Modelling a Steel Mill Slab Design Problem

Ian Miguel
Alan M Frisch

29 October 2013
Slab Design

• The mill can make $\sigma$ different slab sizes.
• Given $d$ input orders each specifying:
  • A *colour* (route through the mill).
  • A *weight*.
• Pack orders onto slabs such that the total slab capacity is minimised, subject to:
  • Capacity constraints.
  • Colour constraints.
Slab Design Constraints

• **Capacity Constraints:**
  • Total weight of orders assigned to each slab cannot exceed the slab’s capacity.

• **Colour Constraints:**
  • Each slab can contain at most $p$ of $k$ total colours.
  • Reason: expensive to cut slabs up to send them to different parts of the mill.
An Example

- Slab Sizes: \( \{1, 3, 4\} \) (\( \sigma = 3 \))
- Orders: \( \{o_a, \ldots, o_i\} \) as shown below (\( d = 9 \))
- Colours: \( \{\text{red, green, blue, orange, brown}\} \) (\( k = 5 \))
- \( p = 2 \)
A Solution

- Use 6 Slabs:

\[
\begin{array}{cccccc}
\text{(size 4)} & \text{(size 3)} & \text{(size 3)} & \text{(size 1)} & \text{(size 1)} & \text{(size 1)} \\
2 & 3 & 1 & 1 & 1 & 1 \\
h & b & e & f & g & c \\
a & (size 4) & (size 3) & (size 3) & (size 1) & (size 1) \\
\end{array}
\]

This is an optimal solution as it produces no wastage.
Abstractly, what must be found?

Here is a hint………

(size 4)  (size 3)  (size 3)  (size 1)  (size 1)  (size 1)
Abstractly, what must be found?

- A partitioning of the orders
- A function from each partition to an available slob size
What’s Interesting about this Problem?

• Many problems exhibit flexibility in portions of their structure.
• Example: the number required of a certain type of variable.
• Flexibility must be resolved during the solution process.
• Slab design is a representative example of this type of problem:
  • Number of slabs and size of each is unknown.
What’s Interesting about this Problem?

• Symmetry.
  • Some instance-based.

• Channelling.
  • Between matrices of decision variables.

• Implied Constraints.
  • Some of which follow from the symmetry-breaking.
How Many Slabs?

• Number of slabs is not fixed.
  – If there are $d$ orders, then an optimal solution uses no more than $d$ slabs.

• An array, $S$, of slab variables: $s[1], \ldots, s[d]$
  • Domains contain all $\sigma$ available slab sizes.

• Objective

$$\text{minimise} \quad \sum_{i} s[i] = optVar$$
Some slab variables may be *redundant*:

- 0 is added to the domain of each $s[i]$.
- If slab $i$ is not necessary to solve the problem, $s[i] = 0$. 
The Order Matrix: $O$

$O[i,j] = 1$ iff order $j$ is placed on slab $i$.

<table>
<thead>
<tr>
<th></th>
<th>$O_a$</th>
<th>$O_b$</th>
<th>$O_c$</th>
<th>$O_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_3$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Capacity constraints:

$\forall i \sum_j \text{weight}(o_j) \times O[i,j] \leq s[i]$

$\forall j \sum_i O[i,j] = 1$  
Every order goes on exactly 1 slab  
Each column sums to 1
Example

<table>
<thead>
<tr>
<th></th>
<th>$o_a$</th>
<th>$o_b$</th>
<th>$o_c$</th>
<th>$o_d$</th>
<th>$o_e$</th>
<th>$o_f$</th>
<th>$o_g$</th>
<th>$o_h$</th>
<th>$o_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>Orange</th>
<th>Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Colour Matrix: $C$

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>Orange</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$s_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_3$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Colour constraints

$\forall i \sum_{j} C[i, j] \leq p$

- Channelling: for all $i, j$

$$O[i, j] = 1 \rightarrow C[i, \text{colour}(j)] = 1$$

- Why $\rightarrow$ and not $\leftrightarrow$?
# Solution

<table>
<thead>
<tr>
<th>O</th>
<th>$o_a$</th>
<th>$o_b$</th>
<th>$o_c$</th>
<th>$o_d$</th>
<th>$o_e$</th>
<th>$o_f$</th>
<th>$o_g$</th>
<th>$o_h$</th>
<th>$o_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$s_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>Orange</th>
<th>Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$s_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_4$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Symmetry

• Slabs are indistinguishable.
• So model suffers from symmetry:
  • Counteract with symmetry-breaking constraints:
    \[ s[1] \geq s[2], \quad s[2] \geq s[3], \ldots \]
Symmetry

• Slab variables assigned the same size are indistinguishable.
• Break this symmetry with the constraint: If $s[i]$ $s[i+1]$ are assigned the same value, then row $i$ of the order matrix must be lexicographically less than or equal to row $i+1$.
  – E.g. 1001 $\geq$ 0110.

$$s[i] = s[i+1] \rightarrow O[i,\_] \leq_{lex} O[i+1,\_]$$
Instance Symmetry

• Orders with same weight, colour are symmetrical.
• Order symmetrical columns lexicographically.
• For all $j, j'$ such that $j < j'$, $\text{weight}(o_j) = \text{weight}(o_{j'})$ and $\text{colour}(o_j) = \text{colour}(o_{j'})$ add the constraint
  \[ O[\_, j] \leq_{\text{lex}} O[\_, j'] \]
Implied Constraints

• Combined weight of input orders is a lower bound on optimisation variable:

\[ \sum_{j} weight(o_j) \leq optVar \]

• How does this help?
  • If find a solution of this quality, can stop immediately.
Implied Constraints

- Lower bound on number of slabs required:
  \[ \text{mins labs} = \frac{\sum \text{weight}(o_j)}{\text{largest slab size}} \]

- With symmetry-breaking constraints, implies unary constraints on slab variables:
  - Since: \( s[1] \geq s[2], \quad s[2] \geq s[3], \ldots \)
  - And: we use at least \( \text{mins labs} \) slabs.
  - Thus: each of the first \( \text{mins labs} \) slab variables are not zero

- Notice that the implied constraint follows from the symmetry-breaking constraint (among others).
Implied Constraints

- $AssWt_i$ is the weight of orders assigned to slab $i$.
  - Prune domains by reasoning about reachable values.
    - Also gives a maximum assigned weight.
    - So, revise lower bound on number of slabs required:
      \[
      \frac{\sum_i weight(o_i)}{\max AssWt}
      \]
  - Incorporate both size and colour information.
  - More powerful if done during search (future work).
Implied Constraints

- $Waste_i = s[i] - Ass Wt_i$  Unused portion of a slab.
- Upper bound on total waste:
  - Assume each order is assigned to an individual slab, with smallest size able to hold it.
  - Sum waste in each case: leads to upper bound for optimisation variable.
  - Upper bound on $Waste_i$ is the worst of these cases.
References


• www.csplib.org (Problem 38).