Constraint Programming in Practice: Scheduling a Rehearsal

Barbara Smith
The Rehearsal Problem

- Originated at Lancaster University; see Adelson, Norman & Laporte, ORQ, 1976
- Sequence an orchestral rehearsal of 9 pieces of music with 5 players
- Players arrive just before the first piece they play in & leave just after the last piece
- Minimize total waiting time i.e. time when players are present but not currently playing
### Problem Data

<table>
<thead>
<tr>
<th>Piece</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Player 2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Player 3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Player 4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Player 5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Duration</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>
## Problem Data

<table>
<thead>
<tr>
<th>Piece</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Player 2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Player 3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Player 4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Player 5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Duration</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

Total waiting time: 49 time units
The Main Question

- Can we solve the rehearsal problem efficiently using constraint programming?
Constraint Satisfaction Problems

- A CSP consists of:
  - a set of *variables*, each with a set of possible values (its *domain*)
  - and a set of *constraints*: a constraint on a subset of the variables specifies which values can be simultaneously assigned to these variables
Solutions to a CSP

- A solution to a CSP is an assignment of a value to every variable in such a way that the constraints are satisfied.
- We might want just one solution (any solution).
- .. or all solutions.
- ... or an optimal solution.
Constraints

• A constraint affects a subset of the variables
• A constraint simply specifies the assignments to these variables that it allows
• Constraints are not limited e.g. to linear inequalities
• This generality allows CSPs to represent a wide range of problems
Examples

- \( x \cdot y = z \) where \( x, y, z \) are variables
- arithmetic expressions involving variables and constants
- \( x_i = 1 \Rightarrow x_{i+1} = 1 \) (i.e. if \( x_i = 1 \) then \( x_{i+1} = 1 \))
- logical constraints can express the logic of the problem directly
- \( t = \sum a_i x_i \) (\( t, a_i, x_i \) constants or variables)
- constraints on arrays of variables
- allDifferent(\( x_1, x_2, \ldots, x_n \))
Constraint programming

- Constraint programming systems, e.g. ILOG Solver, Eclipse, Sicstus Prolog... allow the programmer to:
  - define variables and their domains
  - specify the constraints, using predefined constraint types
  - define new constraints
  - solve the resulting CSP
Solving CSPs

- **Systematic *search***:
  - choose a variable, *var*, that has not yet been assigned a value
  - choose a value in the domain of *var* and assign it
  - backtrack to try another choice if this fails

- **Constraint propagation**:
  - derive new information from the constraints, including *var=val*
    - i.e. every other value has been removed from the domain of this variable
    - → remove values from the domains of future variables that can no longer be used because of this assignment
  - fail if any future variable has no values left
Termination

• Search terminates when
  • *either* every variable has been assigned a value: a solution has been found and we only wanted one
  • *or* there are no more choices to consider: there is no solution, or we have found them all
• Given long enough, the search *will* terminate in either case
Constraint Propagation Example

- Variables $x_1, x_2, \ldots, x_n$, domains $\{0,1\}$
- Constraints $x_i = 1 \Rightarrow x_{i+1} = 1, 1 \leq i \leq n$
- Variable $w$ defined by constraint $w = \sum_i d_i x_i$
- Domain of $w$ is calculated as $\{0, \ldots, \sum_i d_i\}$
- If 1 is assigned to $x_1$ i.e. 0 is removed from its domain, then
  - 0 is removed from the domain of $x_2$, then from $x_3$, ...
  - the lower bound on $w$ is raised each time, until the only value left is $\sum_i d_i$
  - every variable only has one value left, so gets assigned
Rehearsal Problem: Decision Variables

- We have to decide the order of the pieces
- Define variables $s_1, s_2, ..., s_n$ where $s_i = j$ if piece $i$ is in the $j^{th}$ position
- Domain of $s_i$ is $\{1, 2, ..., n\}$
- A valid sequence if \text{allDifferent}(s_1, s_2, ..., s_n) is true
- What about minimizing waiting time?
Optimization

- Include a variable, say $t$, for the objective
- Include constraints (and maybe new variables) linking the decision variables and $t$
- Find a solution in which the value of $t$ is (say) $t_0$
- Add a constraint $t < t_0$ (if minimizing)
- Find a new solution
- Repeat last 2 steps
- When there is no solution, the last solution found has been proved optimal
Rehearsal Problem: Objective

- How do we link the sequence variables \( s_1, s_2, \ldots, s_n \) with \( t \), the total waiting time?
- We need to know the waiting time for each player.
- For each player and each piece (that they don’t play) we need to know
  - whether the player is waiting while this piece is played
  - where this piece is in the sequence
  - whether the player is there then
  - i.e. if the player has arrived and has not yet left
New variables and constraints

• Where each piece is in the sequence
  • $d_j$ is the position in the sequence of piece $j$
  • $d_j = i$ iff $s_i = j$

• For each slot in the sequence, which players are playing
  • $p_{kj} = 1$ iff player $k$ plays the piece in slot $j$
  • $p_{kdj} = \pi_{kj}$ where $\pi_{kj} = 1$ iff player $k$ plays piece $j$
More new variables & constraints

- When each player arrives and leaves
  - \( a_{ki} = 1 \) iff player \( k \) has arrived by the start of slot \( i \)
  - \( l_{ki} = 1 \) iff player \( k \) leaves at the end of slot \( i \) or later
  - \( a_{k1} = p_{k1} \)
  - \( a_{ki} = 1 \) iff \( a_{k,i-1} = 1 \) or \( p_{ki} = 1 \)
  - similarly for \( l_{ki} \)

- Whether a player is present during slot \( i \)
  - \( r_{ki} = 1 \) iff player \( k \) has arrived and not yet left in slot \( i \)
  - \( r_{ki} = a_{ki} l_{ki} \)
And yet more...

- Whether a player is waiting while a piece is rehearsed
  - $w_{kj} = 1$ iff player $k$ waits while piece $j$ is played
  - $w_{kj} = r_{kdj}$ if $\pi_{kj} = 1$, 0 otherwise
- Total waiting time
  - $t = \sum_{k} \left( \sum_{j} w_{kj} \delta_{j} \right)$
Finally...

• When values have been assigned to $s_1, s_2, \ldots, s_n$ a chain of constraint propagation through the new constraints will assign a value to $t$, as required.

• Although we have a lot of new variables and constraints, we still only have $n$ decision variables.
Variable Ordering

- As soon as enough sequence variables have been assigned so that it is known when a player arrives and leaves, the waiting time for that player will be known.
- But if we choose the variables in the order $s_1, s_2, \ldots, s_n$ this won’t happen until the sequence is nearly complete.
- The search algorithm only says “choose a variable that has not yet been assigned a value” - it doesn’t specify a choice.
- A better order is $s_1, s_n, s_2, s_{n-1}, \ldots$. 
Propagating in the Rehearsal Problem

- Suppose the first 4 assignments are $s_1 = 3$, $s_9 = 9$, $s_2 = 8$, $s_8 = 4$
- Player 1 does not play in pieces 3 and 9, but does play in pieces 4 and 8
- After these assignments, it is deduced that:
  - Player 1 arrives before the 2\textsuperscript{nd} piece & leaves after the 8\textsuperscript{th}
  - Player 1 is only waiting during piece 5 (even though it has not been decided when piece 5 will be played)
  - The waiting time for player 1 is 3 (the duration of piece 5)
Results

- Number of backtracks is a good measure of search effort
- It takes nearly as many backtracks to prove optimality as to find the optimal solution, with first-to-last ordering

<table>
<thead>
<tr>
<th>Search order</th>
<th>Backtracks to find optimal</th>
<th>Total backtracks</th>
<th>Run time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First to last</td>
<td>37,213</td>
<td>65,090</td>
<td>23.9</td>
</tr>
<tr>
<td>Ends to middle</td>
<td>1,170</td>
<td>1,828</td>
<td>1.4</td>
</tr>
</tbody>
</table>
Symmetry

- Reversing the sequence does not change waiting time.
- Search finds an optimal sequence starting with 3 and ending with 9, then considers sequences starting with 9 and ending with 3.
- This is wasted effort.
- Can be prevented by adding a constraint that is only true of one of a pair of mirror-image sequences, e.g. $s_1 < s_n$. 
## Results

<table>
<thead>
<tr>
<th>Search order</th>
<th>Backtracks to find optimal</th>
<th>Total backtracks</th>
<th>Run time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First to last</td>
<td>37,213</td>
<td>65,090</td>
<td>23.9</td>
</tr>
<tr>
<td>Ends to middle</td>
<td>1,170</td>
<td>1,828</td>
<td>1.4</td>
</tr>
<tr>
<td>First to last with $s_1 &lt; s_n$</td>
<td>35,679</td>
<td>48,664</td>
<td>18.4</td>
</tr>
<tr>
<td>Ends to middle with $s_1 &lt; s_n$</td>
<td>1,125</td>
<td>1,365</td>
<td>1.0</td>
</tr>
</tbody>
</table>
A Talent Scheduling Problem

- In shooting a film, any actor not involved in the day’s scenes still gets paid.
- Scheduling problem identical to the rehearsal problem - except that actors are paid at different rates.
- Sample problem (for the film ‘Mob Story’) in Cheng et al. is much larger than the rehearsal problem (8 ‘players’, 20 ‘pieces’).
Improved Model

- The existing model cannot solve the talent scheduling problem in a reasonable time
- Waiting time for a player is only known when the first & last pieces he/she plays in are sequenced
- Constraints don’t allow deductions about the sequence from a tighter constraint on the objective
## Optimal Sequence

<table>
<thead>
<tr>
<th>Player</th>
<th>Sequence</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
<td>1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0</td>
<td>10</td>
</tr>
<tr>
<td>Player 2</td>
<td>0 1 1 0 1 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0</td>
<td>4</td>
</tr>
<tr>
<td>Player 3</td>
<td>0 0 0 0 1 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0</td>
<td>5</td>
</tr>
<tr>
<td>Player 4</td>
<td>0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>5</td>
</tr>
<tr>
<td>Player 5</td>
<td>0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 0 1 0 1</td>
<td>5</td>
</tr>
<tr>
<td>Player 6</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0</td>
<td>40</td>
</tr>
<tr>
<td>Player 7</td>
<td>0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0</td>
<td>4</td>
</tr>
<tr>
<td>Player 8</td>
<td>0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0</td>
<td>20</td>
</tr>
</tbody>
</table>
Implied Constraints

- Implied constraints are logically redundant – don’t change the problem, just state part of it differently
- Good implied constraints reduce solution time by increasing constraint propagation
- The pieces the expensive players play in must be together
- Given a good solution (so a tight bound on the total waiting time) if we have placed one of these pieces in the sequence, the others must be very close to it
- This is not being recognised in the existing model
Constraint on Waiting time

• Waiting time for a player is \textit{at least} the number of slots in the sequence between the time they arrive and the time they leave, less the number of pieces they play in.

• This is a lower bound, because it just uses the fact that the duration of a piece is at least 1 time unit.

• This is apparently a weak constraint, but in fact allows a bound on the waiting time to reduce the domains of the sequence variables.
Results

- Adding these implied constraints improves solution time dramatically
- With other constraints, the talent scheduling problem can be solved:

<table>
<thead>
<tr>
<th></th>
<th>Backtracks</th>
<th>Run time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rehearsal problem</td>
<td>448</td>
<td>0.9</td>
</tr>
<tr>
<td>Talent scheduling</td>
<td>576,579</td>
<td>1,120</td>
</tr>
</tbody>
</table>
Conclusions

- The model for the rehearsal problem is complex – but then describing the connection between the sequence of pieces and the waiting time, in words, is also complex.
- This kind of sequencing problem is NP-hard, so it’s not surprising that solving a much larger problem requires a cleverer model.
- Further improvements are possible – e.g. start with a better initial solution.
- Improving the model needs an understanding of how constraints propagate – but mostly insight into the problem.