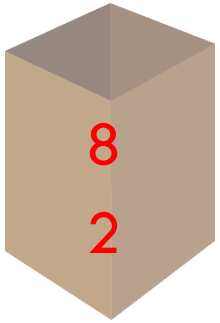
[~~2~~, 0, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, 11]

Terminamos el vector, pero quedan Ids por empaquetar

Abrimos otro contenedor y empezamos desde el primer Id sin empaquetar

Empaquetamos por orden del vector

15 kg



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

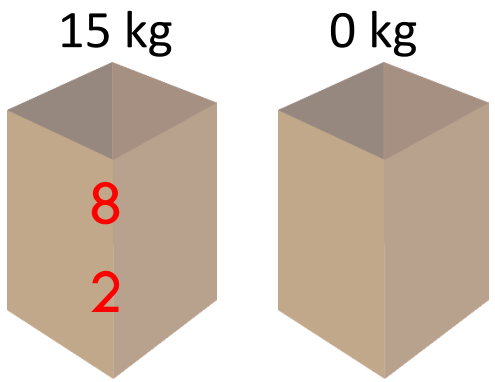
Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, 11]

Terminamos el vector, pero quedan Ids por empaquetar

Abrimos otro contenedor y empezamos desde el primer Id sin empaquetar

Empaquetamos por orden del vector



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem

Cada individuo será un orden de estos IDs

[~~2~~, 0, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, 11]

// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

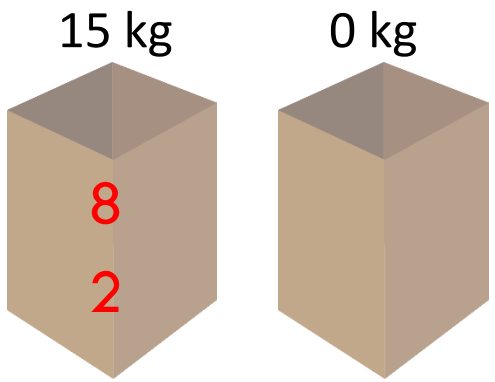
5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Empaquetamos por orden del vector

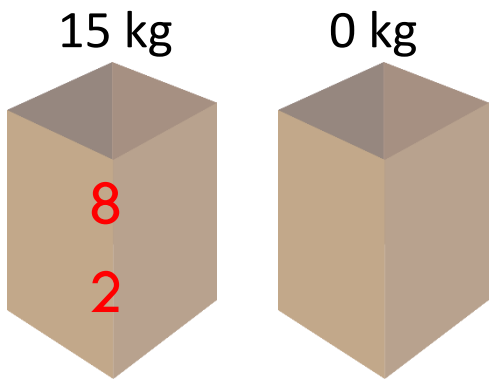


Bin Packing Problem

Cada individuo será un orden de estos IDs

[~~2~~, **0**, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, 11]

Empaquetamos por orden del vector



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

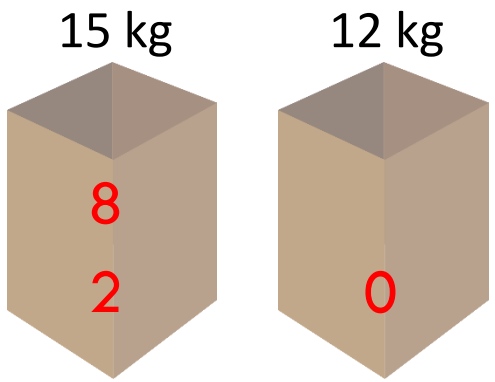
10, 11 -> 1, 2

Bin Packing Problem

Cada individuo será un orden de estos IDs

[~~2~~, ~~0~~, 1, 3, 4, 7, ~~8~~, 9, 5, 6, 10, 11]

Empaquetamos por orden del vector



// Instancia

Datos del problema -> 15, 6

0, 1, 2 -> 12, 3

3, 4 -> 10, 2

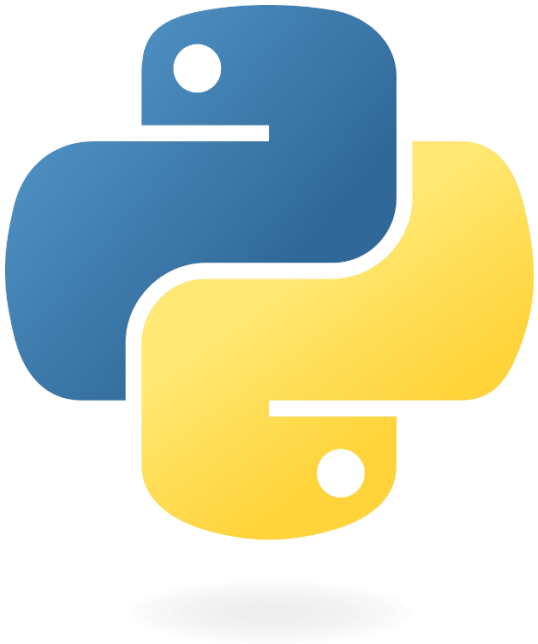
5, 6, 7 -> 4, 3

8 -> 3, 1

9 -> 2, 1

10, 11 -> 1, 2

Bin Packing Problem



Resolución mediante Python usando AG

Bin Packing Problem

BPPConfiguration.py

```
problem = BPPProblem("./bpp.txt")
```

BPPProblem.py

```
class BPPItem

class BPPProblem

class BPPEvaluator
```

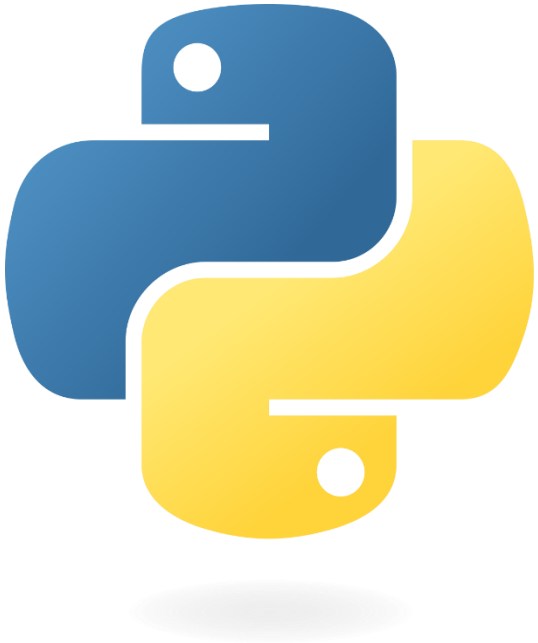
BPPMain.py

```
runGA()

resolver(filename, vector) # Para Debug

Dibujar(problem, bins)
```

Bin Packing Problem



Resolución mediante Python usando AG

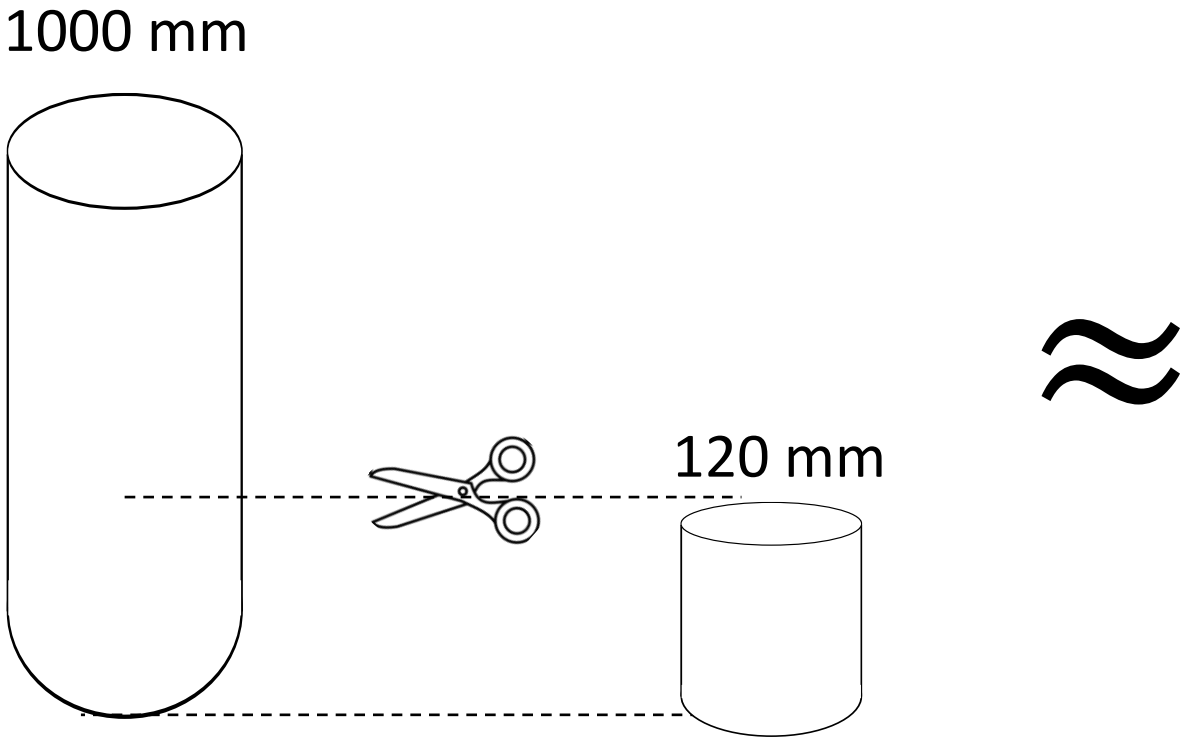
Bin Packing Problem

Pero esto no era de aplicaciones en la vida real??

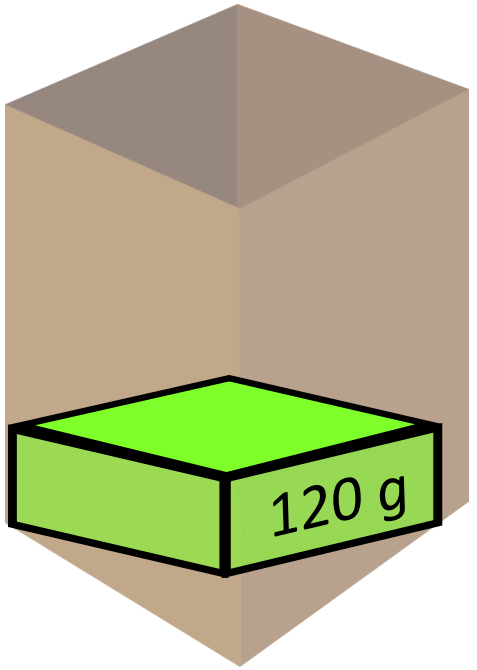
Bin Packing Problem

https://www.youtube.com/watch?v=UN01YCreTZU

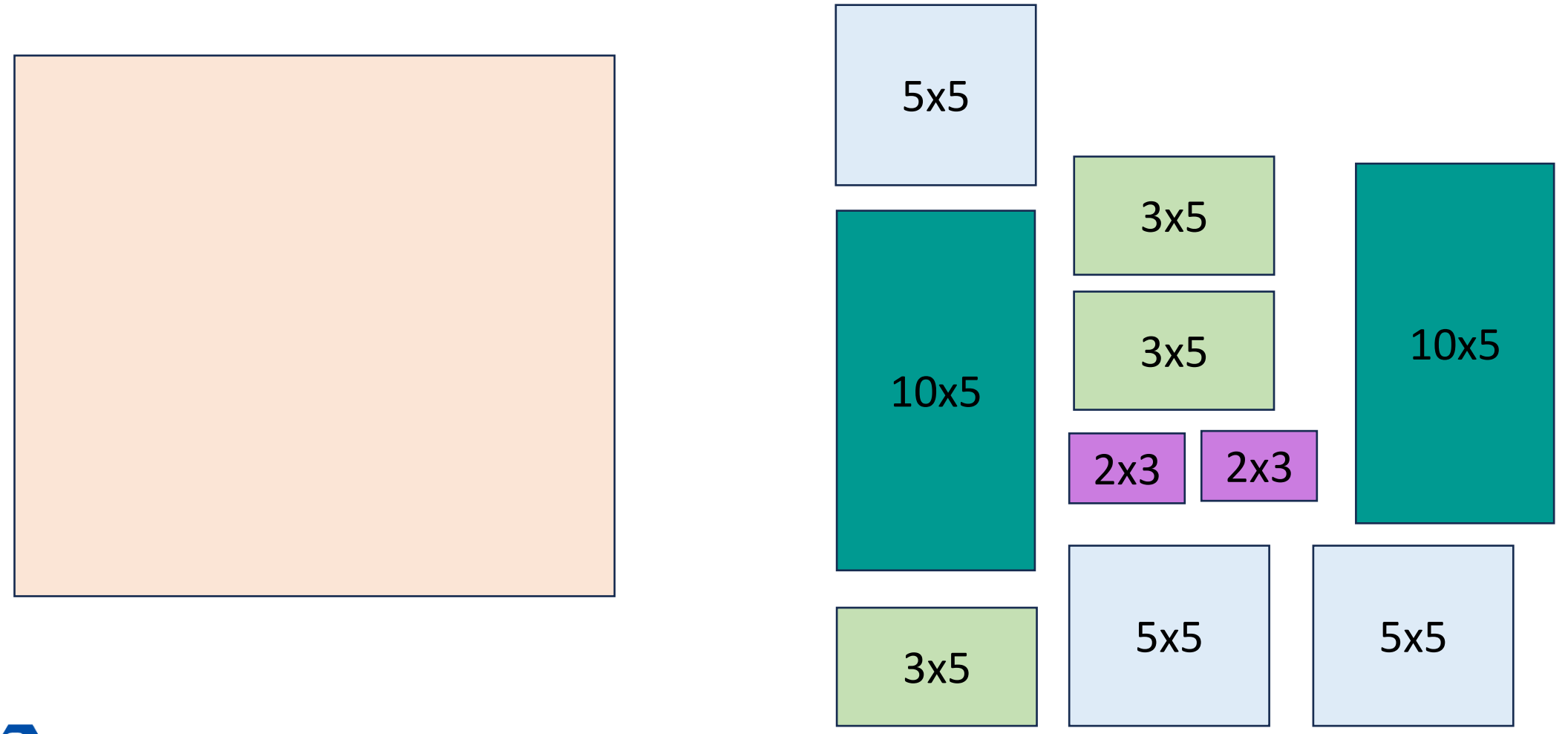
Bin Packing Problem



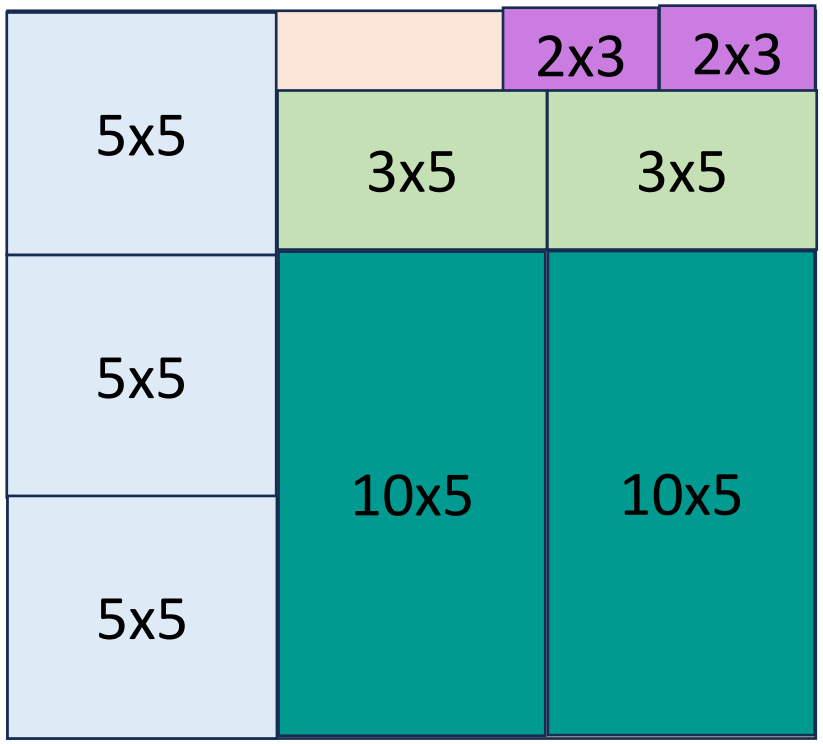
≈

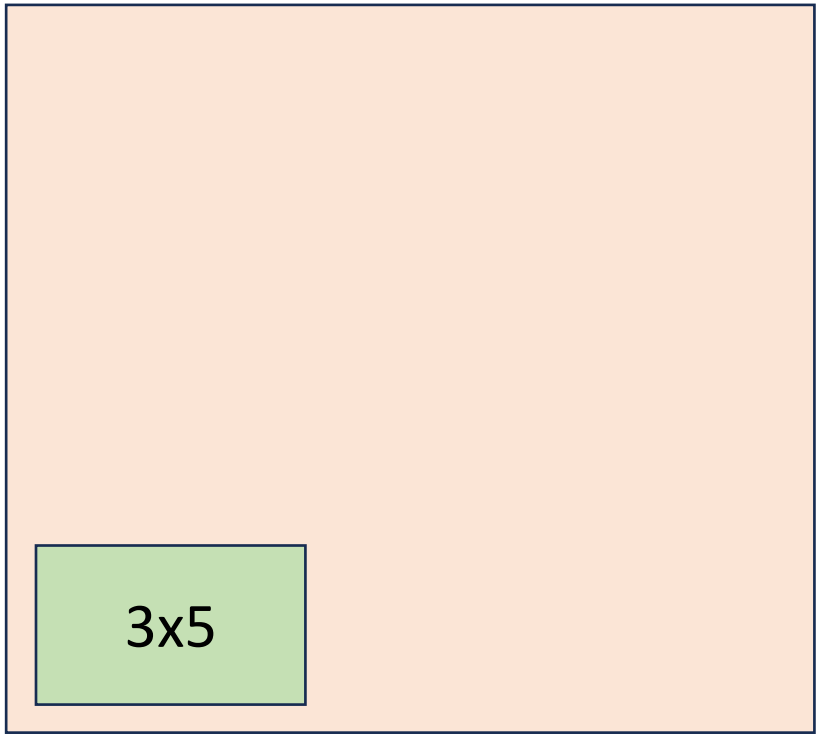


2D Bin Packing Problem



2D Bin Packing Problem





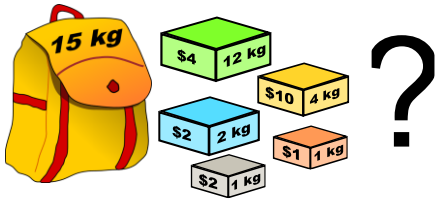
2D Bin Packing Problem

“ShopBot PRS Alpha CNC Router” de Rickwashburn1, disponible en https://commons.wikimedia.org/w/index.php?curid=72933078, bajo licencia Creative Commons Atribución/Compartir-Igual 4.0 (ver licencia en https://creativecommons.org/licenses/by-sa/4.0/deed.en). No se realizaron cambios.

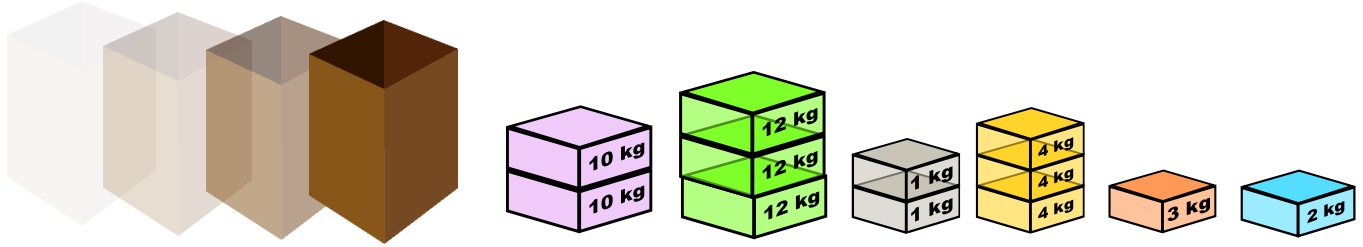
“Corte por plasma” de Diego Baron Carral, disponible en https://commons.wikimedia.org/w/index.php?curid=72933078, bajo licencia Creative Commons CC0 1.0 Universal Public Domain Dedication (ver licencia en https://creativecommons.org/publicdomain/zero/1.0/deed.en). No se realizaron cambios.

Aplicaciones de Scheduling en la vida real

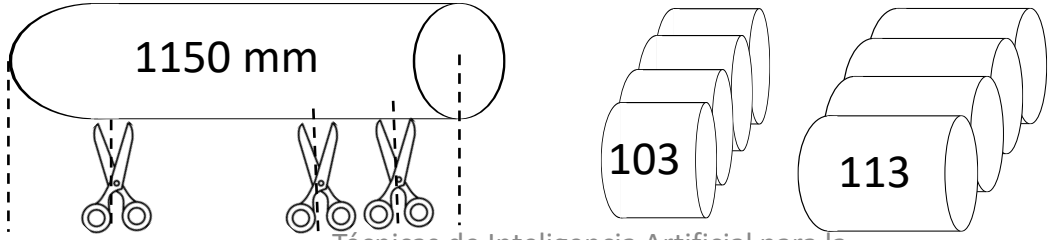
- Problema de la mochila (knapsack problem)



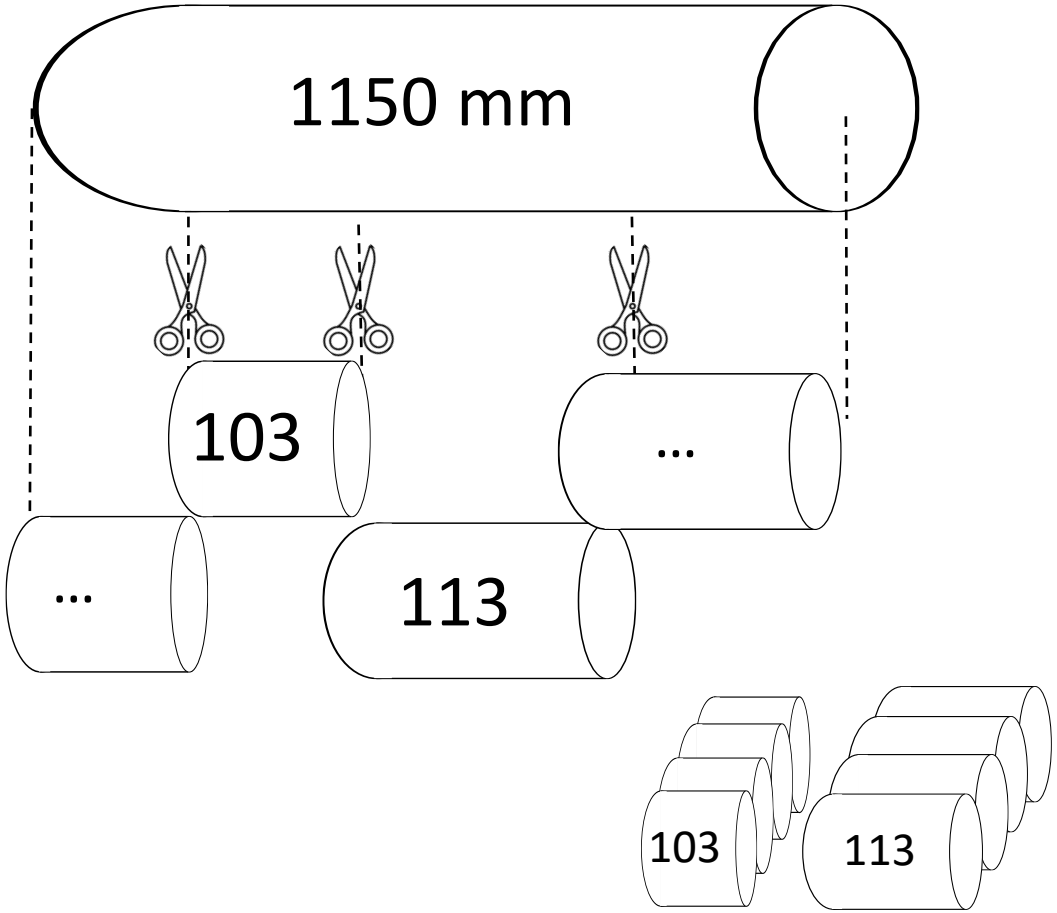
- Problema de empaquetado (Bin Packing Problem)



- Problema de corte de valores (Cutting Stock Problem)

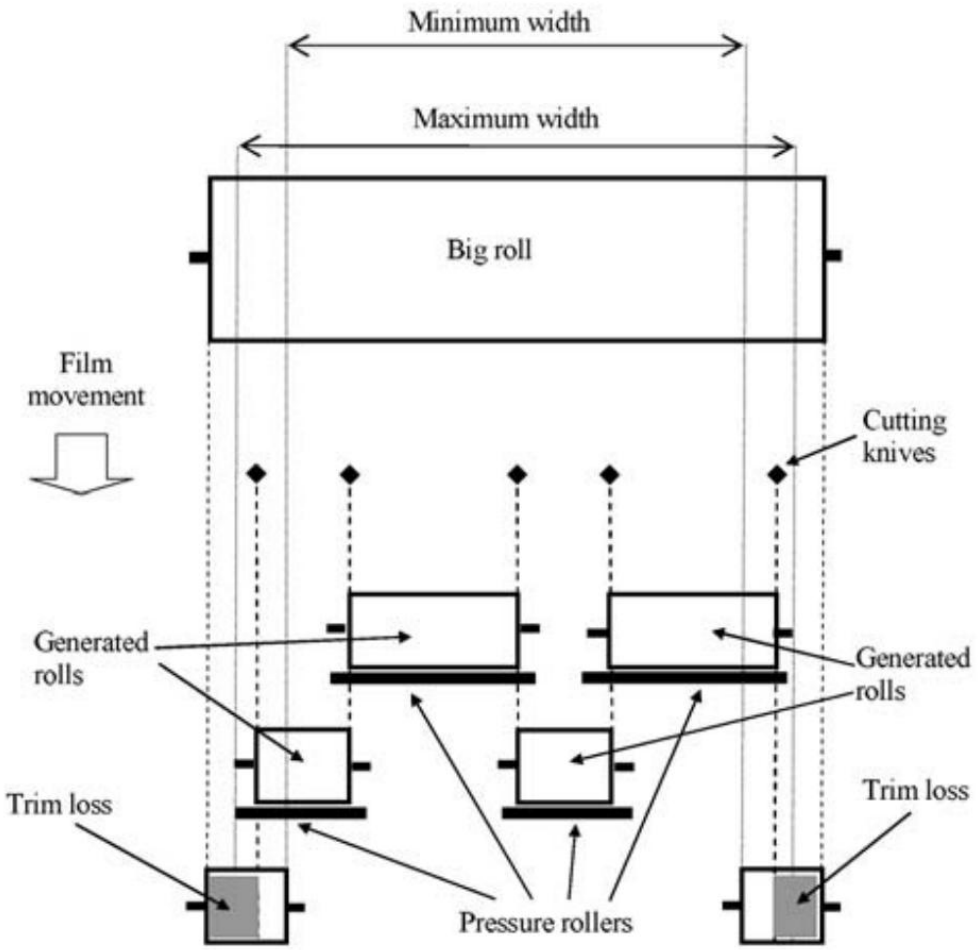


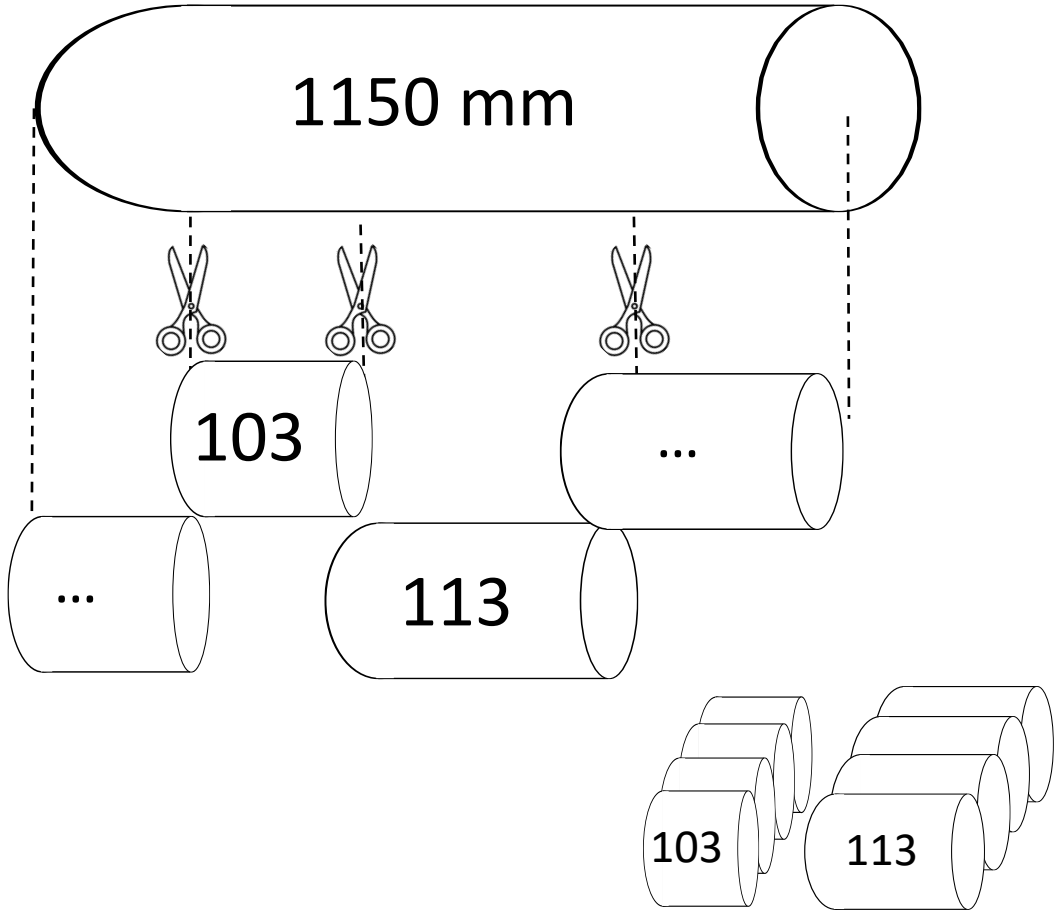
Innovative Film Solutions



innovative
film solutions

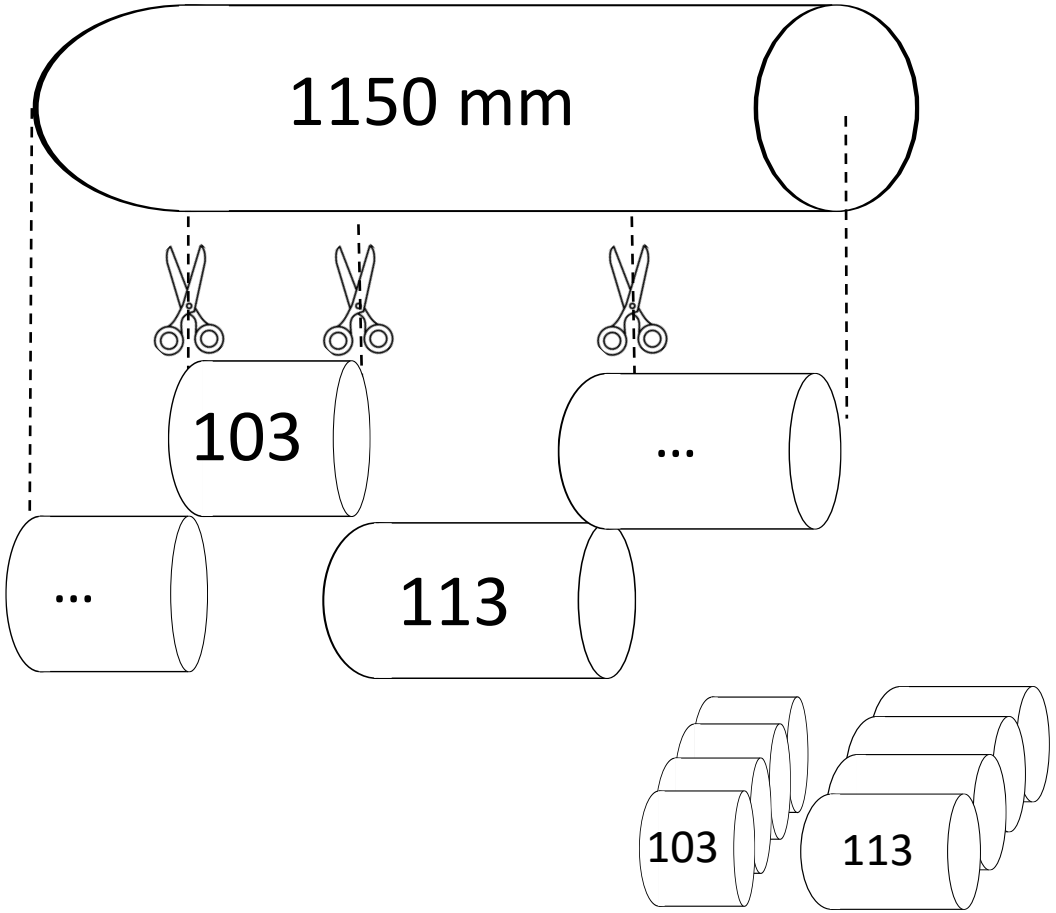
Innovative Film Solutions





Innovative Film Solutions

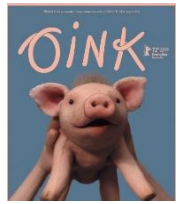
```
// Instancia
1150, 7     // Contenedores de tamaño 1150, 7 anchos
103, 64     // De ancho 103 hay 64 ítems
113, 352    // De ancho 113 hay 352 ítems
118, 112    // ...
125, 112
205 128
210, 64
215, 32
```



Innovative Film Solutions

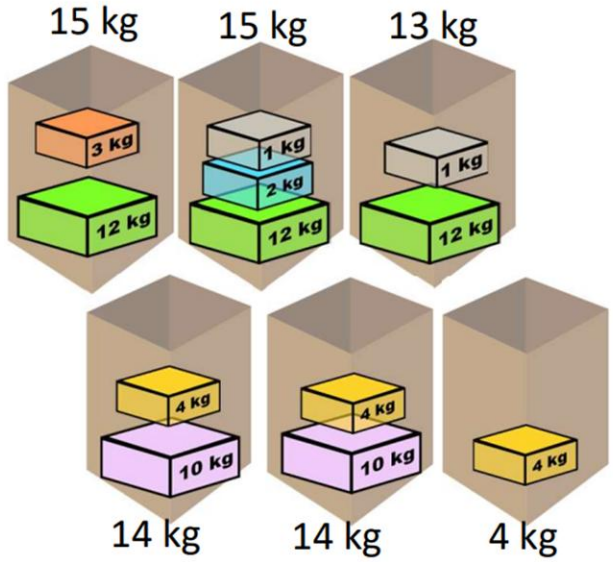
Tu cabecita

Tú



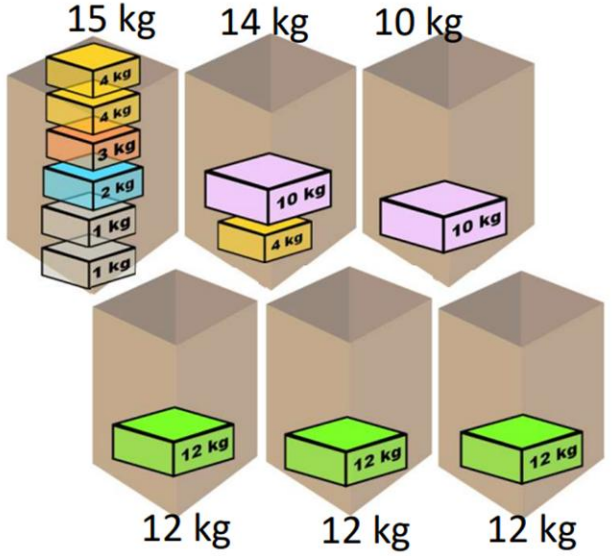
Bin Packing Problem

- Solución #1 N° Contenedores: 6



Empaquetamos el objeto de **mayor peso** que quepa en el contenedor

- Solución #2 N° Contenedores: 6



Empaquetamos el objeto de **menor peso** que quepa en el contenedor

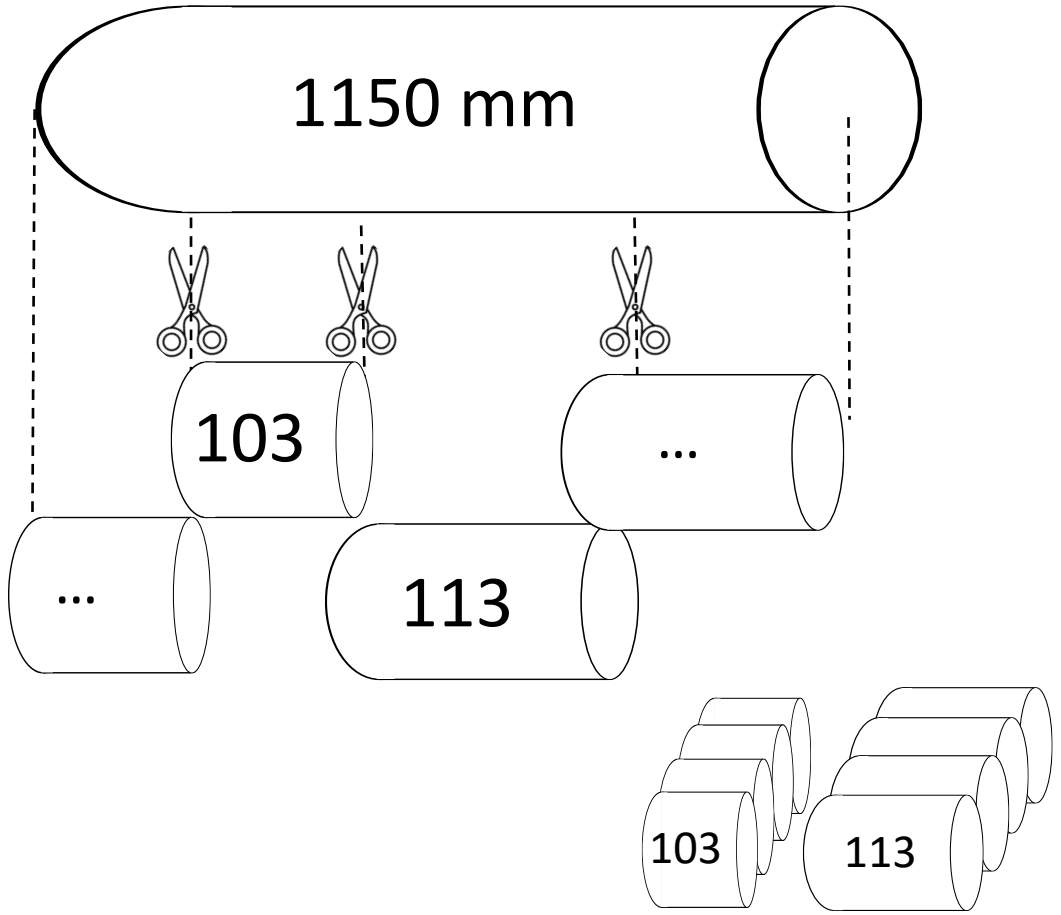
Bin Packing Problem

- ¿Con qué métodos podemos resolver instancias?
- Algoritmo voraz guiado por heurístico
- Algoritmos genéticos
- Programación genética

```
// Instancia
15, 6  // Contenedores de tamaño 15, 6 pesos diferentes
12, 3  // De peso 12 hay 3 ítems
10, 2  // De peso 10 hay 2 ítems
4, 3   // ...
3, 1
2, 1
1, 2
```

Innovative Film Solutions

// Instancia
1150, 7 // Contenedores de tamaño 1150, 7 anchos
103, 64 // De ancho 103 hay 64 ítems
113, 352 // De ancho 113 hay 352 ítems
118, 112 // …
125, 112
205 128
210, 64
215, 32



Programación Genética

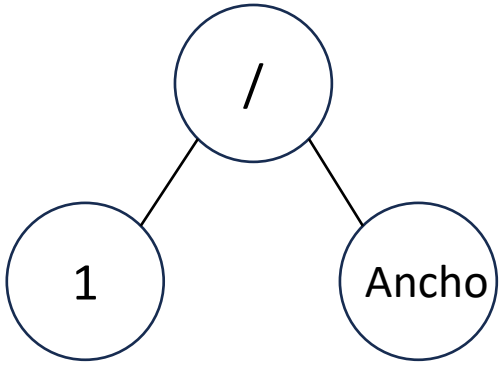
- Algoritmos genéticos
  - Trabajan en el espacio de soluciones
  - Cada individuo codifica una solución

[1, 3, 4, 2, 0, 5, 6, 7, 8]

[8, 3, 0, 4, 6, 5, 2, 7, 1]

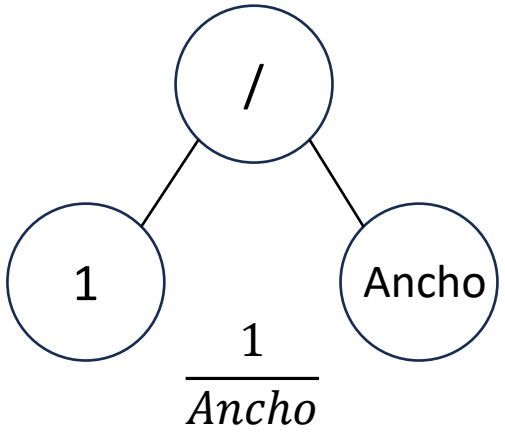
[8, 6, 4, 3, 7, 2, 0, 1, 5]

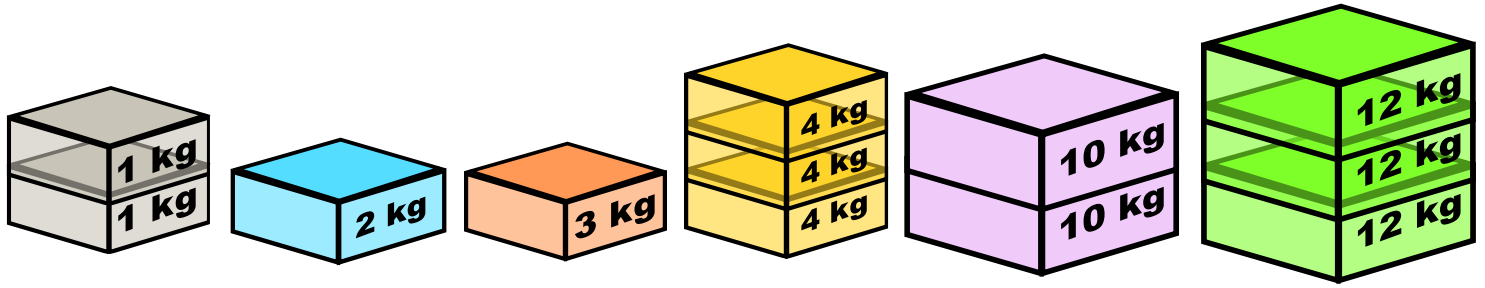
- Programación genética
  - Trabaja en el espacio de heurísticos
  - Cada individuo codifica un heurístico diferente
    - Una heurística es una expresión matemática

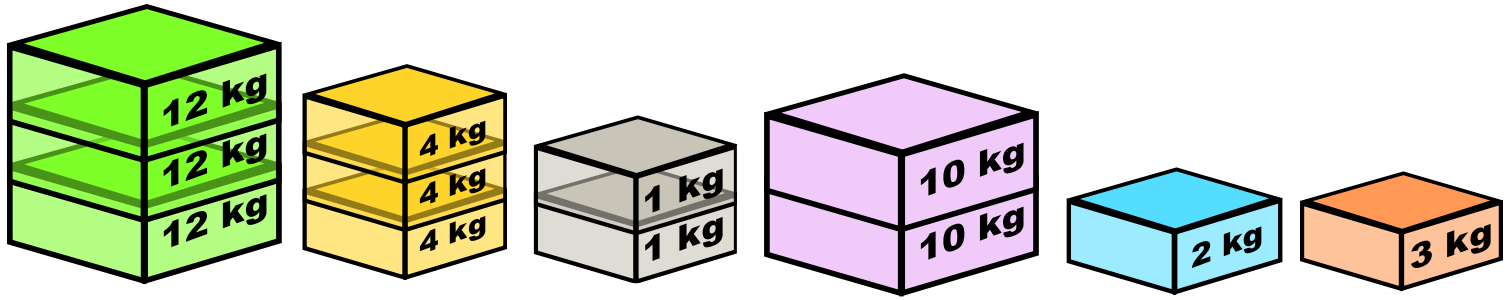


$$\frac{1}{Ancho}$$

Programación Genética

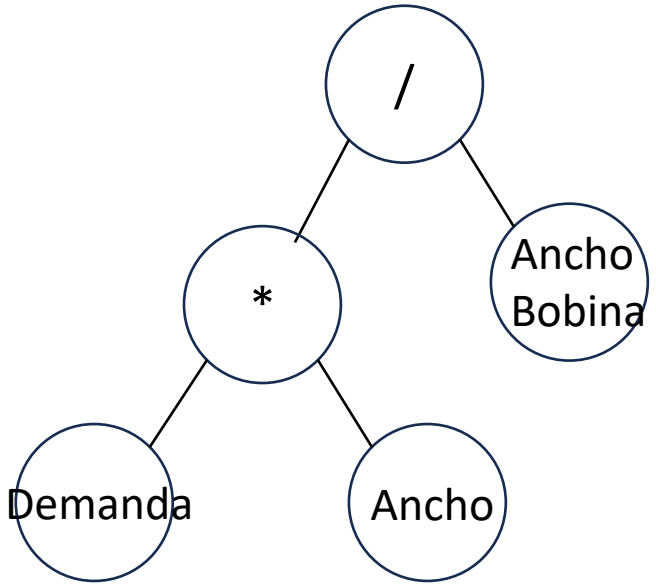






Programación Genética

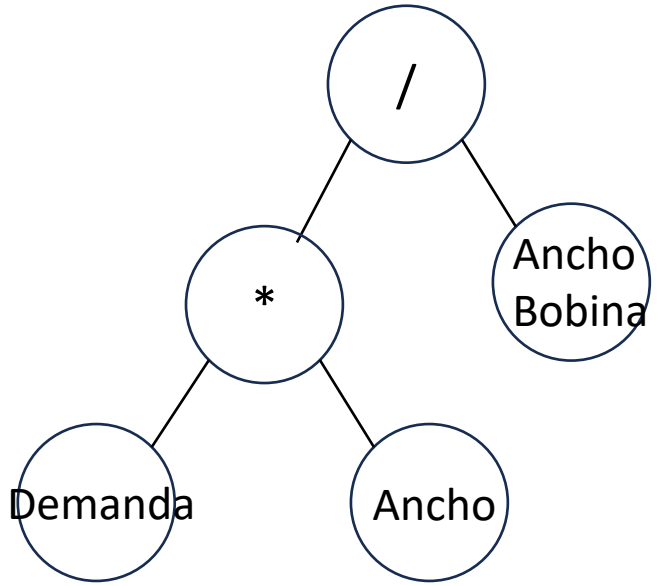
- Programación genética
  - Funciones matemáticas
    - + - * / pow2 exp ln sqrt max min
  - Terminales
    - Ancho Demanda AnchoBobina NúmAnchosEnBobina
    - NúmAnchosPorPlanificar NúmAnchosPlanificados
    - …



$$\frac{Demanda \cdot Ancho}{AnchoBobina}$$

Programación Genética

```
// Instancia
1150, 7    // Contenedores de tamaño 1150, 7 anchos
103, 64    // De ancho 103 hay 64 ítems
113, 352   // De ancho 113 hay 352 ítems
118, 112   // …
125, 112
205 128
210, 64
215, 32
```



$$\frac{Demanda \cdot Ancho}{AnchoBobina}$$

Programación Genética

// Instancia

1150, 7 // Contenedores de tamaño 1150, 7 anchos

103, 64 <- 0, 1, 2, …

113, 352 <- 64, 65, 66, …

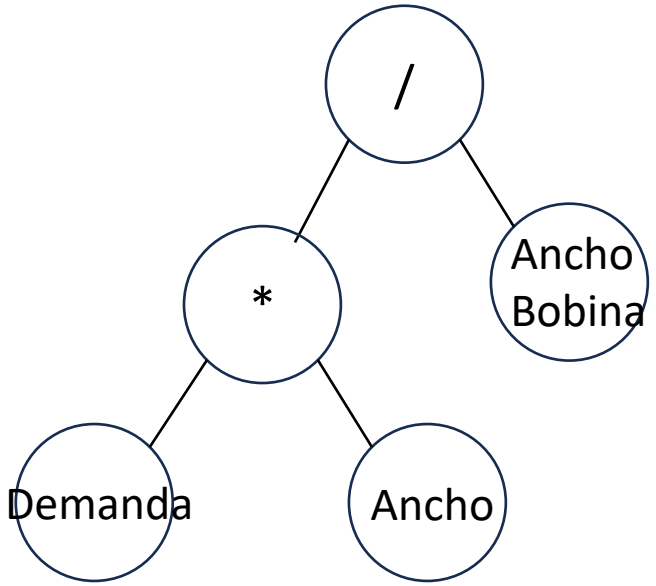
118, 112 <- 416, 417, …

125, 112

205 128

210, 64

215, 32



$$\frac{Demanda \cdot Ancho}{AnchoBobina}$$

Programación Genética

// Instancia

1150, 7 // Contenedores de tamaño 1150, 7 anchos

103, 64 <- 0, 1, 2, …

113, 352 <- 64, 65, 66, …

118, 112 <- 416, 417, …

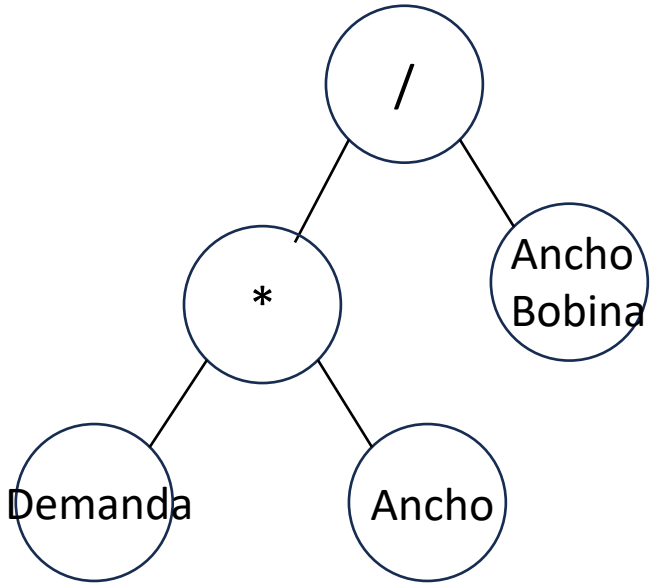
125, 112

205 128

210, 64

215, 32

Id: 65 Prioridad: ?
Ancho: 113
Demanda: 352



$$\frac{Demanda \cdot Ancho}{AnchoBobina}$$

Programación Genética

// Instancia

1150, 7 // Contenedores de tamaño 1150, 7 anchos

103, 64 <- 0, 1, 2, …

113, 352 <- 64, 65, 66, …

118, 112 <- 416, 417, …

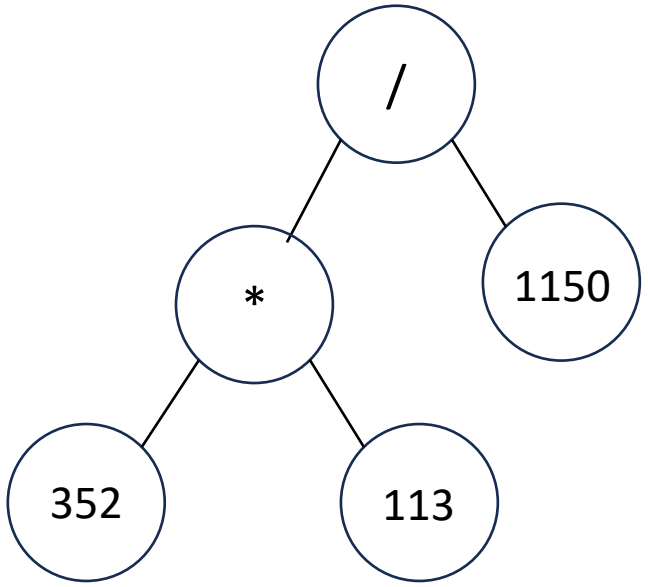
125, 112

205 128

210, 64

215, 32

Id: 65 Prioridad: ?
Ancho: 113
Demanda: 352



$$\frac{352 \cdot 113}{1150}$$

Programación Genética

// Instancia

1150, 7 // Contenedores de tamaño 1150, 7 anchos

103, 64 <- 0, 1, 2, …

113, 352 <- 64, 65, 66, …

118, 112 <- 416, 417, …

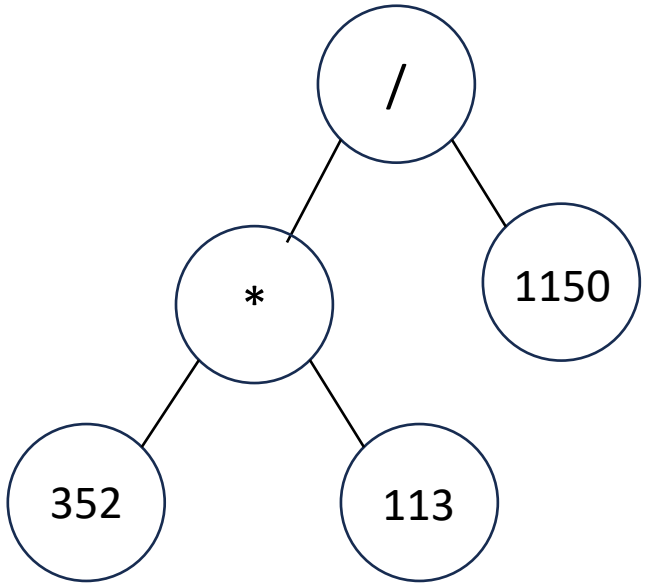
125, 112

205 128

210, 64

215, 32

Id: 65 Prioridad: ?
Ancho: 113
Demanda: 352



$$\frac{352 \cdot 113}{1150} = 34.58782609$$

Programación Genética

// Instancia

1150, 7 // Contenedores de tamaño 1150, 7 anchos

103, 64 <- 0, 1, 2, …

113, 352 <- 64, 65, 66, …

118, 112 <- 416, 417, …

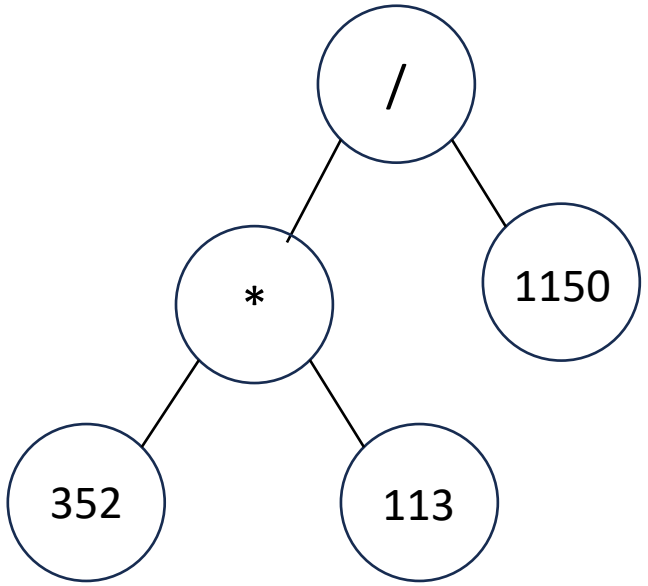
125, 112

205 128

210, 64

215, 32

Id: 65 Prioridad: 34,58
Ancho: 113
Demanda: 352



$$\frac{352 \cdot 113}{1150} = 34.58782609$$

Programación Genética

// Instancia

1150, 7 // Contenedores de tamaño 1150, 7 anchos

103, 64 <- 0, 1, 2, …

113, 352 <- 64, 65, 66, …

118, 112 <- 416, 417, …

125, 112

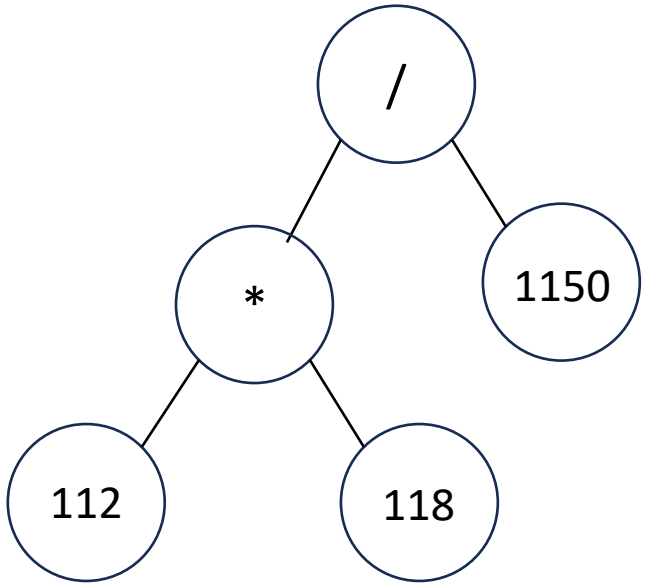
205 128

210, 64

215, 32

Id: 65 Prioridad: 34,58
Ancho: 113
Demanda: 352

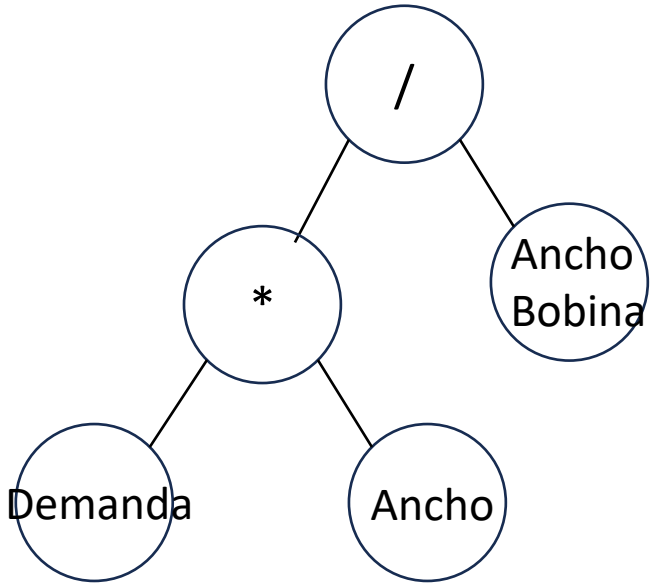
Id: 416 Prioridad: 11,49
Ancho: 118
Demanda: 112



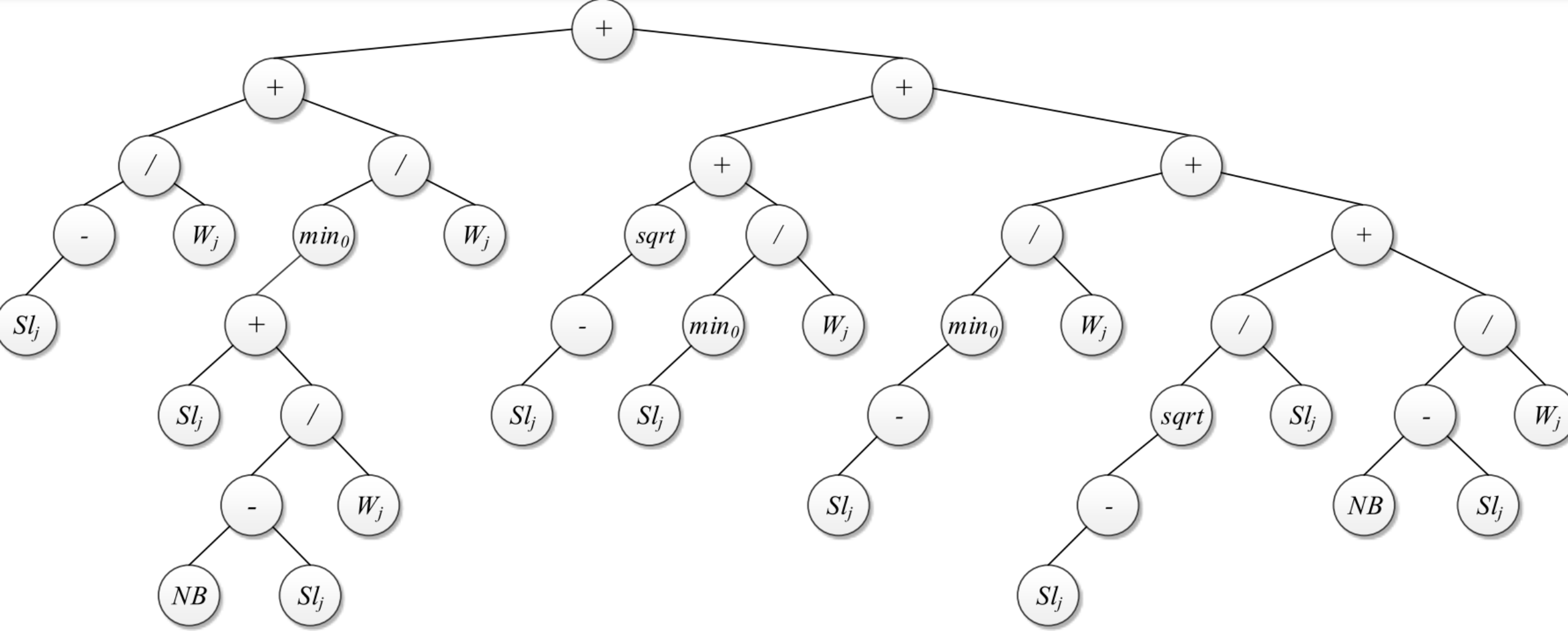
$$\frac{112 \cdot 118}{1150} = 11.4921739$$

Programación Genética

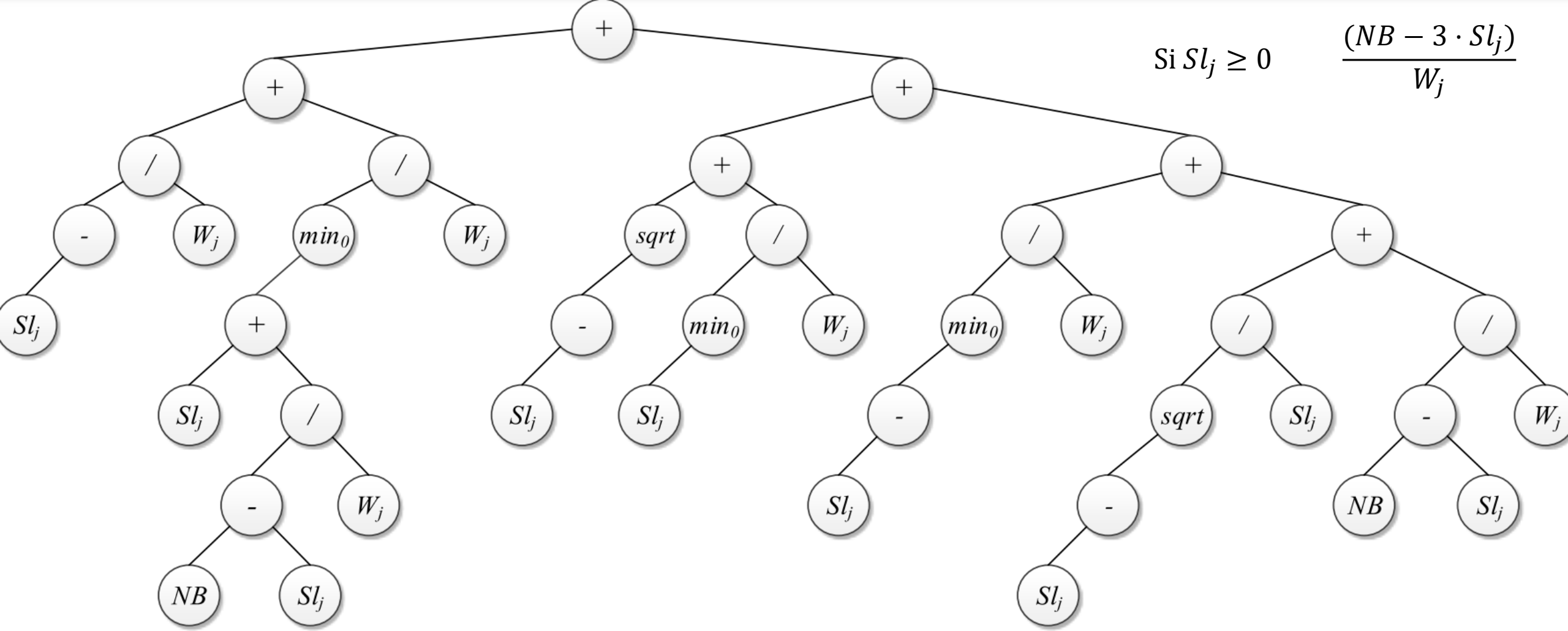
- Programación genética
  - Funciones matemáticas
    - + - * / pow2 exp ln sqrt max min
  - Terminales
    - Ancho Demanda AnchoBobina NúmAnchosEnBobina
    - NúmAnchosPorPlanificar NúmAnchosPlanificados
    - …



$$\frac{Demanda \cdot Ancho}{AnchoBobina}$$



$$\left(\frac{-Sl_j}{W_j}+\left(\frac{min_0\left(Sl_j+\frac{NB-Sl_j}{W_j}\right)}{W_j}\right)\right)+\left(\left(\sqrt{-Sl_j}+\frac{min_0(Sl_j)}{W_j}\right)+\left(\frac{min_0(-Sl_j)}{W_j}+\left(\frac{\sqrt{-Sl_j}}{Sl_j}+\frac{NB-Sl_j}{W_j}\right)\right)\right)$$



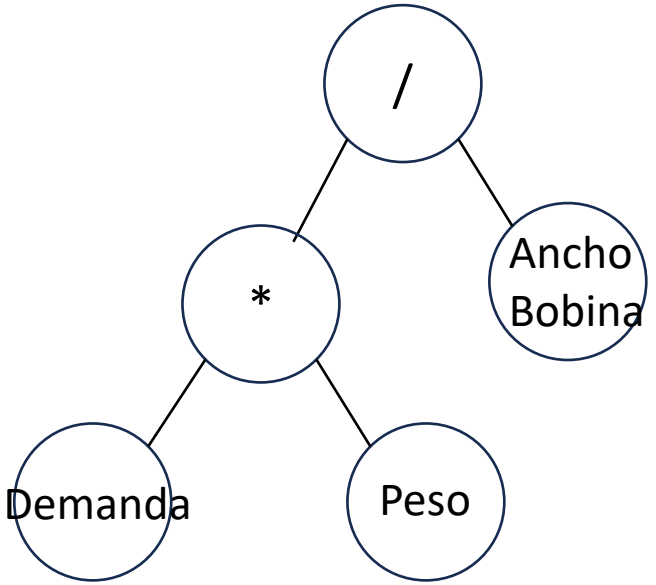
Si $Sl_j \geq 0$ $\quad \frac{(NB - 3 \cdot Sl_j)}{W_j}$

$$\left(\frac{-Sl_j}{W_j} + \left(\frac{min_0\left(Sl_j + \frac{NB - Sl_j}{W_j}\right)}{W_j}\right)\right) + \left(\left(\sqrt{-Sl_j} + \frac{min_0(Sl_j)}{W_j}\right) + \left(\frac{min_0(-Sl_j)}{W_j} + \left(\frac{\sqrt{-Sl_j}}{Sl_j} + \frac{NB - Sl_j}{W_j}\right)\right)\right)$$

Programación Genética

- Programación genética
  - En vez de tener un vector de Ids, tengo un vector de símbolos matemáticos

[/, *, Demanda, Peso, AnchoBobina]



$$\frac{Demanda \cdot Peso}{AnchoBobina}$$

Programación Genética

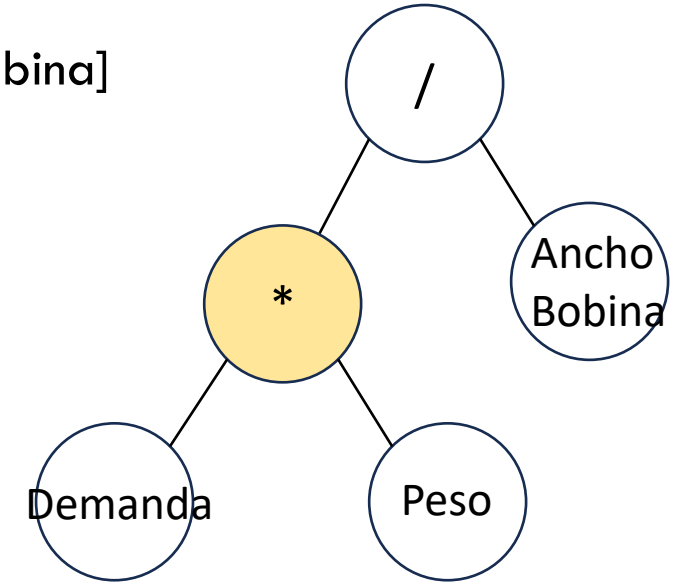
- Programación genética
  - Un cruce entre dos individuos sustituye un subárbol

Padre1 : [/, *, Demanda, Peso, AnchoBobina]

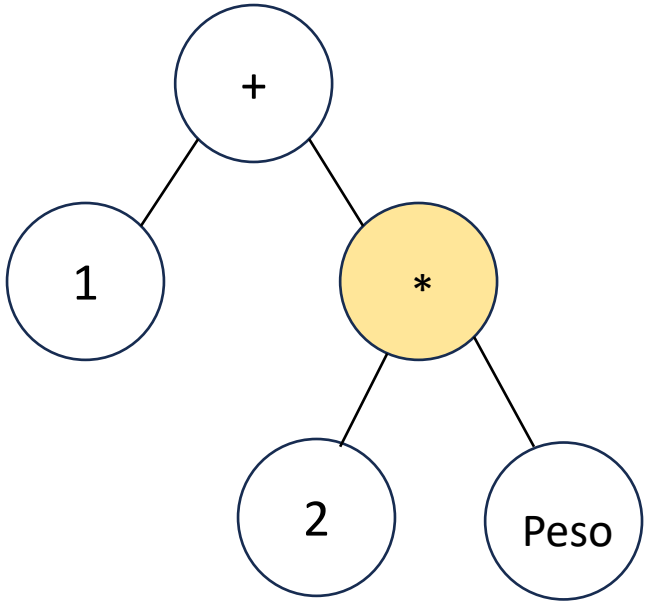
Padre2 : [+, 1, *, 2, Peso]

Hijo1 :

Hijo2 :



$$\frac{Demanda \cdot Peso}{AnchoBobina}$$



$$1 + (2 \cdot Peso)$$

Programación Genética

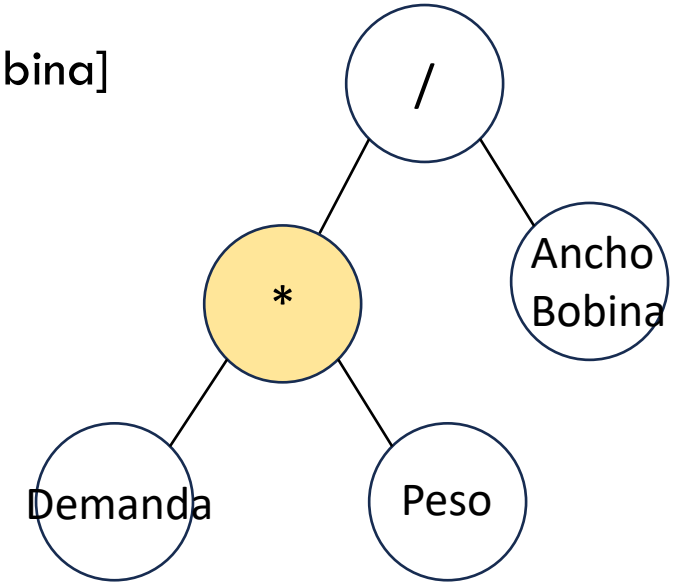
- Programación genética
  - Un cruce entre dos individuos sustituye un subárbol

Padre1 : [/, *, Demanda, Peso, AnchoBobina]

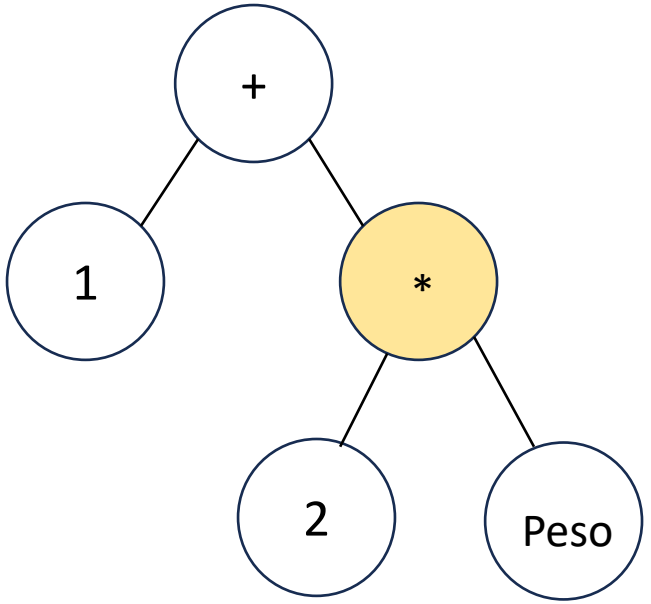
Padre2 : [+, 1, *, 2, Peso]

Hijo1 :

Hijo2 :



$$\frac{Demanda \cdot Peso}{AnchoBobina}$$

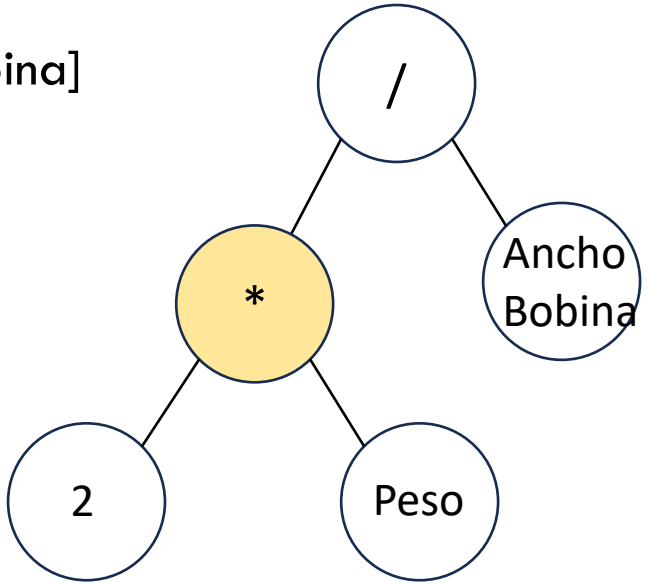


$$1 + (2 \cdot Peso)$$

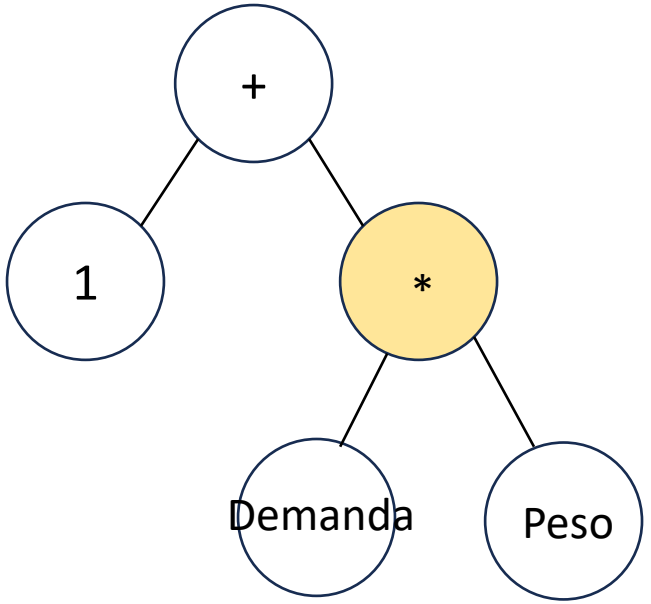
Programación Genética

- Programación genética
  - Un cruce entre dos individuos sustituye un subárbol

Padre1 : [/, *, Demanda, Peso, AnchoBobina]
Padre2 : [+, 1, *, 2, Peso]
Hijo1 : [/, *, 2, Peso, AnchoBobina]
Hijo2 : [+, 1, *, Demanda, Peso]



$$\frac{2 \cdot Peso}{AnchoBobina}$$



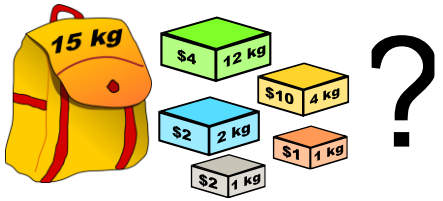
$$1 + (Demanda \cdot Peso)$$

Programación Genética

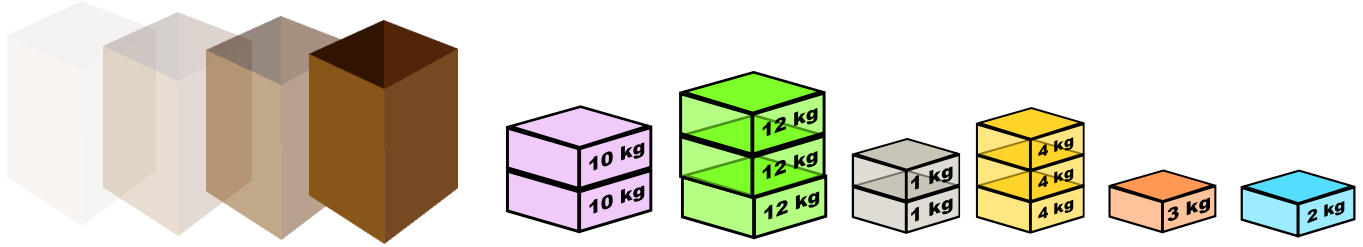
Ejemplo de ejecución con C++ para el CSP

Aplicaciones de Scheduling en la vida real

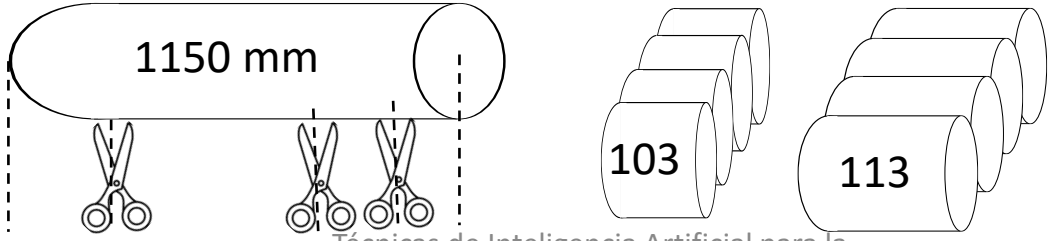
- Problema de la mochila (knapsack problem)



- Problema de empaquetado (Bin Packing Problem)



- Problema de corte de valores (Cutting Stock Problem)



Universidad de Oviedo

Intelligent Scheduling and Optimization

Técnicas de Inteligencia Artificial para la Optimización y Programación de Recursos

Aplicaciones de Scheduling en la vida real I

Jesús Quesada Matilla
{quesadajesus}@uniovi.es

Ciencia de la Computación e Inteligencia Artificial
Departamento de Informática