Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms

Maxence Delorme\textsuperscript{(1)}, Manuel Iori\textsuperscript{(2)}, Silvano Martello\textsuperscript{(1)}

(1) DEI, University of Bologna, Italy,
(2) DISMI, University of Modena and Reggio Emilia, Italy

BOLOGNA 2015
Outline

1 Introduction

2 Most common exact methods for solving the BPP and the CSP

3 Computational Results

4 Conclusion
The Bin Packing Problem (BPP)

Classical Bin Packing Problem

Given a set of weighted items and an unlimited number of identical capacitated bins, the Bin Packing Problem consists in packing all the items into the minimum number of bins.
The Bin Packing Problem (BPP)

Classical Bin Packing Problem

Given a set of weighted items and an unlimited number of identical capacitated bins, the Bin Packing Problem consists in packing all the items into the minimum number of bins.

- Items: 4, 4, 3, 3, 2, 2
- Bins: 9, 9, 9
The Bin Packing Problem (BPP)

**Classical Bin Packing Problem**
Given a set of weighted items and an unlimited number of identical capacitated bins, the Bin Packing Problem consists in packing all the items into the minimum number of bins.
The Cutting Stock Problem (CSP)

Classical Cutting Stock Problem

Given a set of order requirements, each requirement consisting in a demand and a width, and an unlimited number of identical rolls, the classical cutting stock problem consists of determining the smallest number of rolls that have to be cut in order to satisfy all the demands.
The Cutting Stock Problem (CSP)

Classical Cutting Stock Problem

Given a set of order requirements, each requirement consisting in a demand and a width, and an unlimited number of identical rolls, the classical cutting stock problem consists of determining the smallest number of rolls that have to be cut in order to satisfy all the demands.

Order requirements

<table>
<thead>
<tr>
<th>Demand</th>
<th>Width</th>
<th>Rolls</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Delorme, Iori, Martello
Bin Packing and Cutting Stock Problems
BOLOGNA 2015
The Cutting Stock Problem (CSP)

Classical Cutting Stock Problem

Given a set of order requirements, each requirement consisting in a demand and a width, and an unlimited number of identical rolls, the classical cutting stock problem consists of determining the smallest number of rolls that have to be cut in order to satisfy all the demands.

Order requirements

<table>
<thead>
<tr>
<th>Demand</th>
<th>Width</th>
<th>Rolls</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Rolls

<table>
<thead>
<tr>
<th>Demand</th>
<th>Width</th>
<th>Rolls</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>
Some applications for the BPP and the CSP

- Cutting materials in industry (wood, paper, aluminium ...)

- Loading when 1 dimension is considered (file storage on computers, container loading ...)

- Solving more difficult problems (vehicle routing, multi-dimensional BPP, ...)

Delorme, Iori, Martello
Context of our work

- BPP and CSP are more and more studied by researchers
  **The subject is of interest**
- Many techniques can be used to solve the BPP and the CSP
  **A survey is relevant**
- All the literature instances are solved
  **A will to create a new computational challenge**
Objectives of our work

- Gather in a survey the most important articles
- Test the efficiency of some exact methods proposed in the literature (Branch-and-Bound, Branch-and-Price, Pseudo-Polynomial models ...)
- Study the behaviour of those methods when the parameters of the test instances change
- Propose new instances that are difficult to solve in practice
The textbook BPP model solved by ILP solver

Roots in the seminal work by Kantorovich, Mathematical methods of organizing and planning production, *Management Science*, 1960 (originally from 1939)

\[
\begin{align*}
\min & \quad \sum_{i=1}^{m} y_i \\
\text{s.t.} & \quad \sum_{j=1}^{n} w_j x_{ij} \leq c_y i \quad (i = 1, \ldots, m), \\
& \quad \sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1, \ldots, n), \\
& \quad y_i \in \{0, 1\} \quad (i = 1, \ldots, m), \\
& \quad x_{ij} \in \{0, 1\} \quad (i = 1, \ldots, m; j = 1, \ldots, n).
\end{align*}
\]

Easy to implement, may be efficient on small instances \((n \leq 100)\)
Most common exact methods for solving the BPP and the CSP

Branch & Bound algorithms


Main idea: Explore (in a "smart" way) through an enumeration tree all the feasible packings
Smart because:
  ▶ Use of reduction procedures
  ▶ Use of a set of specially designed Lower Bounds (L1, L2, L3 ... L6)
  ▶ Use of fast and good heuristics to get Upper Bounds

Avoid the use of solvers. Generally good on small instances \((n \leq 100)\), or when the preprocessing is efficient.
Branch & Price algorithms

Gilmore, Gomory, A linear programming approach to the CSP, *Operations research*, 1963
Vance, Barnhart, Johnson, Nemhauser, Solving binary CSP by column generation and branch-and-bound, *Computational optimization and applications*, 1994

Main idea: Consider the Bin Packing Problem as a Set Covering Problem

**Set Covering Problem (SCP)**

Given a set of elements \{1, 2, ..., n\} (called the universe) and a family \(S\) of \(m\) sets whose union equals the universe, the set covering problem is to identify the smallest subfamily of \(S\) whose union equals the universe.

- The elements \{1, 2, ..., n\} are the items
- The family \(S\) is composed by every feasible bin (column generation used to generate bins)

Very good when the lower bound is equal to the optimum (true in 99% of the case), conjectured to be at most one bin away from the optimum (called Non-IRUP).
Most common exact methods for solving the BPP and the CSP

Constraints Programming

Shaw, A constraint for bin packing, 2004
Schaus, Régin, Van Schaeren, Dullaert, Raa, Cardinality reasoning for bin-packing constraint: Application to a tank allocation problem, 2012

Main idea: Use properties of feasible solutions to define the search domain.
Shaw did a very good job and created a constraint implemented in CPLEX.

Can be efficient on small instances \((n \leq 100)\) and has the advantage of easily allowing additional constraints.
Pseudo Polynomial Models solved by ILP solver

Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items
Pseudo Polynomial Models solved by ILP solver


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Graph representation of bins and items]
Most common exact methods for solving the BPP and the CSP

**Pseudo Polynomial Models solved by ILP solver**


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Graph](image)
Most common exact methods for solving the BPP and the CSP

**Pseudo Polynomial Models solved by ILP solver**


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Graph diagram](image.png)
Most common exact methods for solving the BPP and the CSP

### Pseudo Polynomial Models solved by ILP solver


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

- Items
  - 2
  - 3
  - 4
  - Full graph
    - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Delorme, Iori, Martello

Bin Packing and Cutting Stock Problems

BOLOGNA 2015
Most common exact methods for solving the BPP and the CSP

Pseudo Polynomial Models solved by ILP solver

- Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010
- Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

```
<table>
<thead>
<tr>
<th>Items</th>
<th>Full graph: 30 arcs, 10 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

```text
x 2
x 2
x 2
```
Most common exact methods for solving the BPP and the CSP

### Pseudo Polynomial Models solved by ILP solver


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Diagram showing bin packing problem and optimal solution](image)
Most common exact methods for solving the BPP and the CSP

**Pseudo Polynomial Models solved by ILP solver**


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

<table>
<thead>
<tr>
<th>Items</th>
<th>Reduced graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>x 2</td>
</tr>
<tr>
<td>3</td>
<td>x 2 0</td>
</tr>
<tr>
<td>4</td>
<td>x 2</td>
</tr>
</tbody>
</table>
Most common exact methods for solving the BPP and the CSP

Pseudo Polynomial Models solved by ILP solver

Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Diagram]

**Items**

- 2
- 3
- 4

**Reduced graph**

- 0 → 4 → 8
- 2 → 2 → 2
Most common exact methods for solving the BPP and the CSP

**Pseudo Polynomial Models solved by ILP solver**


Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010

Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Reduced graph diagram](image-url)
Most common exact methods for solving the BPP and the CSP

Pseudo Polynomial Models solved by ILP solver

Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Reduced graph](image)
Most common exact methods for solving the BPP and the CSP

Pseudo Polynomial Models solved by ILP solver

Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

![Reduced graph: 19 arcs, 9 nodes](image)
Most common exact methods for solving the BPP and the CSP

**Pseudo Polynomial Models solved by ILP solver**

Cambazard, O’Sullivan, Propagating the bin packing constraint using linear programming, 2010
Brandão, Pedroso, Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression, 2013

Main idea: Consider a bin as a path in a graph where arcs are items

```
<table>
<thead>
<tr>
<th>Items</th>
<th>Optimal solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>x 2</td>
</tr>
</tbody>
</table>
```
Pseudo Polynomial Models solved by ILP solver

Model

\[
\begin{align*}
\text{min} \quad & z \\
\text{s.t.} \quad & - \sum_{d \in \delta^{-}(e)} x_{de} + \sum_{f \in \delta^{+}(e)} x_{ef} = \begin{cases} 
    z & \text{if } e = 0; \\
    -z & \text{for } e = c; \\
    0 & \text{otherwise,}
\end{cases} \\
& \sum_{(d, d+ w_i) \in A'} x_{d, d+w_i} \geq b_i \quad (i = 1, ..., m), \\
& x_{de} \geq 0 \text{ and integer} \quad (d, e) \in A',
\end{align*}
\]

where \( \delta^{-}(e) \) (resp. \( \delta^{+}(e) \)) denotes the set of arcs entering (resp. emanating from) \( e \).

Very good when the lower bound is equal to the optimum and/or when the number of arcs is not too big \((c < 1000)\).
Instances

- Literature instances (1615)
  - 1210 instances from Sholl, Klein and Juergens (1997)
  - 28 instances from Schoenfield (2002)
  - 17 instances from Waescher and Gau (1996).

- Randomly generated instances (3840)
  - Variable number of items: 50,...,1000
  - Variable bin capacity: 50,...,1000
  - Variable parameters for the distribution of the items
  - 10 instances for each combination

- Augmented Non-IRUP (ANI) instances (250 + 250 Augmented IRUP (AI))
  - Variable number of items: 201, 402, 600, 801, and 1002
  - Each have its own maximum capacity: 2 500, 10 000, 20 000, 40 000, and 80 000
  - 50 instances for each of the 5 sets
  - An AI instance was created from each ANI instances by splitting 1 item so that the Non-IRUP is lost
Results for literature instances

Number of literature instances (average gap wrt lower bound) solved in less than one minute, on an Intel Xeon 3.10GHz with 8GB of RAM

- Best methods seem to be the B&P algorithm of Belov and the Pseudo Polynomial models
- All instances can be solved in less than 10 minutes by BELOV
Results for randomly generated instances

Number of random instances solved in less than one minute (average gap wrt lower bound) when varying $n$, on an Intel Xeon 3.10GHz with 8GB of RAM

- No change in the efficiency of the methods
- No difficulty for solving random instances even with “large” capacity and number of items
Computational Results

Results for ANI and AI instances

Number of difficult instances (ANI) solved in less than 1 hour (average gap wrt lower bound), on an Intel Xeon 3.10GHz with 8GB of RAM

| n(ANI) | n(AI) | $\sigma$ | ARCFLOW | | | | BISON | | | | BELOV | | | | VPSOLVER | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 201 | 202 | 2500 | 16 (0.7) | 44 (0.1) | 0 (1.0) | 0 (1.0) | 3 (0.9) | 50 (0.0) | 50 (0.0) | 47 (0.1) | 50 (0.0) | 45 (0.1) | 42 (0.2) | 6 (0.9) | 8 (0.8) |
| 402 | 403 | 10000 | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) | 0 (1.0) |
| 600 | 601 | 20000 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 801 | 802 | 40000 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 1002 | 1003 | 80000 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Overall | | | 16 (0.8) | 44 (0.6) | 0 (1.0) | 3 (1.0) | 51 (0.7) | 116 (0.4) | 53 (0.7) | 100 (0.5) |

- No change in the efficiency of the methods
- ANI instance are indeed hard to solve
Results for a subset of instances

Number of selected instances solved [average time in seconds] using different version of CPLEX, on an Intel Xeon 2.66GHz with 24 GB of RAM

|------|------------|------------|------------|------------|------------|------------|------------|-------------|

- Efficiency of pseudo-polynomial models highly depend on the performance of the ILP solver used
According to our results, the best methods to solve the BPP and the CSP seem to be ARCFLOW, VPSOLVER and the B&P algorithm of Belov et al.

- The use of pseudo-polynomial models for solving the BPP became relevant thanks to the performance of the ILP solver.
- There are some properties that make an instance difficult for the tool we tested.
- Some new BPP instances are “open”!