Modeling the Car Sequencing Problem

Gerrit Renker

The Robert Gordon University
Aberdeen
Requirements

- cars can be configured with various *options*
  - indicated by the *car type*
  - for each car type, there is a *demand* of cars.

- **capacity constraints:**
  - at most $M$ out of a sequence of $N$ cars may have option $i$

**goal:**
arrange the *sequence of all requested cars* such that none of the capacity constraints is violated
## Instance Data for the Car Sequencing Problem

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Type</th>
<th>Capacity M/N</th>
<th>Car Type 1</th>
<th>Car Type 2</th>
<th>Car Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunroof</td>
<td>1</td>
<td>2/3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Radio</td>
<td>2</td>
<td>3/4</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Air Cond.</td>
<td>3</td>
<td>2/3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| Number of Cars | 10 | 20 | 20 |

- **total of 50 cars:**
  - 30 cars with sunroofs
  - 30 cars with radios fitted
  - 40 cars with air conditioning
Car represents possible car types
- characterized by type and demand attributes

Slot: the sequence of Car types
context CarSP inv:

nbCars = car->size() and
nbSlots = slot->size() and
nbOptions = option->size()
context Option inv:
  type > 0 and type <= 3

context CarSP inv: -- uniqueness constraint
  option->isUnique(type)
context Car inv:

\[
\text{type} = 1 \implies (\text{demand} = 10 \text{ and } \text{option.type} = \{1, 2\})
\]

and

\[
\text{type} = 2 \implies (\text{demand} = 20 \text{ and } \text{option.type} = \{1, 3\})
\]

and

\[
\text{type} = 3 \implies (\text{demand} = 20 \text{ and } \text{option.type} = \{2, 3\})
\]
Computation of Derived Attributes

- can always be derived from other attributes

**context** Option **inv:**

capacity = M/N and
demand = car.demand->sum()
context CarSP inv:

option->forall(o | Sequence{1..(nbSlots - o.N + 1)}->forall(i |
  slot->subSequence(i, i + o.N - 1).car.option->count(o) <= o.M ))