Two-Dimensional Packing Problems in Telecommunications

Silvano Martello
DEIS, University of Bologna, Italy
silvano.martello@unibo.it

IFORS Distinguished Lecture, CLAIO/SOBRAPO, September 2012, Rio de Janeiro

Two-Dimensional Packing Problems in Telecommunications

Silvano Martello
DEIS, University of Bologna, Italy
silvano.martello@unibo.it

from joint works with A. Lodi & M. Monaci (University of Bologna)

IFORS Distinguished Lecture, CLAIO/SOBRAPO, September 2012, Rio de Janeiro

Two-Dimensional Packing Problems in Telecommunications

Silvano Martello

DEIS, University of Bologna, Italy

silvano.martello@unibo.it

from joint works with A. Lodi & M. Monaci (University of Bologna)

together with

C. Eklund & J. Moilanen (Nokia Siemens Networks)

C. Cicconetti, L. Lenzini & E. Mingozzi (University of Pisa)

IFORS Distinguished Lecture, CLAIO/SOBRAPO, September 2012, Rio de Janeiro

Two-Dimensional Packing Problems in Telecommunications

Silvano Martello
DEIS, University of Bologna, Italy
`silvano.martello@unibo.it`

from joint works with A. Lodi & M. Monaci (University of Bologna)

together with

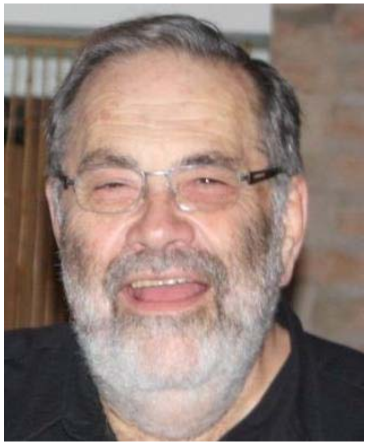
C. Eklund & J. Moilanen (Nokia Siemens Networks)

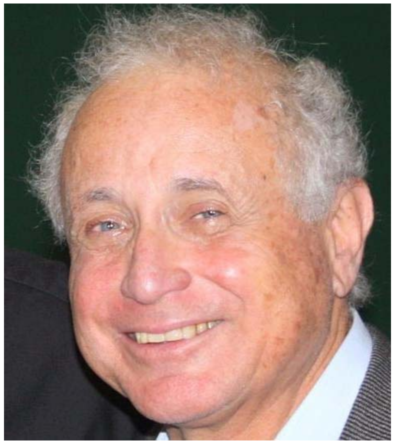
C. Cicconetti, L. Lenzini & E. Mingozzi (University of Pisa)

and

C. Hurkens & G. Woeginger (TU Eindhoven)

IFORS Distinguished Lecture, CLAIO/SOBRAPO, September 2012, Rio de Janeiro






Happy
Birthday!


Happy
Anniversary!

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.
- **Four teams involved:**

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.
- **Four teams involved:** in chronological order,

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.

- **Four teams involved:** in chronological order,

  Nokia Siemens laboratory: research group on the *IEEE 802.16/WiMAX standard*;

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.

- **Four teams involved:** in chronological order,

  Nokia Siemens laboratory: research group on the *IEEE 802.16/WiMAX standard*;

  University of Pisa: research group on *Computer Networking* (**Prof. Luciano Lenzini**);

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.
- **Four teams involved:** in chronological order,

  Nokia Siemens laboratory: research group on the *IEEE 802.16/WiMAX standard*;

  University of Pisa: research group on *Computer Networking* (**Prof. Luciano Lenzini**);

  University of Bologna: research group on *Combinatorial Optimization* (**S.M.**);

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.

- **Four teams involved:** in chronological order,

  Nokia Siemens laboratory: research group on the *IEEE 802.16/WiMAX standard*;

  University of Pisa: research group on *Computer Networking* (**Prof. Luciano Lenzini**);

  University of Bologna: research group on *Combinatorial Optimization* (**S.M.**);

  Technical University of Eindhoven: research group on *Theoretical Combinatorial Optimization* (**Prof. Gerhard J. Woeginger**).

Outline of this talk

- **Objective:** description of the development an interdisciplinary research applicable to real world problems.
- **Four teams involved:** in chronological order,

  Nokia Siemens laboratory: research group on the *IEEE 802.16/WiMAX standard*;

  University of Pisa: research group on *Computer Networking* (**Prof. Luciano Lenzini**);

  University of Bologna: research group on *Combinatorial Optimization* (**S.M.**);

  Technical University of Eindhoven: research group on
  Theoretical Combinatorial Optimization (**Prof. Gerhard J. Woeginger**).

- The whole project has been described in:

  Lodi, Martello, etc ... Efficient two-dimensional packing algorithms for mobile WiMAX.
  Management Science, 2011.

Contents

The project followed the classical steps of an applied OR research:

Contents

The project followed the classical steps of an applied OR research:

1. **birth:** a real-world problem;

Contents

The project followed the classical steps of an applied OR research:

1. birth: a real-world problem;

2. development of mathematical **models** (new two-dimensional packing problems);

Contents

The project followed the classical steps of an applied OR research:

1. **birth:** a real-world problem;

2. development of mathematical **models** (new two-dimensional packing problems);

3. theoretical analysis (**computational complexity**: $\mathcal{NP}$-hard problems);

Contents

The project followed the classical steps of an applied OR research:

1. **birth:** a real-world problem;
2. development of mathematical **models** (new two-dimensional packing problems);
3. theoretical analysis (**computational complexity**: $\mathcal{NP}$-hard problems);
4. definition of mathematical models for **the real-world problems**;

Contents

The project followed the classical steps of an applied OR research:

1. birth: a real-world problem;

2. development of mathematical **models** (new two-dimensional packing problems);

3. theoretical analysis (**computational complexity**: $\mathcal{NP}$-hard problems);

4. definition of mathematical models for **the real-world problems**;

5. evaluation of the **technological constraints** (extremely tough CPU limitations);

Contents

The project followed the classical steps of an applied OR research:

1. **birth:** a real-world problem;

2. development of mathematical **models** (new two-dimensional packing problems);

3. theoretical analysis (**computational complexity**: $\mathcal{NP}$-hard problems);

4. definition of mathematical models for **the real-world problems**;

5. evaluation of the **technological constraints** (extremely tough CPU limitations);

6. development of fast and efficient **algorithms** (derived from the theoretical analysis);

Contents

The project followed the classical steps of an applied OR research:

1. birth: a real-world problem;

2. development of mathematical **models** (new two-dimensional packing problems);

3. theoretical analysis (**computational complexity**: $\mathcal{NP}$-hard problems);

4. definition of mathematical models for **the real-world problems**;

5. evaluation of the **technological constraints** (extremely tough CPU limitations);

6. development of fast and efficient **algorithms** (derived from the theoretical analysis);

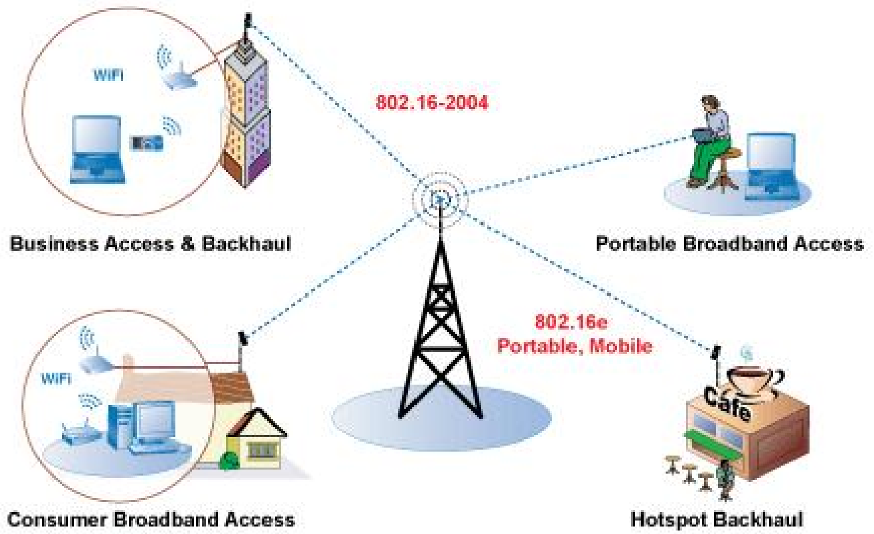
7. implementation and **experimental evaluation** on realistic scenarios.

1. The birth: an optimization problem in telecommunications

Telecommunication systems adopting the **IEEE 802.16/WiMAX** standard:

1. The birth: an optimization problem in telecommunications

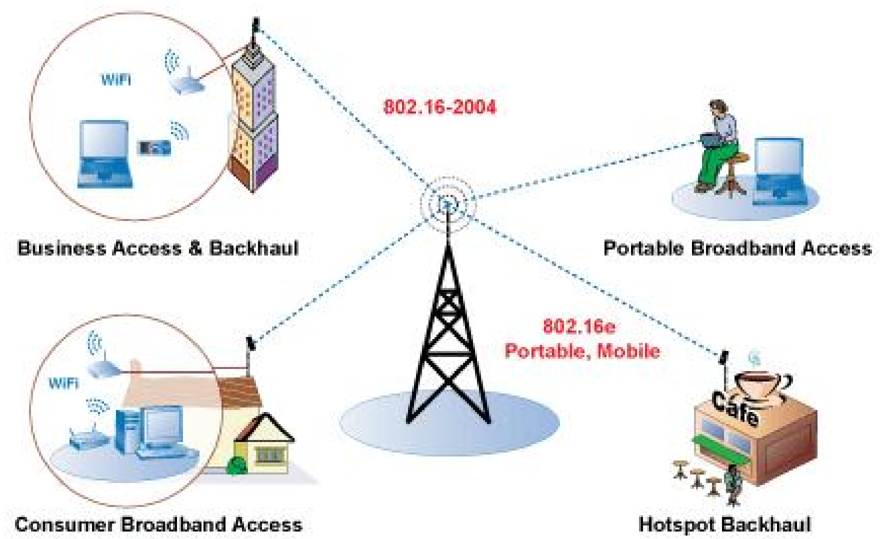
Telecommunication systems adopting the **IEEE 802.16/WiMAX** standard:



- a fixed station transmits and receives data packets
  to and from other stations (e.g., the mobile phones);

1. The birth: an optimization problem in telecommunications

Telecommunication systems adopting the **IEEE 802.16/WiMAX** standard:



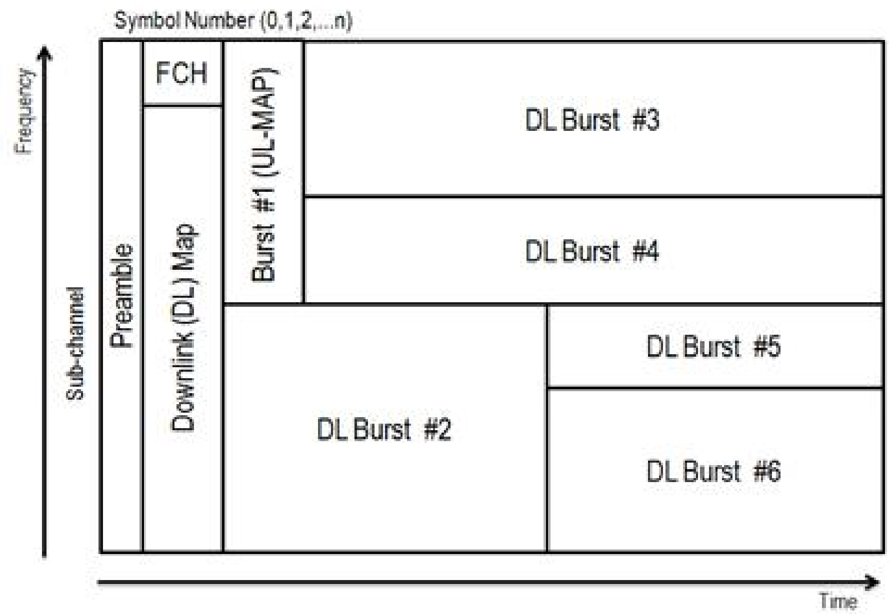
- a fixed station transmits and receives data packets
  to and from other stations (e.g., the mobile phones);
- all transmissions are performed using rectangular frames.

1. The birth: an optimization problem in telecommunications

- In which sense are all transmissions performed using rectangular frames?

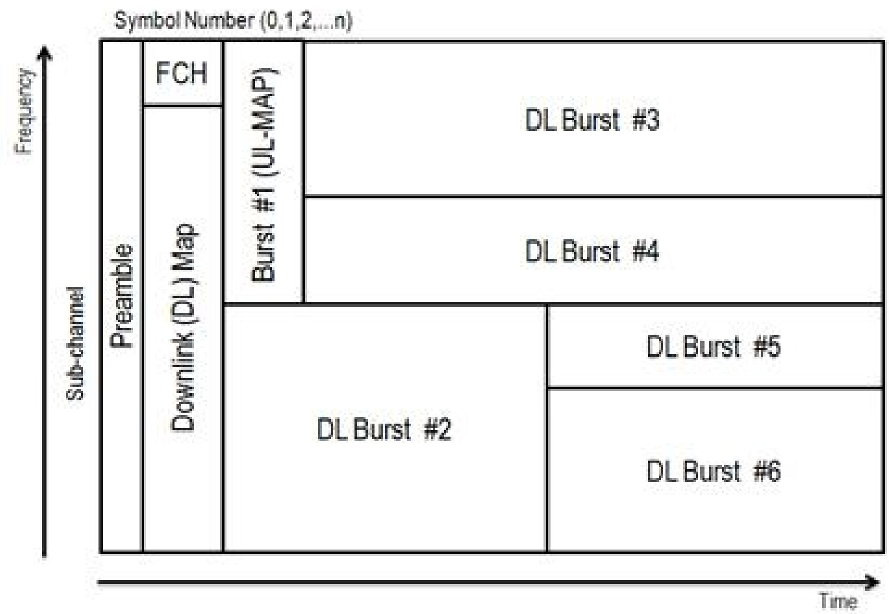
1. The birth: an optimization problem in telecommunications

- In which sense are all transmissions performed using rectangular frames?
- The data to be transmitted are split into packets, which are arranged into [time $\times$ frequency] rectangles, and allocated to the **downlink zones**:



1. The birth: an optimization problem in telecommunications

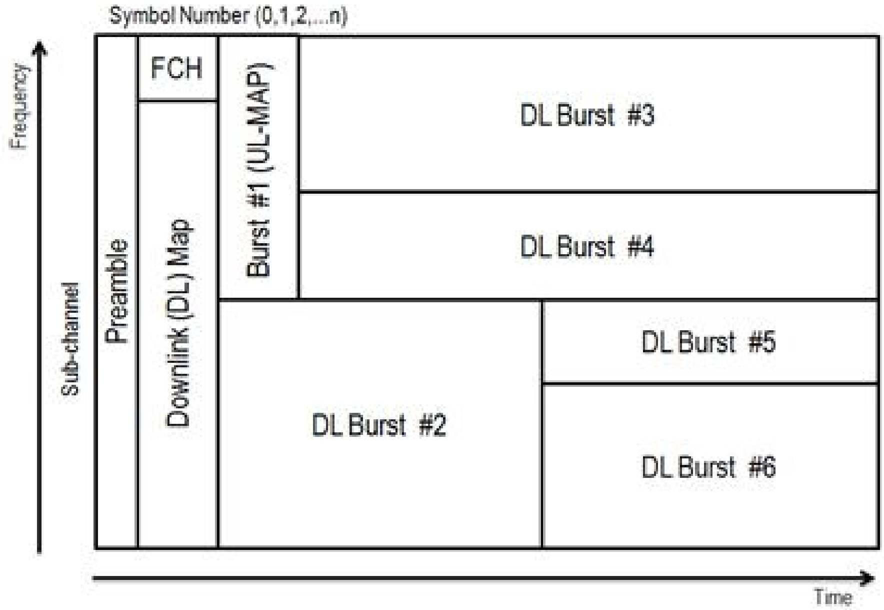
- In which sense are all transmissions performed using rectangular frames?
- The data to be transmitted are split into packets, which are arranged into [time $\times$ frequency] rectangles, and allocated to the **downlink zones**:



- The fixed station must maximize the frame utilization by
  1. deciding which packets will be included in the next transmission phase;

1. The birth: an optimization problem in telecommunications

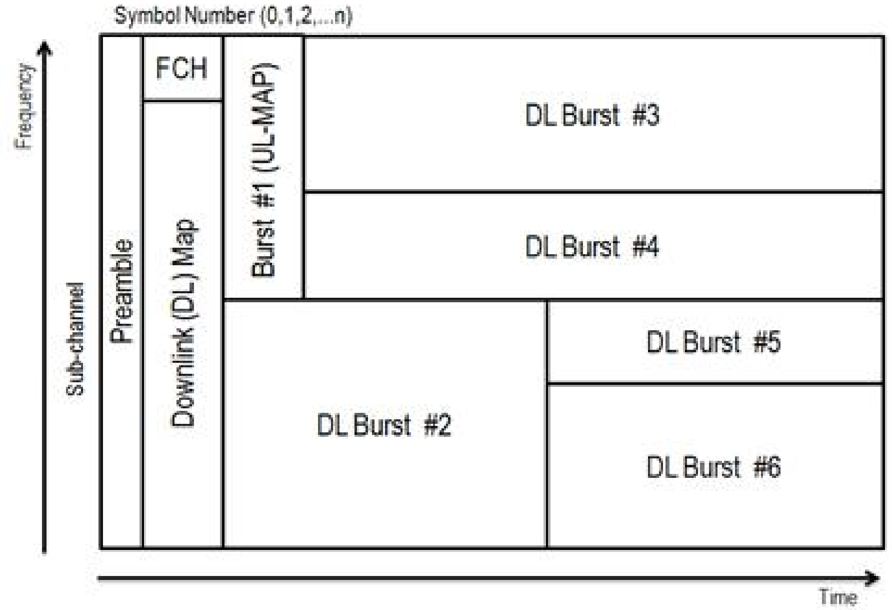
- In which sense are all transmissions performed using rectangular frames?
- The data to be transmitted are split into packets, which are arranged into [time $\times$ frequency] rectangles, and allocated to the **downlink zones**:



- The fixed station must maximize the frame utilization by
  1. deciding which packets will be included in the next transmission phase;
  2. arranging each selected packet into one or more rectangular regions;

1. The birth: an optimization problem in telecommunications

- In which sense are all transmissions performed using rectangular frames?
- The data to be transmitted are split into packets, which are arranged into [time $\times$ frequency] rectangles, and allocated to the **downlink zones**:



- The fixed station must maximize the frame utilization by
  1. deciding which packets will be included in the next transmission phase;
  2. arranging each selected packet into one or more rectangular regions;
  3. allocating the regions to the frame (without overlapping).

2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem:

2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:

2. The models: a look at the combinatorial optimization literature

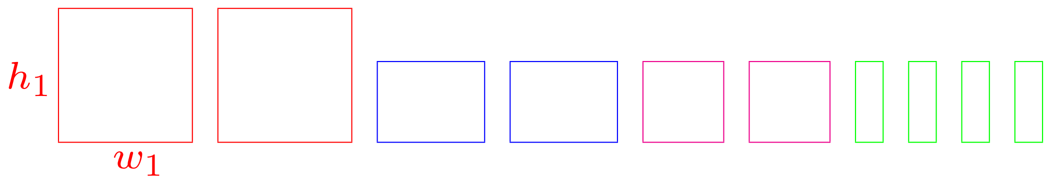
(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W: allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \dots, n)$,

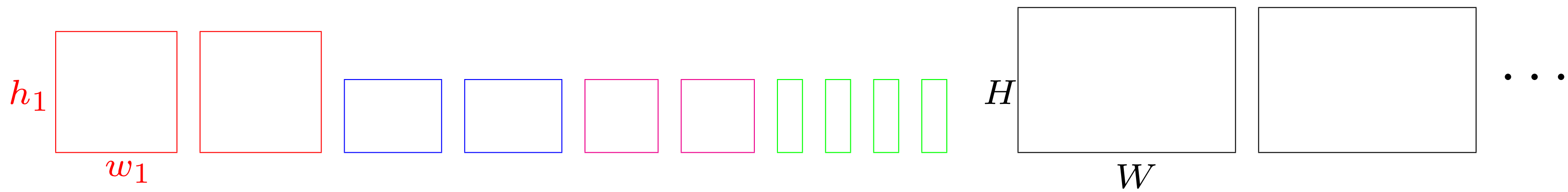


2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



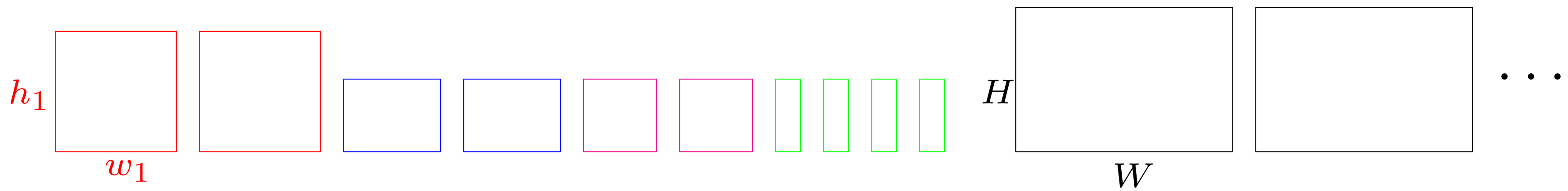
- and an unlimited number of large rectangles (**bins**), having width W and height H,

2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



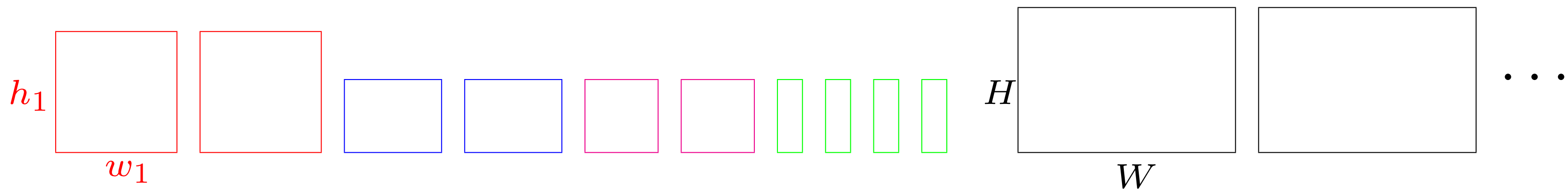
- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:

2. The models: a look at the combinatorial optimization literature

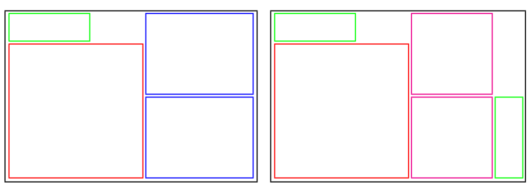
(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W: allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:

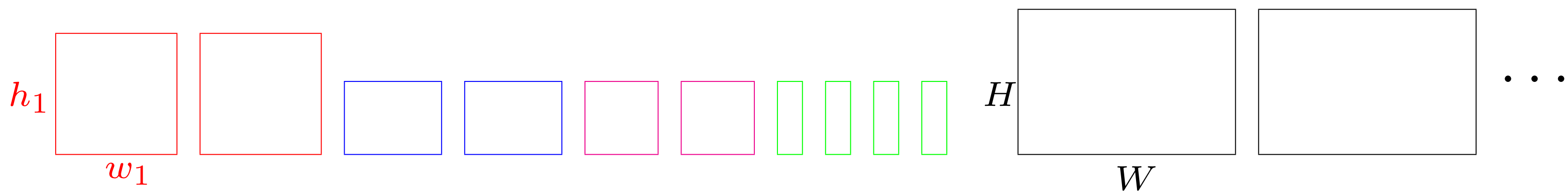


2. The models: a look at the combinatorial optimization literature

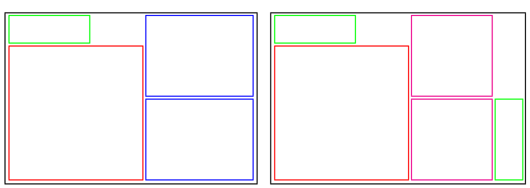
(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:



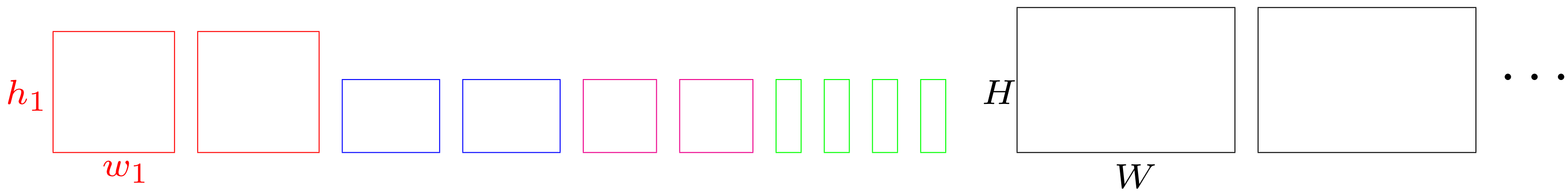
or

2. The models: a look at the combinatorial optimization literature

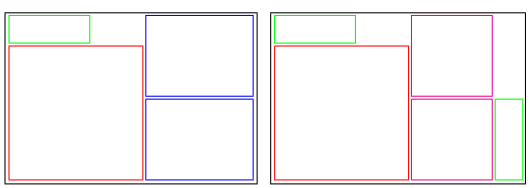
(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:



or

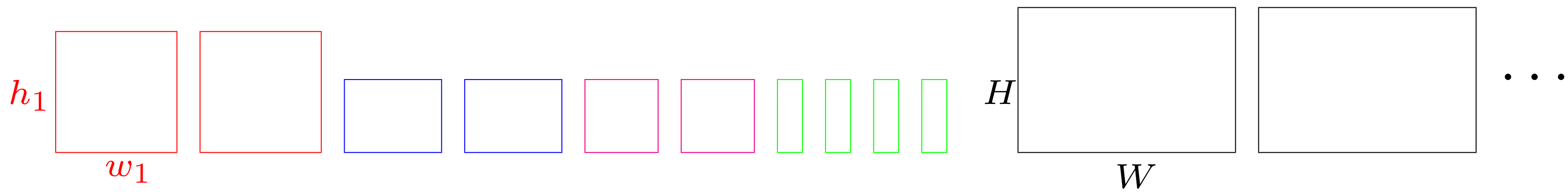
- **B. pack a subset of items**, without overl., in a **single bin maximizing the packed area**.

2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:

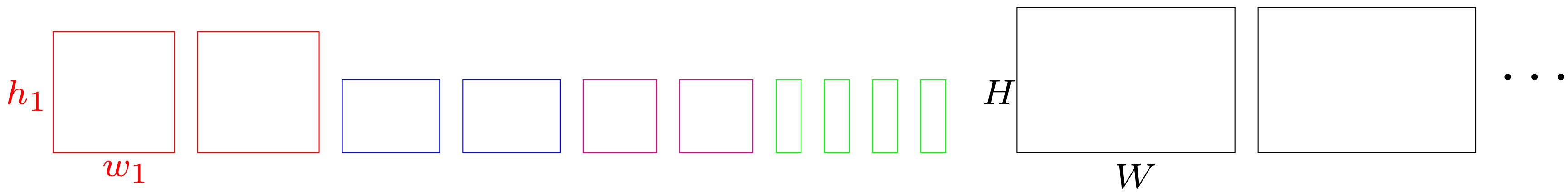
- **B. pack a subset of items**, without overl., in a **single bin maximizing the packed area**.

2. The models: a look at the combinatorial optimization literature

(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W: allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:

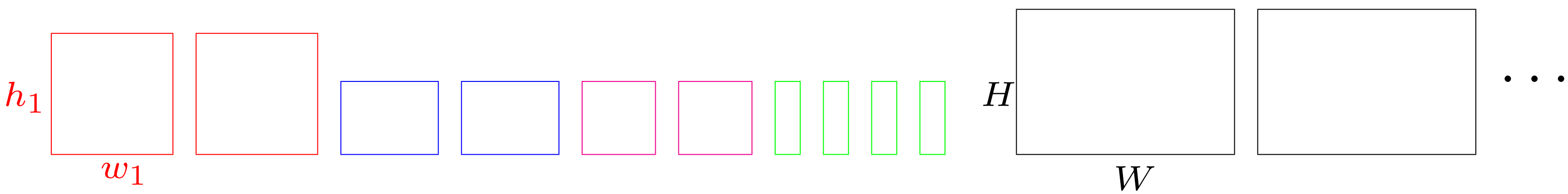
- **B. pack a subset of items**, without overl., in a **single bin maximizing the packed area**.
- **Many variants**: The items may/may not be **rotated**;

2. The models: a look at the combinatorial optimization literature

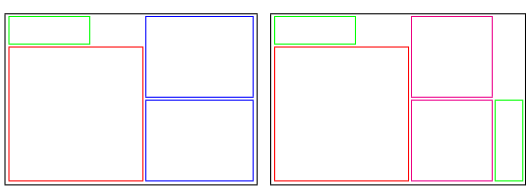
(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

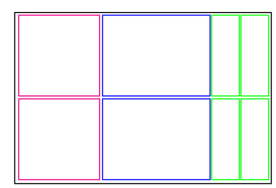
- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:



or



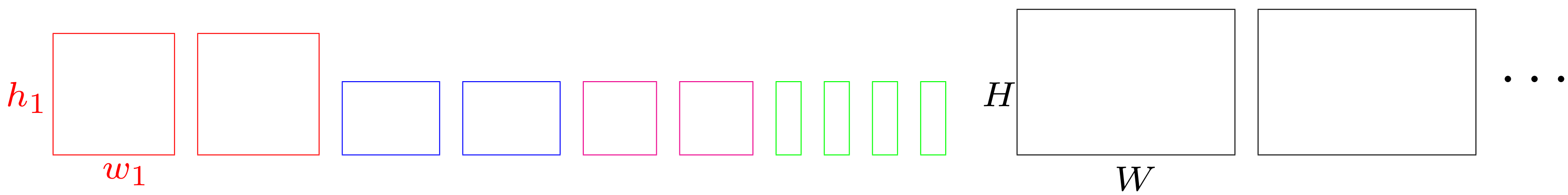
- **B. pack a subset of items**, without overl., in a **single bin maximizing the packed area**.
- **Many variants**: The items may/may not be **rotated**; by **90°/any angle**;

2. The models: a look at the combinatorial optimization literature

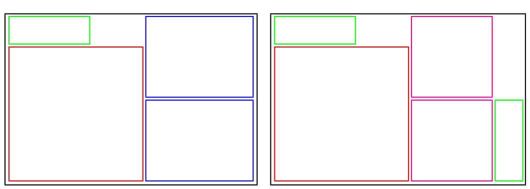
(One-Dimensional) **Bin Packing** problem: n items of size w_j, ∞ bins of size W:
allocate all items to the minimum number of bins.

Standard **Two-Dimensional Bin Packing Problem (2BP)**

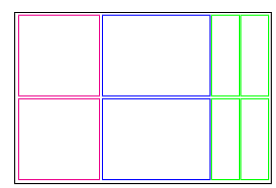
- given n **rectangles** (**items**), having width w_j and height h_j $(j = 1, \ldots, n)$,



- and an unlimited number of large rectangles (**bins**), having width W and height H,
- **A. pack all the items**, without overlapping, into the **minimum number of bins**:



or



- **B. pack a subset of items**, without overl., in a **single bin maximizing the packed area**.
- **Many variants**: The items may/may not be **rotated**; by **90°/any angle**;

  guillotine cutting may/may not be imposed (items must be obtained through a sequence of edge-to-edge cuts parallel to the edges of the bin); . . . **huge literature**

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;
  - this area must be allocated to a $w_j \times h_j$ rectangle such that $w_j h_j \geq a_j$,

2. The models: our problems vs standard 2BPs

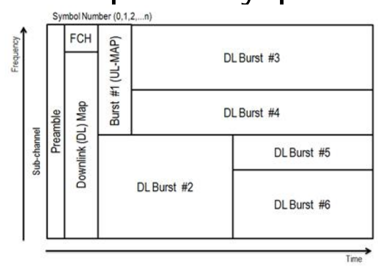
Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;
  - this area must be allocated to a $w_j \times h_j$ rectangle such that $w_j h_j \geq a_j$,
  - **or** to a number m_j of rectangles such that $w_{j_1} h_{j_1} + \ldots + w_{jm_j} h_{jm_j} \geq a_j$;

2. The models: our problems vs standard 2BPs

Main difference

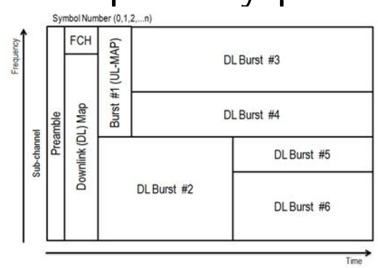
- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;
  - this area must be allocated to a $w_j \times h_j$ rectangle such that $w_j h_j \geq a_j$,
  - **or** to a number m_j of rectangles such that $w_{j_1} h_{j_1} + \ldots + w_{j m_j} h_{j m_j} \geq a_j$;
  - the selected rectangles must then be optimally packed in the **downlink zone** (the **bin**):



2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;
  - this area must be allocated to a $w_j \times h_j$ rectangle such that $w_j h_j \geq a_j$,
  - **or** to a number m_j of rectangles such that $w_{j_1} h_{j_1} + \ldots + w_{j m_j} h_{j m_j} \geq a_j$;
  - the selected rectangles must then be optimally packed in the **downlink zone** (the **bin**):

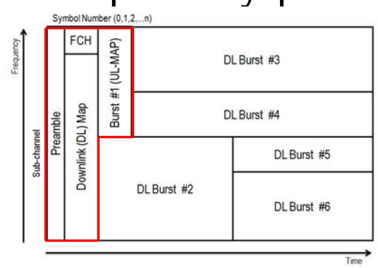


  - each packed rectangle needs information in the downlink zone (sizes, coordinates), i.e.,

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;
  - this area must be allocated to a $w_j \times h_j$ rectangle such that $w_j h_j \geq a_j$,
  - **or** to a number m_j of rectangles such that $w_{j_1} h_{j_1} + \ldots + w_{jm_j} h_{jm_j} \geq a_j$;
  - the selected rectangles must then be optimally packed in the **downlink zone** (the **bin**):

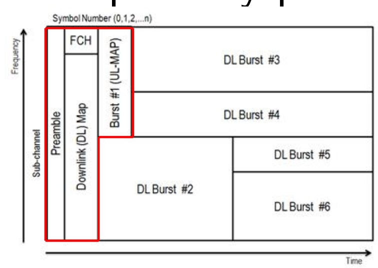


  - each packed rectangle needs information in the downlink zone (sizes, coordinates), i.e.,
  - part of the bin is used for **maps transmission**: size proportional to number of rectangles;

2. The models: our problems vs standard 2BPs

Main difference

- **Input to 2BP**: set of rectangles to be packed.
- **Input to the telecommunication problems**: set of data packets to be packed:
  - a **data packet** is an amount of information, in practice a **number**;
  - this number may be interpreted as an **area** a_j;
  - this area must be allocated to a $w_j \times h_j$ rectangle such that $w_j h_j \geq a_j$,
  - **or** to a number m_j of rectangles such that $w_{j_1} h_{j_1} + \ldots + w_{jm_j} h_{jm_j} \geq a_j$;
  - the selected rectangles must then be optimally packed in the **downlink zone** (the **bin**):



  - each packed rectangle needs information in the downlink zone (sizes, coordinates), i.e.,
  - part of the bin is used for **maps transmission**: size proportional to number of rectangles;
  - hence the need of limiting the number of rectangles.

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?
- Can they be solved efficiently in practice?

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?
- Can they be solved efficiently in practice?

To answer these questions, let us consider the simplest combinatorial optimization problem we can "extract" from the given problems:

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?
- Can they be solved efficiently in practice?

To answer these questions, let us consider the simplest combinatorial optimization problem we can "extract" from the given problems:

Problem P0 (Area Packing):

- n areas;

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?
- Can they be solved efficiently in practice?

To answer these questions, let us consider the simplest combinatorial optimization problem we can "extract" from the given problems:

Problem P0 (Area Packing):

- n areas;
- a single bin;

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?
- Can they be solved efficiently in practice?

To answer these questions, let us consider the simplest combinatorial optimization problem we can "extract" from the given problems:

Problem P0 (Area Packing):

- n areas;
- a single bin;
- allocate each area to one rectangle, and

3. Theoretical analysis: Problem P0

Questions:

- How difficult are the telecommunication problems at hand?
- Can they be solved in polynomial time? If not,
- Can they be solved in pseudo-polynomial time? If not
- Can they be approximated with worst-case performance guarantee in polynomial time?
- Can they be solved efficiently in practice?

To answer these questions, let us consider the simplest combinatorial optimization problem we can "extract" from the given problems:

Problem P0 (Area Packing):

- n areas;
- a single bin;
- allocate each area to one rectangle, and

  pack all the rectangles into the bin without overlapping.

3. Theoretical analysis: recognition version of P0

- We are given:
  - n integer areas $a_j,\ j \in J = \{1, \ldots, n\}$ and

3. Theoretical analysis: recognition version of P0

- We are given:
  - n integer areas $a_j, \ j \in J = \{1, \ldots, n\}$ and
  - a single bin of integer sizes $W \times H$, with $W \cdot H \geq \sum_{j \in J} a_j$

3. Theoretical analysis: recognition version of P0

- We are given:
  - n integer areas $a_j, \; j \in J = \{1, \ldots, n\}$ and
  - a single bin of integer sizes $W \times H$, with $W \cdot H \geq \sum_{j \in J} a_j$
- Is it possible to find integers $w_1, \ldots, w_n$ and $h_1, \ldots, h_n$ such that:

3. Theoretical analysis: recognition version of P0

- We are given:
  - n integer areas $a_j,\ j \in J = \{1, \dots, n\}$ and
  - a single bin of integer sizes $W \times H$, with $W \cdot H \geq \sum_{j \in J} a_j$
- Is it possible to find integers $w_1, \dots, w_n$ and $h_1, \dots, h_n$ such that:
  - $a_j = w_j h_j,\ j \in J$, and

3. Theoretical analysis: recognition version of P0

- We are given:
  - n integer areas $a_j,\ j \in J = \{1, \dots, n\}$ and
  - a single bin of integer sizes $W \times H$, with $W \cdot H \geq \sum_{j \in J} a_j$
- Is it possible to find integers $w_1, \dots, w_n$ and $h_1, \dots, h_n$ such that:
  - $a_j = w_j h_j,\ j \in J$, and
  - the n rectangles $R_j = [w_j, h_j],\ j \in J$, can be packed into the bin without overlapping?

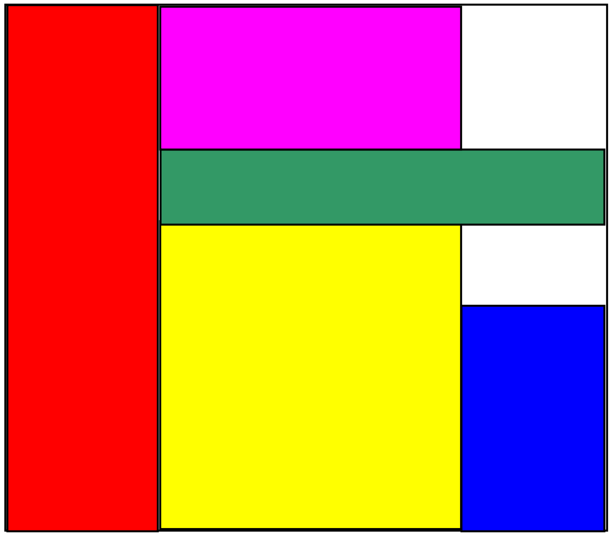
3. Theoretical analysis: recognition version of P0

- We are given:
  - n integer areas $a_j,\ j \in J = \{1, \ldots, n\}$ and
  - a single bin of integer sizes $W \times H$, with $W \cdot H \geq \sum_{j \in J} a_j$
- Is it possible to find integers $w_1, \ldots, w_n$ and $h_1, \ldots, h_n$ such that:
  - $a_j = w_j h_j,\ j \in J$, and
  - the n rectangles $R_j = [w_j, h_j],\ j \in J$, can be packed into the bin without overlapping?

- 8 x 7

- 14
- 6
- 8
- 6
- 16

- 14 = 2 x 7
- 6 = 2 x 3
- 8 = 4 x 2
- 6 = 6 x 1
- 16 = 4 x 4



3. Theoretical analysis: Complexity of P0

3. Theoretical analysis: Complexity of P0

- A simple (although non-trivial) transformation from a variant of PARTITION shows that **P0 is ordinary NP-complete**.

3. Theoretical analysis: Complexity of P0

- A simple (although non-trivial) transformation from a variant of PARTITION shows that **P0 is ordinary NP-complete**.
- Sophisticated techniques using
  - tools from number theory,

3. Theoretical analysis: Complexity of P0

- A simple (although non-trivial) transformation from a variant of PARTITION shows that **P0 is ordinary NP-complete**.
- Sophisticated techniques using
  - tools from number theory,
  - transformation from a variant of THREE-PARTITION

  prove that **P0 is strongly NP-complete**.

3. Theoretical analysis: Complexity of P0

- A simple (although non-trivial) transformation from a variant of PARTITION shows that **P0 is ordinary NP-complete**.

- Sophisticated techniques using
  - tools from number theory,
  - transformation from a variant of THREE-PARTITION

  prove that **P0 is strongly NP-complete**.

  Hurkens, Lodi, Martello, Monaci and Woeginger
  Complexity and approximation of an area packing problem
  Optimization Letters, 2012.

3. Theoretical analysis: Complexity of P0

- A simple (although non-trivial) transformation from a variant of PARTITION shows that **P0 is ordinary NP-complete**.
- Sophisticated techniques using
  - tools from number theory,
  - transformation from a variant of THREE-PARTITION

  prove that **P0 is strongly NP-complete**.

  Hurkens, Lodi, Martello, Monaci and Woeginger
  Complexity and approximation of an area packing problem
  Optimization Letters, 2012.

- Hence P0 **cannot be solved** in polynomial time, nor in pseudo-polynomial time unless $\mathcal{P} = \mathcal{NP}$.

3. Theoretical analysis: Complexity of P0

- A simple (although non-trivial) transformation from a variant of PARTITION shows that **P0 is ordinary NP-complete**.

- Sophisticated techniques using
  - tools from number theory,
  - transformation from a variant of THREE-PARTITION

  prove that **P0 is strongly NP-complete**.

  Hurkens, Lodi, Martello, Monaci and Woeginger
  Complexity and approximation of an area packing problem
  Optimization Letters, 2012.

- Hence P0 **cannot be solved** in polynomial time, nor in pseudo-polynomial time unless $\mathcal{P} = \mathcal{NP}$.

- However its optimization version can be approximated with worst-case performance guarantee in polynomial time.

3. Theoretical analysis: Optimization version of P0

- The recognition version of problem P0 can be transformed into the following **optimization version** (closer to the real-world problem):

3. Theoretical analysis: Optimization version of P0

- The recognition version of problem P0 can be transformed into the following **optimization version** (closer to the real-world problem):
  - assume that any area a_j can be arbitrarily **split** into any number of integer rectangular sub-areas (at most a_j 1×1 (unit) squares);

3. Theoretical analysis: Optimization version of P0

- The recognition version of problem P0 can be transformed into the following **optimization version** (closer to the real-world problem):
  - assume that any area a_j can be arbitrarily **split** into any number of integer rectangular sub-areas (at most a_j 1×1 (unit) squares);
  - objective: pack all areas into the bin without overlapping

    by minimizing the number of created rectangular sub-areas.

3. Theoretical analysis: Optimization version of P0

- The recognition version of problem P0 can be transformed into the following **optimization version** (closer to the real-world problem):
  - assume that any area a_j can be arbitrarily **split** into any number of integer rectangular sub-areas (at most a_j 1×1 (unit) squares);
  - objective: pack all areas into the bin without overlapping by minimizing the number of created rectangular sub-areas.

- Of course, if the optimal solution to the optimization version has value n, i.e., a unique rectangular sub-area is created for each original area, then the recognition version has answer "YES".

3. Theoretical analysis: Optimization version of P0

- The recognition version of problem P0 can be transformed into the following **optimization version** (closer to the real-world problem):
  - assume that any area a_j can be arbitrarily **split** into any number of integer rectangular sub-areas (at most a_j 1×1 (unit) squares);
  - objective: pack all areas into the bin without overlapping by minimizing the number of created rectangular sub-areas.
- Of course, if the optimal solution to the optimization version has value n, i.e., a unique rectangular sub-area is created for each original area, then the recognition version has answer "YES".
- This version makes sense by itself as a very naïve approximation of the application at hand. In other words, the best configuration is obtained by minimizing the number of sub-areas.

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

  A. Split each **area** $a_j, \ j \in J$, into **two parts**:

  A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

$$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

  A. Split each **area** $a_j, \ j \in J$, into **two parts**:

  A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

  $$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

  A.2 a one dimensional (vertical) **strip**, i.e., a rectangle of size $1 \times \widetilde{h}_j$ with

  $$\widetilde{h}_j = a_j - \widetilde{w}_j H$$

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

  A. Split each **area** $a_j,\ j \in J$, into **two parts**:

  A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

  $$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

  A.2 a one dimensional (vertical) **strip**, i.e., a rectangle of size $1 \times \widetilde{h}_j$ with

  $$\widetilde{h}_j = a_j - \widetilde{w}_j H$$

  (possibly only one part is created)

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

  A. Split each **area** $a_j,\ j \in J$, into **two parts**:

  A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

  $$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

  A.2 a one dimensional (vertical) **strip**, i.e., a rectangle of size $1 \times \widetilde{h}_j$ with

  $$\widetilde{h}_j = a_j - \widetilde{w}_j H$$

  (possibly only one part is created)

  B. Subdivide the **bin** into **two parts**:

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

A. Split each **area** a_j, $j \in J$, into **two parts**:

A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

$$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

A.2 a one dimensional (vertical) **strip**, i.e., a rectangle of size $1 \times \widetilde{h}_j$ with

$$\widetilde{h}_j = a_j - \widetilde{w}_j H$$

(possibly only one part is created)

B. Subdivide the **bin** into **two parts**:

B.1 a **"large" portion** of size $W_\ell (= \sum_{j \in J} \widetilde{w}_j) \times H$ that allocates the rectangles;

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

A. Split each **area** $a_j,\ j \in J$, into **two parts**:

A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

$$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

A.2 a one dimensional (vertical) **strip**, i.e., a rectangle of size $1 \times \widetilde{h}_j$ with

$$\widetilde{h}_j = a_j - \widetilde{w}_j H$$

(possibly only one part is created)

B. Subdivide the **bin** into **two parts**:

B.1 a **"large" portion** of size $W_\ell (= \sum_{j \in J} \widetilde{w}_j) \times H$ that allocates the rectangles;

B.2 a **"small" portion** of size $W_s (= W - W_\ell) \times H$,

whose W_s columns, of size $1 \times H$, are treated as W_s 1-dimensional bins:

consecutively allocate the one dimensional strips to the 1-dimensional bins

3. Theoretical analysis: a 3-approx algorithm for P0

- The general philosophy of the algorithm consists of the following phases:

  A. Split each **area** $a_j, \ j \in J$, into **two parts**:

  A.1 a **"large" rectangle** of size $\widetilde{w}_j \times H$ (H the height of the bin) , with

  $$\widetilde{w}_j = \left\lfloor \frac{a_j}{H} \right\rfloor, \text{ and}$$

  A.2 a one dimensional (vertical) **strip**, i.e., a rectangle of size $1 \times \widetilde{h}_j$ with

  $$\widetilde{h}_j = a_j - \widetilde{w}_j H$$

  (possibly only one part is created)

  B. Subdivide the **bin** into **two parts**:

  B.1 a **"large" portion** of size $W_\ell (= \sum_{j \in J} \widetilde{w}_j) \times H$ that allocates the rectangles;

  B.2 a **"small" portion** of size $W_s (= W - W_\ell) \times H$,

  whose W_s columns, of size $1 \times H$, are treated as W_s 1-dimensional bins:

  consecutively allocate the one dimensional strips to the 1-dimensional bins

  by further splitting only when necessary.

  C. Post-optimize the solution. (Not needed for the worst-case guarantee.)

3. Theoretical analysis: a 3-approx algorithm for P0, example

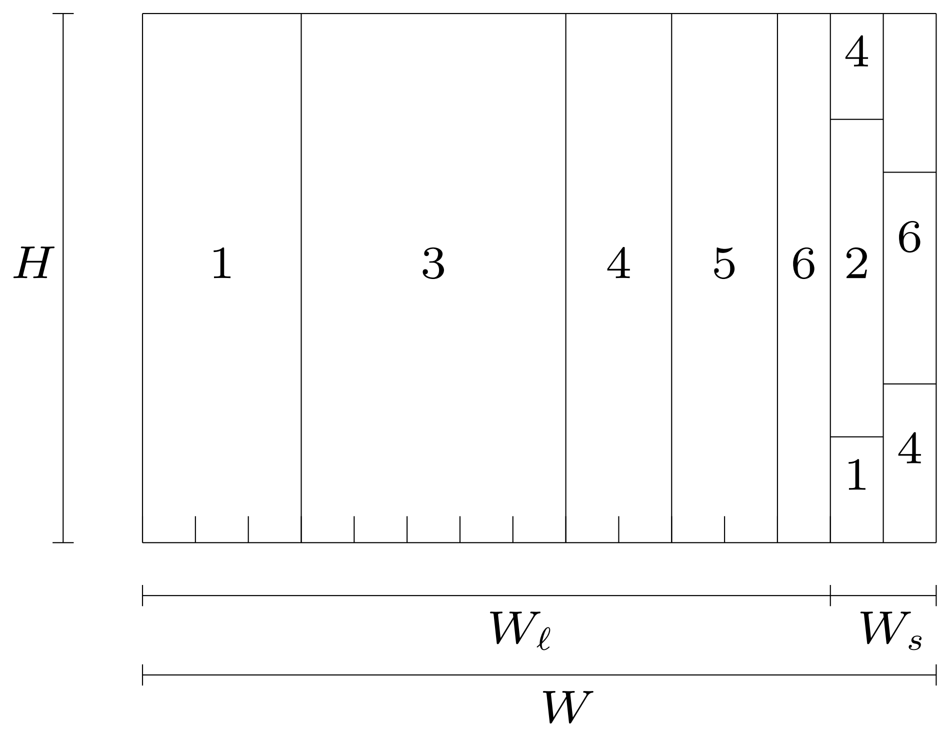
Instance with $W = 15, H = 10$

area	a_j	$\widetilde{w}_j$	$\widetilde{h}_j$
1	32	3	2
2	6	-	6
3	50	5	-
4	25	2	5
5	20	2	-
6	14	1	4

3. Theoretical analysis: a 3-approx algorithm for P0, example

Instance with $W = 15, H = 10$

area	a_j	$\widetilde{w}_j$	$\widetilde{h}_j$
1	32	3	2
2	6	-	6
3	50	5	-
4	25	2	5
5	20	2	-
6	14	1	4



3. Theoretical analysis: a 3-approx algorithm for P0, proof

- Consecutively pack the strips in the first column until a strip is found that does not fit;

3. Theoretical analysis: a 3-approx algorithm for P0, proof

- Consecutively pack the strips in the first column until a strip is found that does not fit; split such strip, packing the largest feasible part in the current column;

3. Theoretical analysis: a 3-approx algorithm for P0, proof

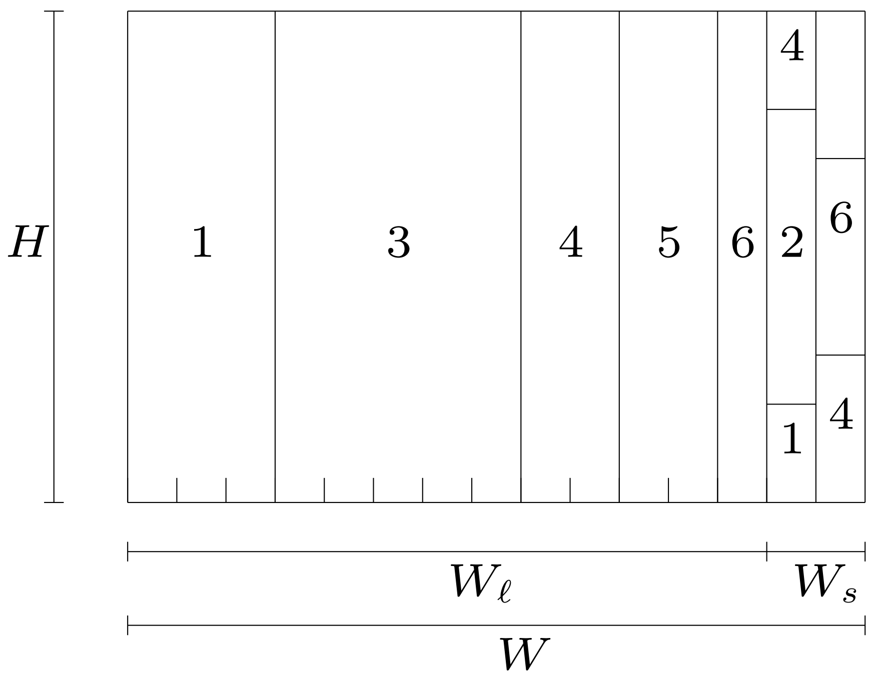
- Consecutively pack the strips in the first column until a strip is found that does not fit;
  split such strip, packing the largest feasible part in the current column;
  initialize the next column with the remaining part, and continue until all strips are packed.

3. Theoretical analysis: a 3-approx algorithm for P0, proof

- Consecutively pack the strips in the first column until a strip is found that does not fit;
  split such strip, packing the largest feasible part in the current column;
  initialize the next column with the remaining part, and continue until all strips are packed.
- **Hence** each strip is split at most once (recall that each strip has size $\widetilde{h}_j < H$).

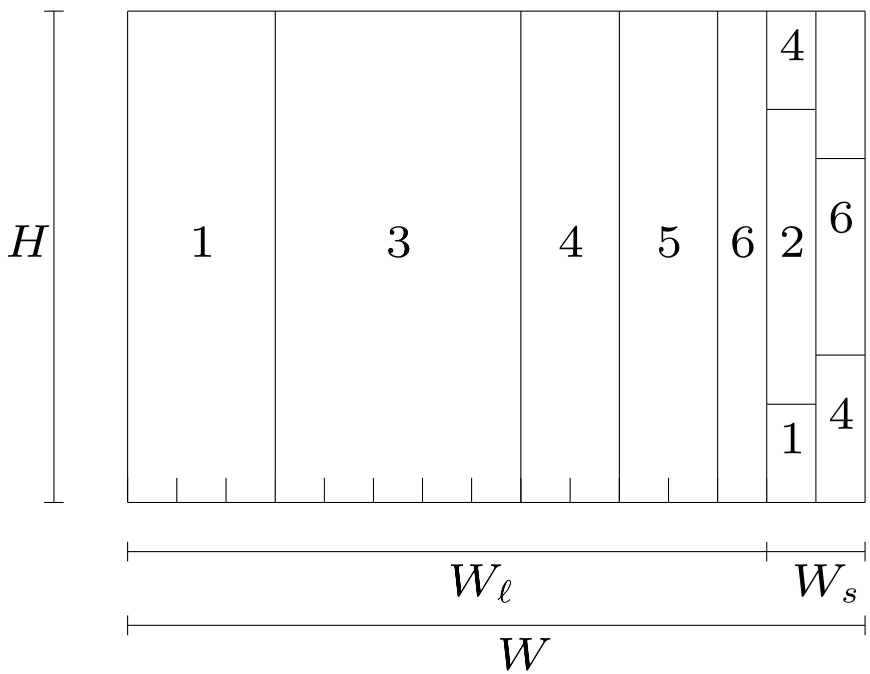
3. Theoretical analysis: a 3-approx algorithm for P0, proof

- Consecutively pack the strips in the first column until a strip is found that does not fit; split such strip, packing the largest feasible part in the current column; initialize the next column with the remaining part, and continue until all strips are packed.
- **Hence** each strip is split at most once (recall that each strip has size $\widetilde{h}_j < H$).
- **Hence** each area produces at most three sub-areas, which proves the worst case behavior.



3. Theoretical analysis: a 3-approx algorithm for P0, proof

- Consecutively pack the strips in the first column until a strip is found that does not fit; split such strip, packing the largest feasible part in the current column; initialize the next column with the remaining part, and continue until all strips are packed.
- **Hence** each strip is split at most once (recall that each strip has size $\widetilde{h}_j < H$).
- **Hence** each area produces at most three sub-areas, which proves the worst case behavior.



- It can be shown that the bound is tight.

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;
- it is trivial to construct instances for which such a solution does not exist;

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;
- it is trivial to construct instances for which such a solution does not exist;
- it is always possible to construct a solution that has at most **three rectangles per area**;

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;
- it is trivial to construct instances for which such a solution does not exist;
- it is always possible to construct a solution that has at most **three rectangles per area**;
- such a solution can be found in **linear time**;

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;
- it is trivial to construct instances for which such a solution does not exist;
- it is always possible to construct a solution that has at most **three rectangles per area**;
- such a solution can be found in **linear time**;
- what about the intermediate case (**two rectangles per area**)?

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;
- it is trivial to construct instances for which such a solution does not exist;
- it is always possible to construct a solution that has at most **three rectangles per area**;
- such a solution can be found in **linear time**;
- what about the intermediate case (**two rectangles per area**)?
- it can be proved that all instances with $n \leq 3$ **areas** have a feasible solution with two rectangles per area;

3. Conclusions of the theoretical analysis

Given an instance of Area Packing Problem (P0),

- it is **strongly $\mathcal{NP}$-complete** to decide whether there is a feasible solution that has a **single rectangle per area**;
- it is trivial to construct instances for which such a solution does not exist;
- it is always possible to construct a solution that has at most **three rectangles per area**;
- such a solution can be found in **linear time**;
- what about the intermediate case (**two rectangles per area**)?
- it can be proved that all instances with $n \leq 3$ **areas** have a feasible solution with two rectangles per area;
- Conjecture: Every instance possesses a feasible solution with at most two rectangles per area.

Post-optimization of the approximation algorithm

- Post-optimization is useful in practice when there are areas j such that:

  (i) both the associated large rectangle and one dimensional strip have been created, and

Post-optimization of the approximation algorithm

- Post-optimization is useful in practice when there are areas j such that:

  (i) both the associated large rectangle and one dimensional strip have been created, and

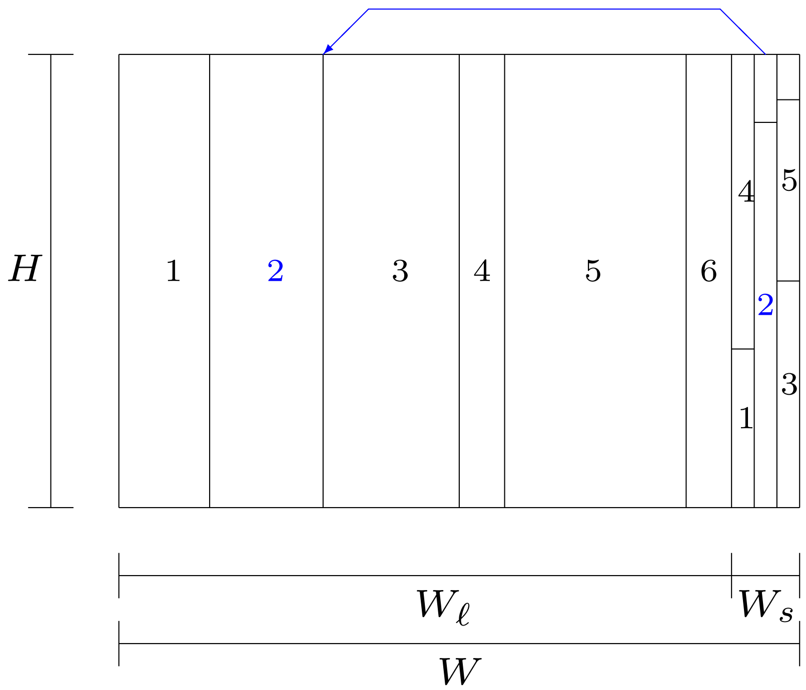
  (ii) strip j is packed alone in a 1-dimensional bin (column):

Post-optimization of the approximation algorithm

- Post-optimization is useful in practice when there are areas j such that:

  (i) both the associated large rectangle and one dimensional strip have been created, and

  (ii) strip j is packed alone in a 1-dimensional bin (column):

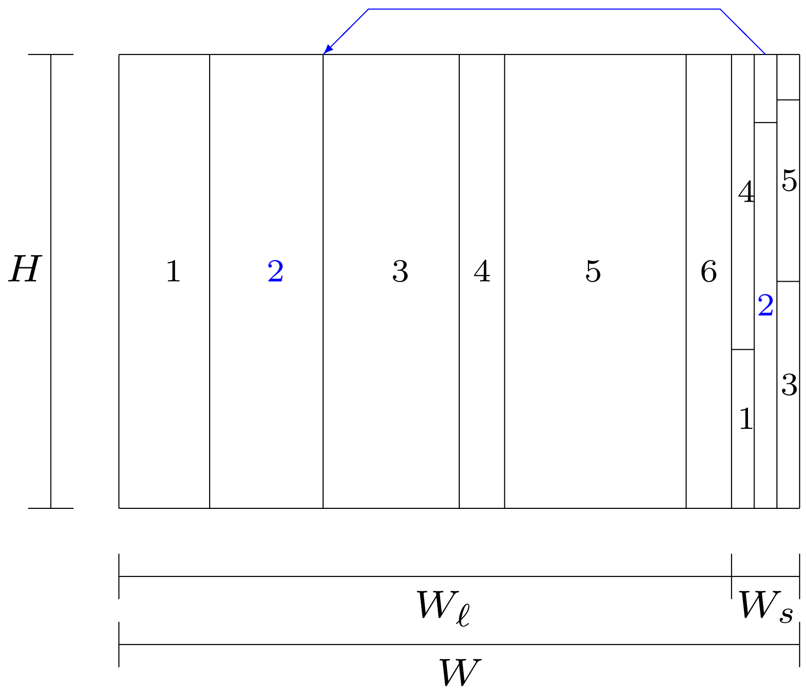


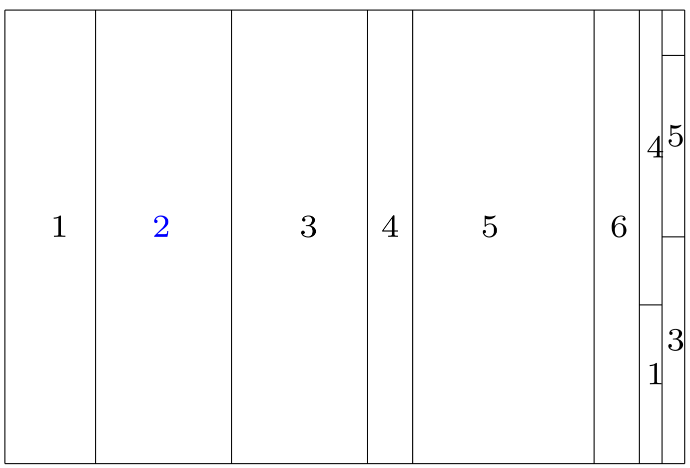
Post-optimization of the approximation algorithm

- Post-optimization is useful in practice when there are areas j such that:

  (i) both the associated large rectangle and one dimensional strip have been created, and

  (ii) strip j is packed alone in a 1-dimensional bin (column):





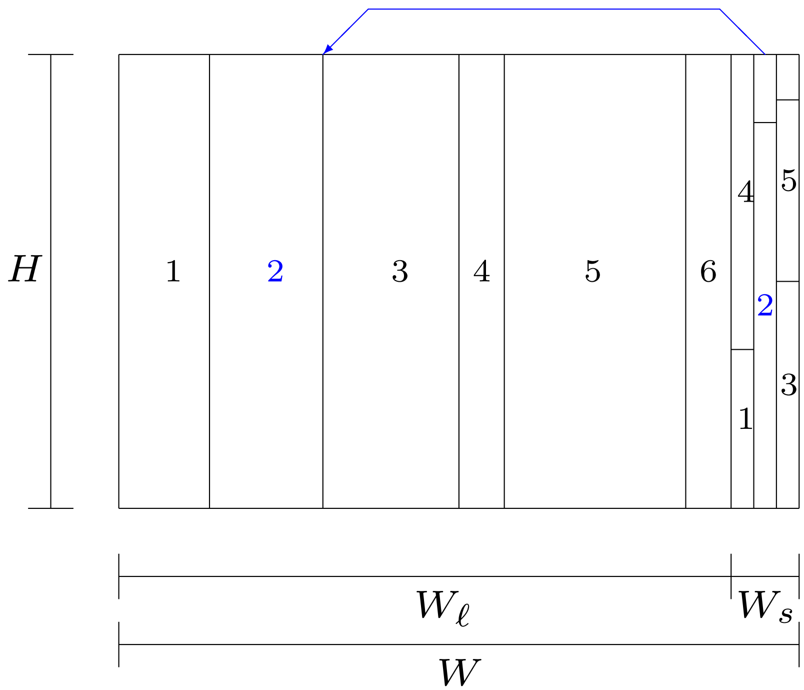
- Move the 1-dimensional bin that packs strip j close to the rectangle associated with area j.

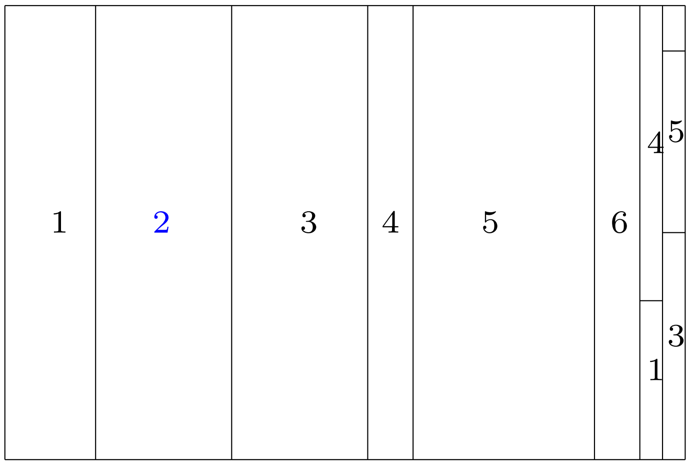
Post-optimization of the approximation algorithm

- Post-optimization is useful in practice when there are areas j such that:

  (i) both the associated large rectangle and one dimensional strip have been created, and

  (ii) strip j is packed alone in a 1-dimensional bin (column):





- Move the 1-dimensional bin that packs strip j close to the rectangle associated with area j.

- $\Rightarrow$ New solution in which area j is packed with a unique rectangle $(\widetilde{w}_j + 1) \times H$.

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j $(j \in J)$, m_j sub-areas, each having a specified integer value

  $$a_{jl} \ (j \in J,\, l \in L_j = \{1, \ldots, m_j\}),$$

  are given in input, such that

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j ($j \in J$), m_j sub-areas, each having a specified integer value

$$a_{jl} \ (j \in J, l \in L_j = \{1, \ldots, m_j\}),$$

are given in input, such that

$$\sum_{l \in L_j} a_{jl} = a_j \ \forall \ j \in J$$

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j $(j \in J)$, m_j sub-areas, each having a specified integer value

$$a_{jl} \; (j \in J, \, l \in L_j = \{1, \ldots, m_j\}),$$

are given in input, such that

$$\sum_{l \in L_j} a_{jl} = a_j \; \forall \; j \in J$$

- The sub-areas cannot be split.

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j $(j \in J)$, m_j sub-areas, each having a specified integer value

$$a_{jl} \; (j \in J, \, l \in L_j = \{1, \ldots, m_j\}),$$

are given in input, such that

$$\sum_{l \in L_j} a_{jl} = a_j \; \forall \; j \in J$$

- The sub-areas cannot be split.

- For each area we must define one or more rectangles containing the sub-areas.

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j ($j \in J$), m_j sub-areas, each having a specified integer value

$$a_{jl} \; (j \in J, \, l \in L_j = \{1, \ldots, m_j\}),$$

are given in input, such that

$$\sum_{l \in L_j} a_{jl} = a_j \;\forall\; j \in J$$

- The sub-areas cannot be split.
- For each area we must define one or more rectangles containing the sub-areas.
- This can make it impossible to completely pack all areas.

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j $(j \in J)$, m_j sub-areas, each having a specified integer value

$$a_{jl} \; (j \in J, \, l \in L_j = \{1, \ldots, m_j\}),$$

  are given in input, such that

$$\sum_{l \in L_j} a_{jl} = a_j \; \forall \; j \in J$$

- The sub-areas cannot be split.
- For each area we must define one or more rectangles containing the sub-areas.
- This can make it impossible to completely pack all areas.

(II) Each sub-area has a profit (priority):

4. The real-world problems

Three main differences in the telecommunication problems at hand:

(I) The areas cannot be arbitrarily split:

- For each area a_j $(j \in J)$, m_j sub-areas, each having a specified integer value

$$a_{jl} \ (j \in J, l \in L_j = \{1, \ldots, m_j\}),$$

are given in input, such that

$$\sum_{l \in L_j} a_{jl} = a_j \ \forall \ j \in J$$

- The sub-areas cannot be split.
- For each area we must define one or more rectangles containing the sub-areas.
- This can make it impossible to completely pack all areas.

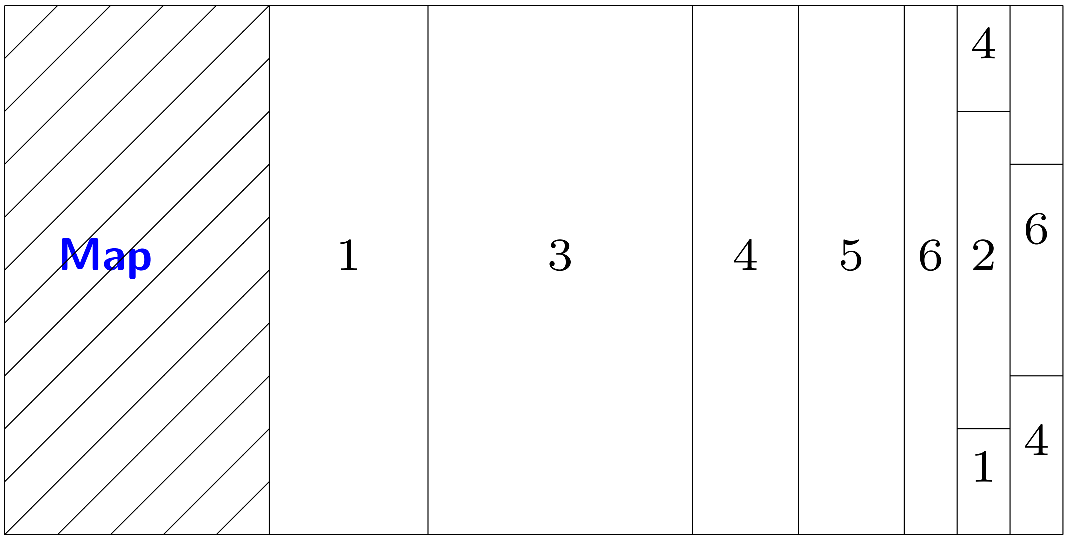
(II) Each sub-area has a profit (priority):

- The objective function is to maximize the total packed profit.

4. The real world problems

(III) The mapping of the packing must be stored in the frame.

- Each packed rectangle requires additional information (size and position of the rectangle, pointer to the associated area, . . .);

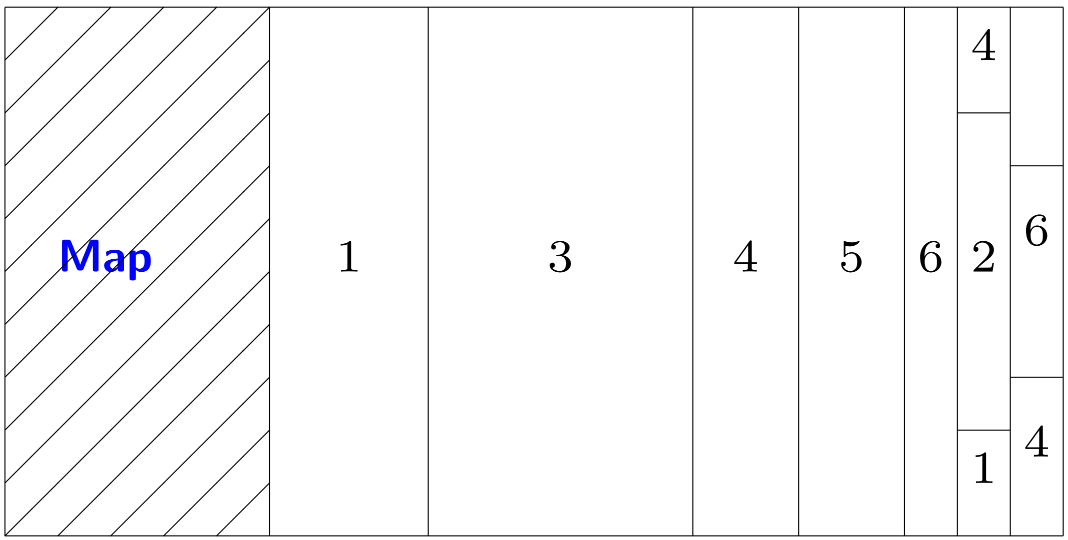


- minimizing the number of rectangles leads to minimizing the size of the map. **However**.

4. The real world problems

(III) The mapping of the packing must be stored in the frame.

- Each packed rectangle requires additional information (size and position of the rectangle, pointer to the associated area, . . .);



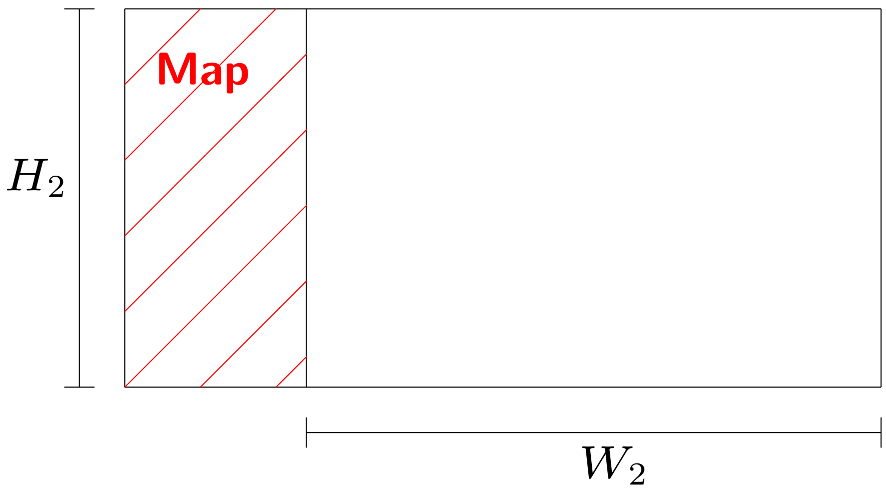
- minimizing the number of rectangles leads to minimizing the size of the map. **However**.
- **the actual size of the map can only be computed once the packing is known**.

4. Real-world problems: P1 and P2 (Distributed Permutation Zone)

Two possible map structures have been investigated:

4. Real-world problems: P1 and P2 (Distributed Permutation Zone)

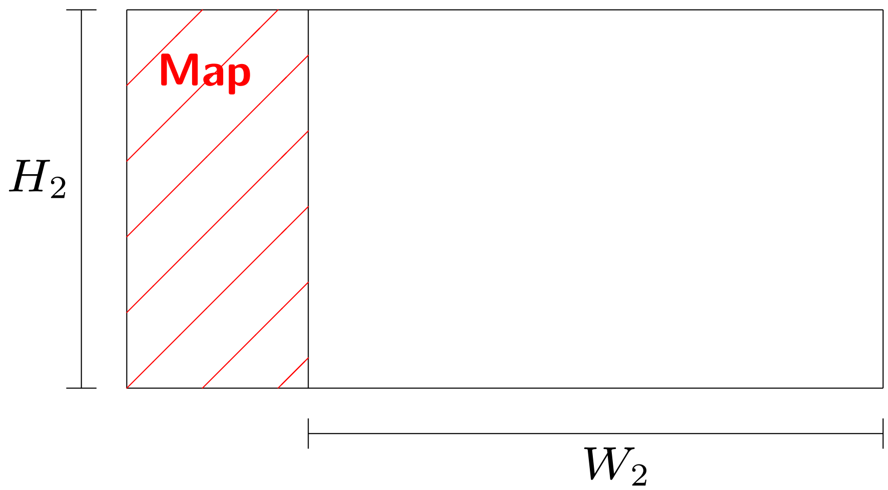
Two possible map structures have been investigated:



$\longleftarrow$ Bin structure for **Problem P1**

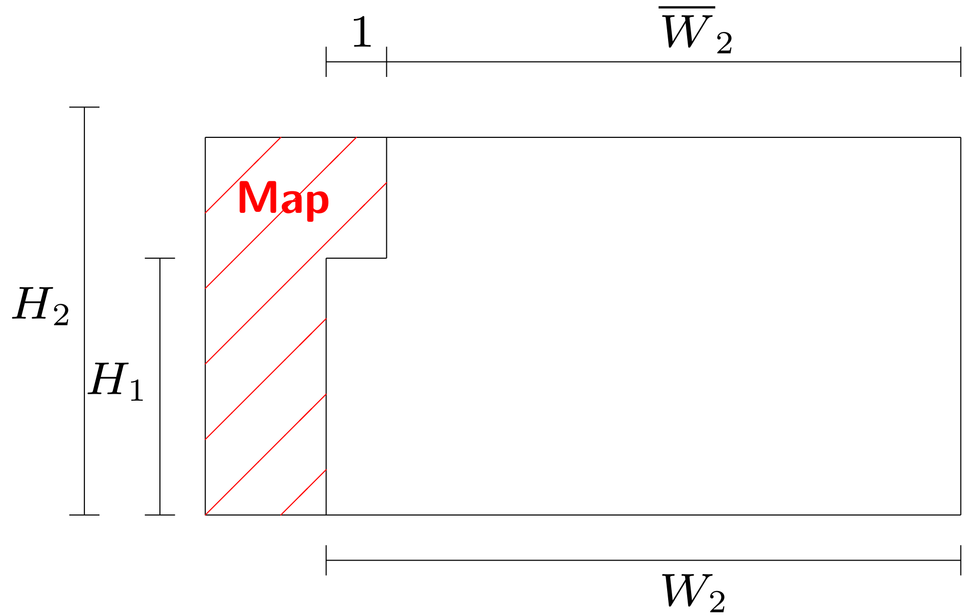
4. Real-world problems: P1 and P2 (Distributed Permutation Zone)

Two possible map structures have been investigated:



⟵ Bin structure for **Problem P1**

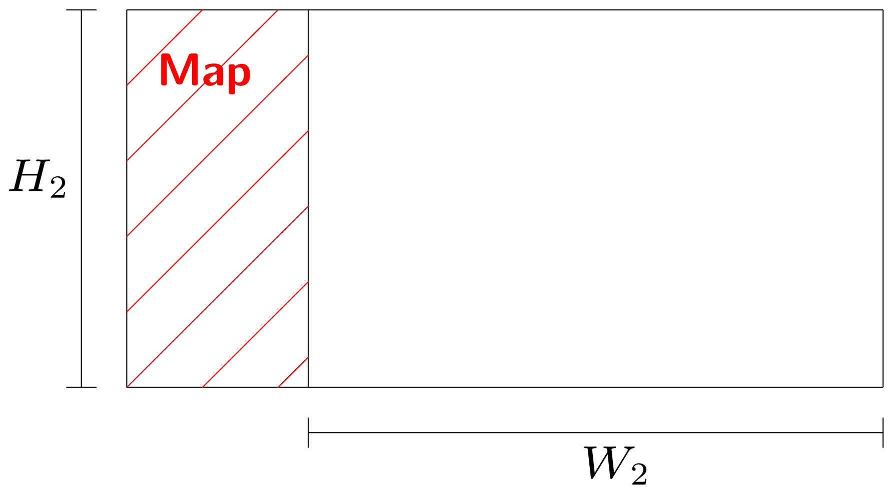
Bin structure for **Problem P2** ⟶



P2 is a generalization of **P1**

4. Real-world problems: P1 and P2 (Distributed Permutation Zone)

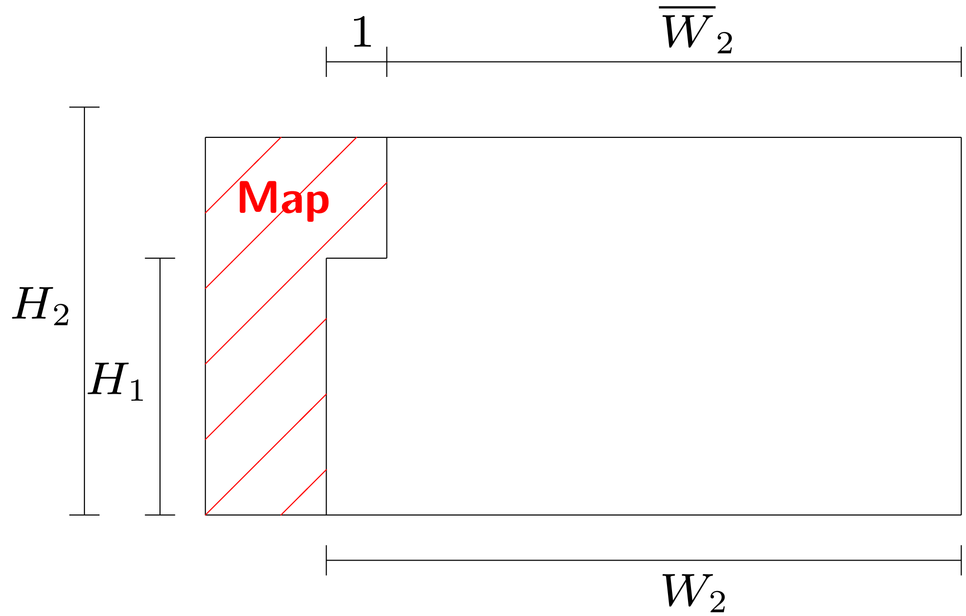
Two possible map structures have been investigated:



$\longleftarrow$ Bin structure for **Problem P1**

Bin structure for **Problem P2** $\longrightarrow$

P2 is a generalization of **P1**



- A third real-world problem (**P3**) will be discussed later.

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);
  - transmit the corresponding packets.

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);
  - transmit the corresponding packets.
- The bad news is that

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);
  - transmit the corresponding packets.
- The bad news is that each **transmission takes 1 millisecond**, i.e.,

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);
  - transmit the corresponding packets.
- The bad news is that each **transmission takes 1 millisecond**, i.e.,
- each instance must be completely solved (packing and map) within **1 millisecond!**

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);
  - transmit the corresponding packets.
- The bad news is that each **transmission takes 1 millisecond**, i.e.,
- each instance must be completely solved (packing and map) within **1 millisecond!**

  (Although real instances are "small", this requirement was really tough!)

5. Evaluation of the technological constraints

- The planned system must use sets of standard PCs;
- each PC must perform **500 transmissions per second**, i.e.,
- every **2 milliseconds** it is necessary to
  - read the input;
  - execute the algorithm;
  - produce the output (packing and map);
  - transmit the corresponding packets.
- The bad news is that each **transmission takes 1 millisecond**, i.e.,
- each instance must be completely solved (packing and map) within **1 millisecond!**

  (Although real instances are "small", this requirement was really tough!)

C. Cicconetti, L. Lenzini, A. Lodi, S. Martello, E. Mingozzi, M. Monaci.

Efficient two-dimensional data allocation in IEEE 802.16 OFDMA

Proceedings of IEEE INFOCOM 2010.

6. Development of heuristic algorithms: Stripes

- **Two fast heuristics** embedded in a **recursive algorithm**.

6. Development of heuristic algorithms: Stripes

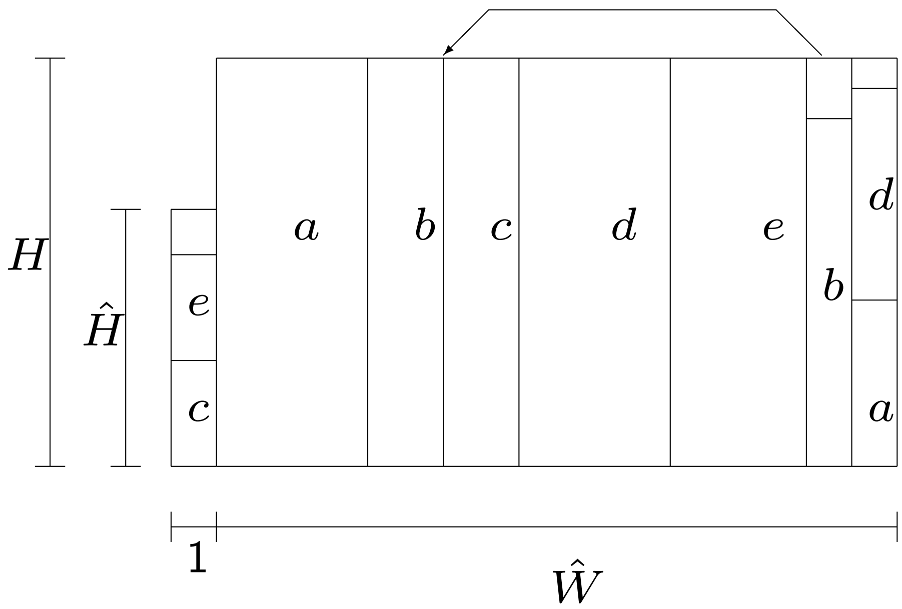
- **Two fast heuristics** embedded in a **recursive algorithm**.
- Description for the more general problem **P2**.

6. Development of heuristic algorithms: Stripes

- **Two fast heuristics** embedded in a **recursive algorithm**.
- Description for the more general problem **P2**.
- **First heuristic: Stripes**, derived from the 3-approx algorithm for **P0**:

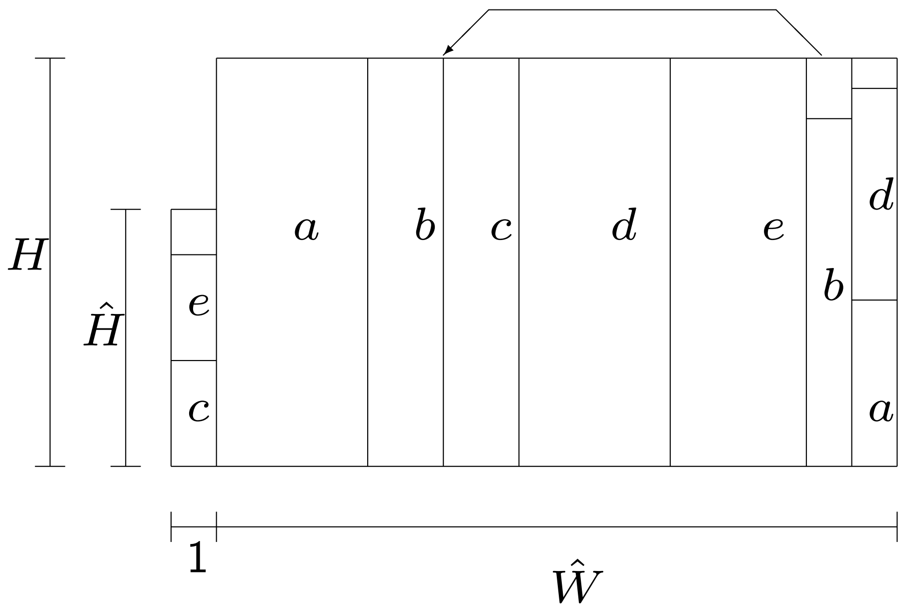
6. Development of heuristic algorithms: Stripes

- **Two fast heuristics** embedded in a **recursive algorithm**.
- Description for the more general problem **P2**.
- **First heuristic: Stripes**, derived from the 3-approx algorithm for **P0**:



6. Development of heuristic algorithms: Stripes

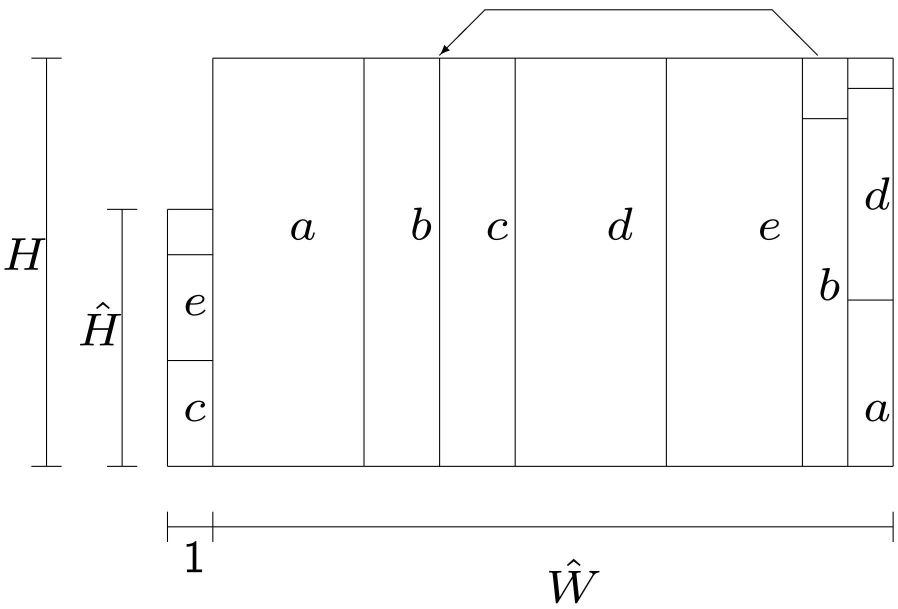
- **Two fast heuristics** embedded in a **recursive algorithm**.
- Description for the more general problem **P2**.
- **First heuristic: Stripes**, derived from the 3-approx algorithm for **P0**:



- the packing depends on the profit per unit area;

6. Development of heuristic algorithms: Stripes

- **Two fast heuristics** embedded in a **recursive algorithm**.
- Description for the more general problem **P2**.
- **First heuristic: Stripes**, derived from the 3-approx algorithm for **P0**:



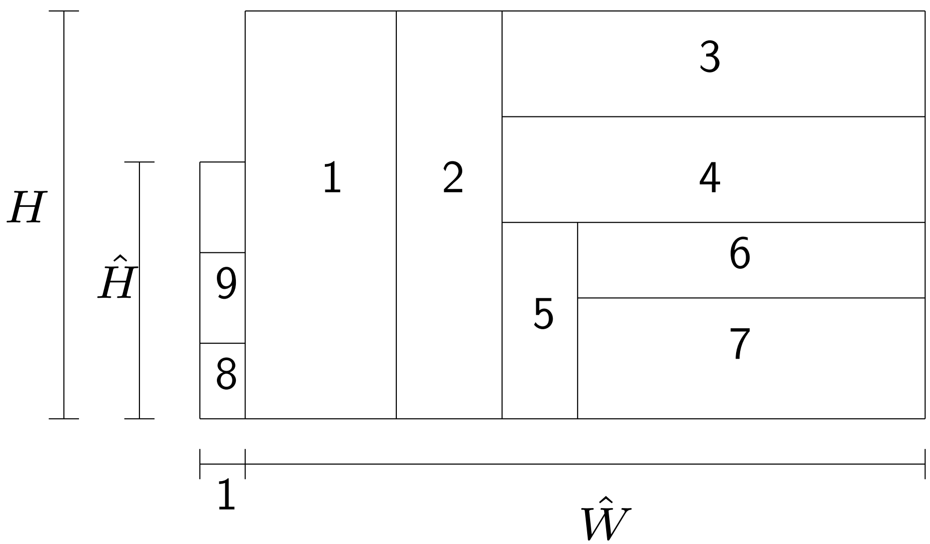
- the packing depends on the profit per unit area;
- the partial left column is used for the strips.

6. Development of heuristic algorithms: Tiles

- **Second heuristic: Tiles**, totally different philosophy, totally different solutions:

6. Development of heuristic algorithms: Tiles

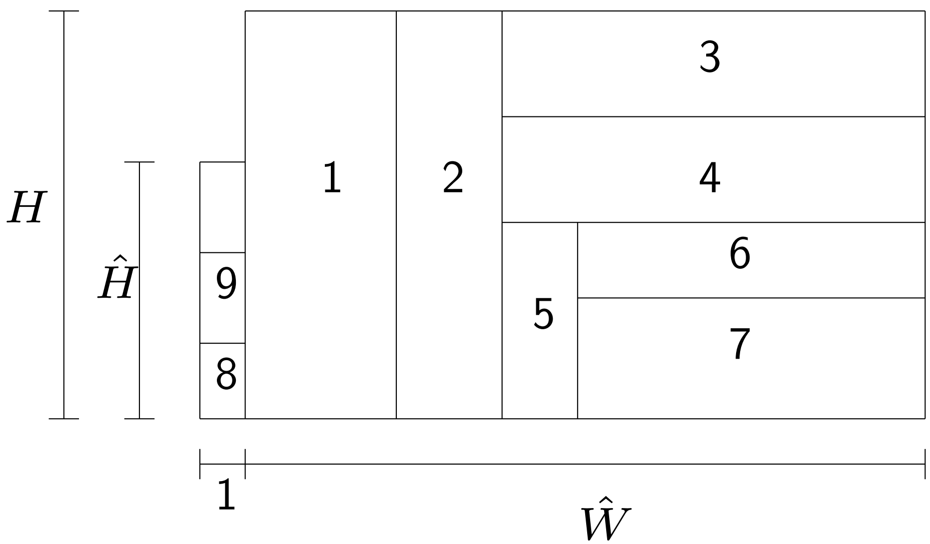
- **Second heuristic: Tiles**, totally different philosophy, totally different solutions:



- At each iteration, the best **vertical** **or** **horizontal** packing of an item is computed;

6. Development of heuristic algorithms: Tiles

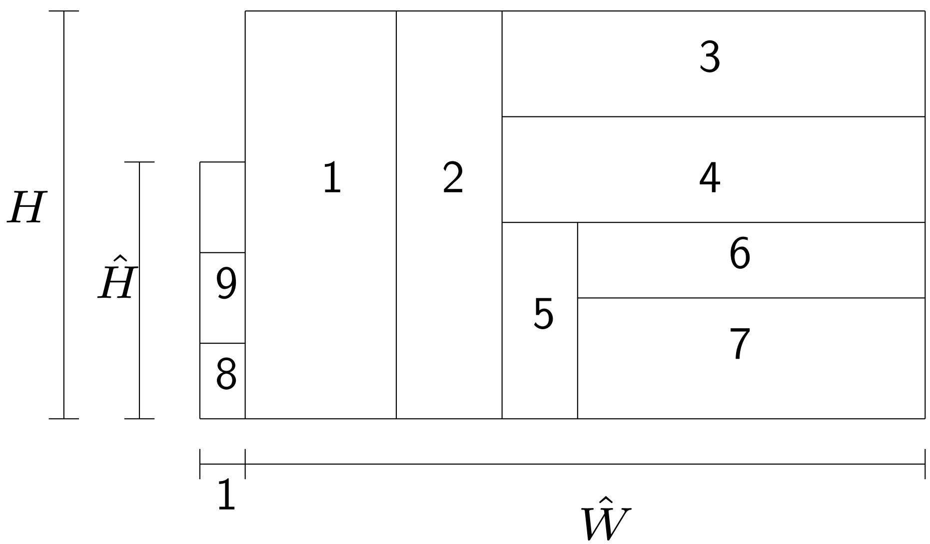
- **Second heuristic: Tiles**, totally different philosophy, totally different solutions:



- At each iteration, the best **vertical** **or** **horizontal** packing of an item is computed;
- best $\simeq$ minimum waste;

6. Development of heuristic algorithms: Tiles

- **Second heuristic: Tiles**, totally different philosophy, totally different solutions:



- At each iteration, the best **vertical or horizontal** packing of an item is computed;
- best $\simeq$ minimum waste;
- the partial left column is used for the residual sub-areas.

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

  sort the sub-items according to non-increasing value of their profit per unit area;

  initialize the incumbent solution to empty;

  initialize S to contain all sub-items;

  repeat (comment: iterate Tiles&Stripes)

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

  sort the sub-items according to non-increasing value of their profit per unit area;

  initialize the incumbent solution to empty;

  initialize S to contain all sub-items;

  repeat (comment: iterate Tiles&Stripes)

      define initial tentative values for W and H (comment: usable bin);

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

  sort the sub-items according to non-increasing value of their profit per unit area;

  initialize the incumbent solution to empty;

  initialize S to contain all sub-items;

  repeat (comment: iterate Tiles&Stripes)

      define initial tentative values for W and H (comment: usable bin);

      repeat (comment: try to pack the sub-item set S)

          execute Tiles(S) for the current W and H;

          execute Stripes(S) for the current W and H;

          compute the corresponding maps, and let σ be the best feasible solution, if any;

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

  sort the sub-items according to non-increasing value of their profit per unit area;

  initialize the incumbent solution to empty;

  initialize S to contain all sub-items;

  repeat (comment: iterate Tiles&Stripes)

      define initial tentative values for W and H (comment: usable bin);

      repeat (comment: try to pack the sub-item set S)

          execute Tiles(S) for the current W and H;

          execute Stripes(S) for the current W and H;

          compute the corresponding maps, and let σ be the best feasible solution, if any;

          if a feasible σ has been found **then**

              possibly update the incumbent with σ, and increase the current W and H

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

  sort the sub-items according to non-increasing value of their profit per unit area;

  initialize the incumbent solution to empty;

  initialize S to contain all sub-items;

  repeat (comment: iterate Tiles&Stripes)

      define initial tentative values for W and H (comment: usable bin);

      repeat (comment: try to pack the sub-item set S)

          execute Tiles(S) for the current W and H;

          execute Stripes(S) for the current W and H;

          compute the corresponding maps, and let σ be the best feasible solution, if any;

          if a feasible σ has been found **then**

              possibly update the incumbent with σ, and increase the current W and H

          else decrease the current W and H

      until σ includes all sub-items of S **or** limit on number of iterations has been reached;

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

sort the sub-items according to non-increasing value of their profit per unit area;

initialize the incumbent solution to empty;

initialize S to contain all sub-items;

repeat (comment: iterate Tiles&Stripes)

  define initial tentative values for W and H (comment: usable bin);

  repeat (comment: try to pack the sub-item set S)

    execute Tiles(S) for the current W and H;

    execute Stripes(S) for the current W and H;

    compute the corresponding maps, and let σ be the best feasible solution, if any;

    if a feasible σ has been found **then**

      possibly update the incumbent with σ, and increase the current W and H

    else decrease the current W and H

  until σ includes all sub-items of S **or** limit on number of iterations has been reached;

  if all sub-items of the instance have been allocated **then** terminate;

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

  sort the sub-items according to non-increasing value of their profit per unit area;

  initialize the incumbent solution to empty;

  initialize S to contain all sub-items;

  repeat (comment: iterate Tiles&Stripes)

      define initial tentative values for W and H (comment: usable bin);

      repeat (comment: try to pack the sub-item set S)

          execute Tiles(S) for the current W and H;

          execute Stripes(S) for the current W and H;

          compute the corresponding maps, and let σ be the best feasible solution, if any;

          if a feasible σ has been found **then**

              possibly update the incumbent with σ, and increase the current W and H

          else decrease the current W and H

      until σ includes all sub-items of S **or** limit on number of iterations has been reached;

      if all sub-items of the instance have been allocated **then** terminate;

      if all sub-items of S have been allocated **then** add sub-items to S;

      else remove sub-items from S

6. Development of heuristic algorithms: Tiles&Stripes

- **Overall heuristic: Tiles&Stripes**:

sort the sub-items according to non-increasing value of their profit per unit area;

initialize the incumbent solution to empty;

initialize S to contain all sub-items;

repeat (comment: iterate Tiles&Stripes)

    define initial tentative values for W and H (comment: usable bin);

    repeat (comment: try to pack the sub-item set S)

        execute Tiles(S) for the current W and H;

        execute Stripes(S) for the current W and H;

        compute the corresponding maps, and let σ be the best feasible solution, if any;

        if a feasible σ has been found **then**

            possibly update the incumbent with σ, and increase the current W and H

        else decrease the current W and H

    until σ includes all sub-items of S **or** limit on number of iterations has been reached;

    if all sub-items of the instance have been allocated **then** terminate;

    if all sub-items of S have been allocated **then** add sub-items to S;

    else remove sub-items from S

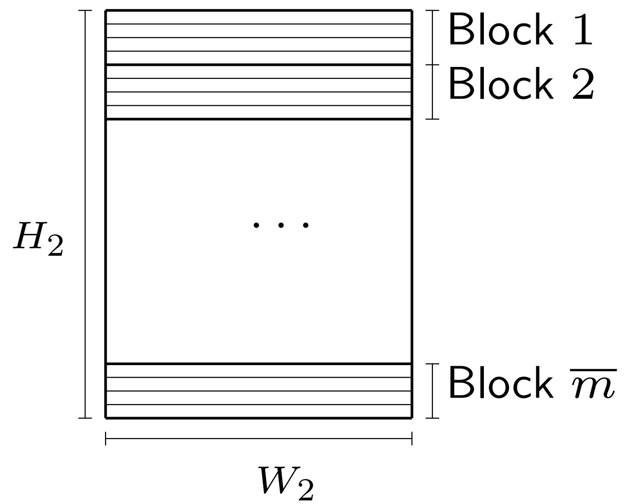
until a prefixed maximum number of iterations has been executed.

Back to the real world problems: P3 (Adjacent Permutation Zone)

- The structure of the bins is totally different, and is organized into **blocks** of standard sizes, with complicated packing rules.

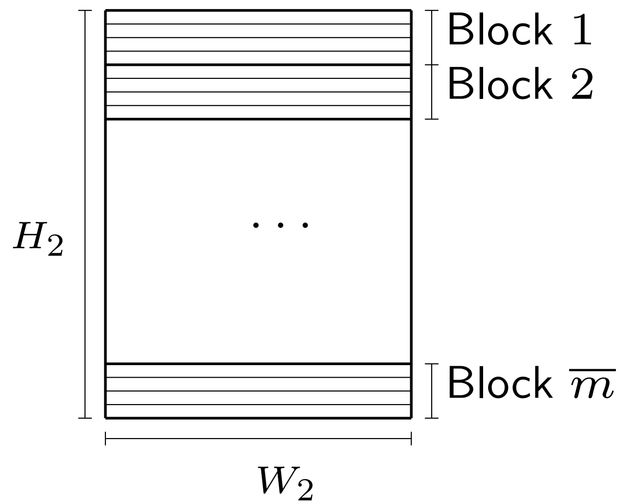
Back to the real world problems: P3 (Adjacent Permutation Zone)

- The structure of the bins is totally different, and is organized into **blocks** of standard sizes, with complicated packing rules.



Back to the real world problems: P3 (Adjacent Permutation Zone)

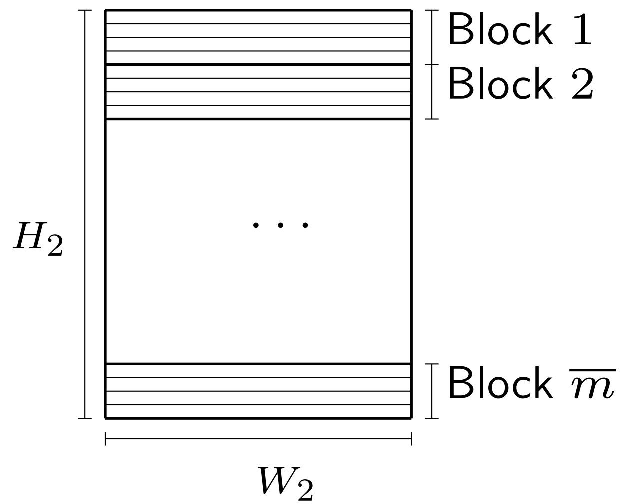
- The structure of the bins is totally different, and is organized into **blocks** of standard sizes, with complicated packing rules.



- Packing a maximum profit subset of packets is a **strongly $\mathcal{NP}$-hard** problem.
  Proof: transformation from the one-dimensional bin packing problem.

Back to the real world problems: P3 (Adjacent Permutation Zone)

- The structure of the bins is totally different, and is organized into **blocks** of standard sizes, with complicated packing rules.



- Packing a maximum profit subset of packets is a **strongly $\mathcal{NP}$-hard** problem.
  Proof: transformation from the one-dimensional bin packing problem.
- Attacked through Generalized Assignment Problems and maximum regret strategies.

C. Cicconetti, L. Lenzini, A. Lodi, S. Martello, E. Mingozzi, M. Monaci.

A Fast and Efficient Algorithm to Exploit Multi-user Diversity in IEEE 802.16 BandAMC.

Computer Networks, 2011.

7. Implementation and experimental evaluation on realistic scenarios.

- All algorithms have been coded in C
  and run on a 2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux.

7. Implementation and experimental evaluation on realistic scenarios.

- All algorithms have been coded in C
  and run on a 2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux.
- The computer networking group (University of Pisa) and the Nokia Siemens laboratory
  implemented a realistic simulator for both kinds of model:

7. Implementation and experimental evaluation on realistic scenarios.

- All algorithms have been coded in C
  and run on a 2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux.
- The computer networking group (University of Pisa) and the Nokia Siemens laboratory implemented a realistic simulator for both kinds of model:
- mix of data and voice users;

7. Implementation and experimental evaluation on realistic scenarios.

- All algorithms have been coded in C
  and run on a 2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux.
- The computer networking group (University of Pisa) and the Nokia Siemens laboratory implemented a realistic simulator for both kinds of model:
- mix of data and voice users;
- higher priority to packets directed to users with an ongoing voice conversation;

7. Implementation and experimental evaluation on realistic scenarios.

- All algorithms have been coded in C
  and run on a 2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux.
- The computer networking group (University of Pisa) and the Nokia Siemens laboratory implemented a realistic simulator for both kinds of model:
- mix of data and voice users;
- higher priority to packets directed to users with an ongoing voice conversation;
- different packet sizes for data and voice traffic;

7. Implementation and experimental evaluation on realistic scenarios.

- All algorithms have been coded in C
  and run on a 2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux.
- The computer networking group (University of Pisa) and the Nokia Siemens laboratory implemented a realistic simulator for both kinds of model:
- mix of data and voice users;
- higher priority to packets directed to users with an ongoing voice conversation;
- different packet sizes for data and voice traffic;
- different ratios between the number of users with data traffic and those with voice conversations.

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission
- Computing times in CPU milliseconds.

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission
- Computing times in CPU milliseconds.

			Optimality				Time (ms)	
Set	# inst	n	# potential	# opt	# good	avg z/U	avg	max
B1	23,040	[1,13]	23,040	22,114	22,846	0,9971	0.038	0.41
B2	23,040	[1,15]	23,040	21,840	23,014	0.9977	0.078	0.54
C1	23,210	[1,15]	10,158	8,340	13,719	0.9241	0.085	0.55
C2	23,317	[1,26]	2,512	1,788	4,544	0.8378	0.196	0.96

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission
- Computing times in CPU milliseconds.

			Optimality				Time (ms)	
Set	# inst	n	# potential	# opt	# good	avg z/U	avg	max
B1	23,040	[1,13]	23,040	22,114	22,846	0,9971	0.038	0.41
B2	23,040	[1,15]	23,040	21,840	23,014	0.9977	0.078	0.54
C1	23,210	[1,15]	10,158	8,340	13,719	0.9241	0.085	0.55
C2	23,317	[1,26]	2,512	1,788	4,544	0.8378	0.196	0.96

- # potential = instances for which
  (Total area) + (map space for a solution with one rectangle per packet) $\leq WH$;

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission
- Computing times in CPU milliseconds.

			Optimality				Time (ms)	
Set	# inst	n	# potential	# opt	# good	avg z/U	avg	max
B1	23,040	[1,13]	23,040	22,114	22,846	0,9971	0.038	0.41
B2	23,040	[1,15]	23,040	21,840	23,014	0.9977	0.078	0.54
C1	23,210	[1,15]	10,158	8,340	13,719	0.9241	0.085	0.55
C2	23,317	[1,26]	2,512	1,788	4,544	0.8378	0.196	0.96

- \# potential = instances for which
  (Total area) + (map space for a solution with one rectangle per packet) $\leq WH$;
- U = simple (and very optimistic) upper bound on the maximum area that can be packed;

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission
- Computing times in CPU milliseconds.

			Optimality				Time (ms)	
Set	# inst	n	# potential	# opt	# good	avg z/U	avg	max
B1	23,040	[1,13]	23,040	22,114	22,846	0,9971	0.038	0.41
B2	23,040	[1,15]	23,040	21,840	23,014	0.9977	0.078	0.54
C1	23,210	[1,15]	10,158	8,340	13,719	0.9241	0.085	0.55
C2	23,317	[1,26]	2,512	1,788	4,544	0.8378	0.196	0.96

- # potential = instances for which (Total area) + (map space for a solution with one rectangle per packet) $\leq WH$;
- U = simple (and very optimistic) upper bound on the maximum area that can be packed;
- # opt. = instances for which $z = U$;

7. Computational experiments, Probl. P2 (Distributed Permutation Zone)

- More than 90,000 instances representing different scenarios of transmission
- Computing times in CPU milliseconds.

			Optimality				Time (ms)	
Set	# inst	n	# potential	# opt	# good	avg z/U	avg	max
B1	23,040	[1,13]	23,040	22,114	22,846	0,9971	0.038	0.41
B2	23,040	[1,15]	23,040	21,840	23,014	0.9977	0.078	0.54
C1	23,210	[1,15]	10,158	8,340	13,719	0.9241	0.085	0.55
C2	23,317	[1,26]	2,512	1,788	4,544	0.8378	0.196	0.96

- # potential = instances for which
  (Total area) + (map space for a solution with one rectangle per packet) $\leq WH$;
- U = simple (and very optimistic) upper bound on the maximum area that can be packed;
- # opt. = instances for which $z = U$;
- # good = instances for which the ratio z/maximum packable area $\geq$ 0.9

Conclusions

- we have considered real-world packing problems arising in wireless telecommunications, and especially in orthogonal frequency division multiple access (OFDMA);

Conclusions

- we have considered real-world packing problems arising in wireless telecommunications, and especially in orthogonal frequency division multiple access (OFDMA);
- these real-world packing problems are challenging per se BUT they become even more difficult because of technological constraints which require to solve them within one millisecond;

Conclusions

- we have considered real-world packing problems arising in wireless telecommunications, and especially in orthogonal frequency division multiple access (OFDMA);
- these real-world packing problems are challenging per se BUT they become even more difficult because of technological constraints which require to solve them within one millisecond;
- we have defined a clean and easy-to-state packing problem (P0) that is the core of some of these problems;

Conclusions

- we have considered real-world packing problems arising in wireless telecommunications, and especially in orthogonal frequency division multiple access (OFDMA);
- these real-world packing problems are challenging per se BUT they become even more difficult because of technological constraints which require to solve them within one millisecond;
- we have defined a clean and easy-to-state packing problem (P0) that is the core of some of these problems;
- we have proved the complexity status of P0, and we have defined an approximation algorithm with worst-case guarantee;

Conclusions

- we have considered real-world packing problems arising in wireless telecommunications, and especially in orthogonal frequency division multiple access (OFDMA);
- these real-world packing problems are challenging per se BUT they become even more difficult because of technological constraints which require to solve them within one millisecond;
- we have defined a clean and easy-to-state packing problem (P0) that is the core of some of these problems;
- we have proved the complexity status of P0, and we have defined an approximation algorithm with worst-case guarantee;
- we have derived fast and efficient heuristics for the real-world problems.

Conclusions

- we have considered real-world packing problems arising in wireless telecommunications, and especially in orthogonal frequency division multiple access (OFDMA);
- these real-world packing problems are challenging per se BUT they become even more difficult because of technological constraints which require to solve them within one millisecond;
- we have defined a clean and easy-to-state packing problem (P0) that is the core of some of these problems;
- we have proved the complexity status of P0, and we have defined an approximation algorithm with worst-case guarantee;
- we have derived fast and efficient heuristics for the real-world problems.

Thank you for your attention