EURO2019: ROADEF Challenge 2018

Franco Peschiera, Nicolas Dupin, Luca Mossina

June 9, 2019
1 Introduction

2 Solving method

3 Simulated annealing

4 Results
Introduction

Franco Peschiera, Nicolas Dupin, Luca Mossina
EURO2019: ROADEF Challenge 2018
Motivation

- Column generation heuristic for this cutting problem?
Motivation

- Column generation heuristic for this cutting problem?
- Sequence constraints seem a priori a difficult point
Motivation

- Column generation heuristic for this cutting problem?
- Sequence constraints seem a priori a difficult point
- Using LocalSolver? Good structure for sequence constraints.
Motivation

- Column generation heuristic for this cutting problem?
- Sequence constraints seem a priori a difficult point
- Using LocalSolver? Good structure for sequence constraints.
- But some guys from LocalSolver are in the competition (and performed very well in the sprint phase), bad idea to challenge them with their techniques.
Motivation

- Column generation heuristic for this cutting problem?
- Sequence constraints seem a priori a difficult point
- Using LocalSolver? Good structure for sequence constraints.
- But some guys from LocalSolver are in the competition (and performed very well in the sprint phase), bad idea to challenge them with their techniques.
- Find another approach.
Solving method
Solving method

- Tree-structure modelling of guillotine cuts.
Solving method

- Tree-structure modelling of guillotine cuts.
- Construction heuristics. An implementation with a MIP formulation or greedy heuristics.
Solving method

- Tree-structure modelling of guillotine cuts.
- Construction heuristics. An implementation with a MIP formulation or greedy heuristics.
- Simulated annealing for repairing sequence and defect constraints and for improving solutions.
Solution encoding

A list of trees representing all the needed jumbos to cut. Each arc in tree represents a cut being made to a given node and each leaf node represents a waste or an item.
Simulated annealing
Allowed moves

- Insertion: extracting a node from one place and inserting it before another node.
Allowed moves

- **Insertion**: extracting a node from one place and inserting it before another node.
- **Exchange**: swapping two nodes’ positions.
Example (1)

A simple insertion of nodes of the same depth.
Example (2)

An insertion between nodes of different depth including the rotation of the node.
Candidate fitness

To measure if a candidate swap is an improvement three criteria are used:

- sequence violations.

A weighted sum of all three is calculated.
Candidate fitness

To measure if a candidate swap is an improvement three criteria are used:

- sequence violations.
- defect violations.

A weighted sum of all three is calculated.
Candidate fitness

To measure if a candidate swap is an improvement three criteria are used:

- sequence violations.
- defect violations.
- empty space used.

A weighted sum of all three is calculated.
Candidate choice

At each iteration, a random candidate is chosen using adjusted fitness quality as probability.

For chosen candidate, a simulated annealing function is applied to decide if the swap is done or not.
For small instances (most A instances), relatively good solutions were found. For bigger instances (B instances) finding feasible solutions from the incomplete initial solutions proved challenging.
Slow improvements

The methodology offered somewhat fast initial solutions but had trouble improving the solution. Convergence for B12 instance:
### Instance B results

Instances not solved after one hour are shown with a “-”.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>22178118</td>
</tr>
<tr>
<td>B2</td>
<td>88894055</td>
</tr>
<tr>
<td>B3</td>
<td>92990513</td>
</tr>
<tr>
<td>B4</td>
<td>50024725</td>
</tr>
<tr>
<td>B5</td>
<td>-</td>
</tr>
<tr>
<td>B6</td>
<td>-</td>
</tr>
<tr>
<td>B7</td>
<td>58810599</td>
</tr>
<tr>
<td>B8</td>
<td>122042899</td>
</tr>
<tr>
<td>B9</td>
<td>-</td>
</tr>
<tr>
<td>B10</td>
<td>131248823</td>
</tr>
<tr>
<td>B11</td>
<td>-</td>
</tr>
<tr>
<td>B12</td>
<td>85946167</td>
</tr>
<tr>
<td>B13</td>
<td>-</td>
</tr>
<tr>
<td>B14</td>
<td>68570980</td>
</tr>
<tr>
<td>B15</td>
<td>-</td>
</tr>
</tbody>
</table>
Perspectives?

Better neighborhoods?

We lacked time to implement diversification strategies, a GRASP heuristic?

However, with unefficient neighborhoods and repairing strategies, does it make sense?

Dynamic programming, beam search seem to be better ideas for the problem . . .
Questions?

Thanks for your attention