Lecture 9: Symmetry Breaking

Jediah Katz jediahk@seas.upenn.edu
Due date reminders

- Project partners due 11/2 on Gradescope!
- Project proposal and HW3 due next week (11/9)
- No quiz for this lecture so you can focus on that
Recall: Coloring Symmetry

- From lecture 2: graph coloring problem (with SAT)
  - How to color vertices of a graph s.t. no two neighbors have the same color?
- Issue: symmetry
  - These two colorings are symmetric but technically different
Recall: Symmetry Breaking

- Can add **symmetry-breaking constraints** that prevent checking multiple symmetric assignments
  - Ex: add constraint \( \text{color}(1) = \text{BLUE} \)
    - The larger the clique of vertices that we fix colors for, the more symmetries are broken
Preliminary Definitions

- Recall that a constraint program contains a set of variables $x_1, \ldots, x_n$, each with a range of values.
- A permutation $\sigma$ of a set $S$ is a bijection (1-to-1 correspondence) from $S$ to $S$.

Ex: $2, 3, 1$
Value Symmetry

- A value symmetry in a constraint program is a feasibility-preserving permutation $\sigma$ of values.
- That is, for any choice of values $v_1, \ldots, v_n$:

  \[
  \{x_1 = v_1; x_2 = v_2; \ldots; x_n = v_n\} \text{ is feasible if and only if }
  \{x_1 = \sigma(v_1); x_2 = \sigma(v_2); \ldots; x_n = \sigma(v_n)\} \text{ is feasible}
  \]
The graph coloring example contains value symmetry:

- Variables: \( c_0, c_1, c_2 \in \{R, B, G\} \)
- Constraints: \( \{c_1 \neq c_0, c_1 \neq c_2\} \)

Example symmetries:

\[
\sigma_1 = \{R \rightarrow B; B \rightarrow R; G \rightarrow G\}
\]
\[
\sigma_2 = \{R \rightarrow B; B \rightarrow G; G \rightarrow R\}
\]

In fact, any permutation is a value symmetry in graph coloring...
A variable symmetry in a constraint program is a feasibility-preserving permutation $\sigma$ of variables.

That is, for any choice of values $v_1, \ldots, v_n$:

$$\{x_1 = v_1; \ x_2 = v_2; \ldots; \ x_n = v_n\}$$

is feasible if and only if

$$\{\sigma(x_1) = v_1; \ \sigma(x_2) = v_2; \ldots; \ \sigma(x_n) = v_n\}$$

is feasible.
Ex: Variable Symmetry

- **Variables:** $x_1, x_2, x_3 \in \{1, 2, 3\}$
- **Constraints:** $\{x_1 = x_2 + x_3\}$
- **Feasible solns:** $\{(2,1,1), (3,1,2), (3,2,1)\}$

- **Variable symmetry:** $\{x_1 \rightarrow x_1; x_2 \rightarrow x_3; x_3 \rightarrow x_2\}$
  - Intuition: addition is commutative

- No other (nontrivial) variable symmetries
Variable-Value Symmetry

- It's possible to have both variable symmetry $\sigma_1$ and value symmetry $\sigma_2$ concurrently.
- That is, for any choice of values $v_1, ..., v_n$:

$$\{x_1 = v_1; \; x_2 = v_2; \; ...; \; x_n = v_n\} \text{ is feasible}$$

if and only if

$$\{\sigma_1(x_1) = \sigma_2(v_1); \; \sigma_1(x_2) = \sigma_2(v_2); \; ...; \; \sigma_1(x_n) = \sigma_2(v_n)\} \text{ is feasible}$$
Ex: Variable-Value Symmetry

- In the **sports tournament problem**:
  - $n$ teams play over an $(n - 1)$-week season
  - $n/2$ timeslots for 1-on-1 matches per week
  - each team plays each other team exactly once
  - each team plays exactly once per week
  - each team plays at most twice per timeslot

<table>
<thead>
<tr>
<th>Slot 1</th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T1 vs T2</td>
<td>T1 vs T3</td>
<td>T3 vs T6</td>
<td>T2 vs T6</td>
<td>T4 vs T5</td>
</tr>
<tr>
<td>Slot 2</td>
<td>T3 vs T4</td>
<td>T2 vs T5</td>
<td>T2 vs T4</td>
<td>T3 vs T5</td>
<td>T1 vs T6</td>
</tr>
<tr>
<td>Slot 3</td>
<td>T5 vs T6</td>
<td>T4 vs T6</td>
<td>T1 vs T5</td>
<td>T1 vs T4</td>
<td>T2 vs T3</td>
</tr>
</tbody>
</table>
Ex: Variable-Value Symmetry

- Suppose we use variables: $x_{swp} \in \{T_1, T_2, \ldots, T_n\}$
  - $x_{swp} = \text{team playing in slot } s \text{ of week } w \text{ in position } p \in \{0, 1\}$

- Symmetries in sports tournament problem:
  - Timeslots can be permuted: $(n/2)!$ variable symmetries
  - Weeks can be permuted: $(n - 1)!$ variable symmetries
  - Positions for each game """: $2!^{(n-1)(n/2)}$ variable symmetries
  - Team names """: $n!$ value symmetries

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Slot 1</strong></td>
<td>T1 vs T2</td>
<td>T1 vs T3</td>
<td>T3 vs T6</td>
<td>T2 vs T6</td>
</tr>
<tr>
<td><strong>Slot 2</strong></td>
<td>T3 vs T4</td>
<td>T2 vs T5</td>
<td>T2 vs T4</td>
<td>T3 vs T5</td>
</tr>
<tr>
<td><strong>Slot 3</strong></td>
<td>T5 vs T6</td>
<td>T4 vs T6</td>
<td>T1 vs T5</td>
<td>T1 vs T4</td>
</tr>
</tbody>
</table>
Symmetry Breaking

- How can we spend less effort examining symmetric solutions in our search space?
- Known as symmetry breaking – maybe better to think of it as exploiting symmetry
Reformulation

● Sometimes, symmetry is a product of the model and not the underlying problem itself
● Can **reformulate** our model to remove symmetries
● Some constraint solvers (not OR-Tools) provide special constructs for flexible model formulations
  ○ E.g. **set variables**
Ex: Reformulation

- For sports tournament problem:
- Represent each game between a pair of teams as a number from 1 to \(\binom{n}{2}\) = total number of pairs
- Can use variables \(x_{sw} \in \{G_1, G_2, \ldots, G_{\binom{n}{2}}\}\)
  - \(x_{sw}\) = game played in slot \(s\) of week \(w\)
- Removes ordering of teams for each game
- Other symmetries can be broken with more effort
Downsides of Reformulation

- May be difficult to express constraints in reformulated model
- Often complex and ad-hoc for different problems
- Sometimes symmetry is inherent in the problem, not just the model
Symmetry Breaking Constraints

- **Idea:** add new constraints to our model to exploit problem/model symmetry
- Comparatively simple: don't need to change the old model, just add to it
Ex: Sym. Breaking Constraints

- For sports tournament problem:
- Using original variables: $x_{swp} \in \{T_1, T_2, \ldots, T_n\}$
  - $x_{swp} =$ team playing in slot $s$ of week $w$ in position $p \in \{0, 1\}$
- How to remove ordering of teams for each game?
- Simple: fix an ordering for the two teams
- That is, add constraint: $x_{sw1} < x_{sw2}$
Generalizing Order Constraints

- Easy to enforce order between teams in a game
- But how do we generalize this to break the other variable symmetries (weeks, slots)?
- For simplicity, think of variables as 2D matrix $x_{sw}$ by reformulation

<table>
<thead>
<tr>
<th></th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$w = 3$</th>
<th>$w = 4$</th>
<th>$w = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 1$</td>
<td>$G_{11}$</td>
<td>$G_{12}$</td>
<td>$G_{13}$</td>
<td>$G_{14}$</td>
<td>$G_{15}$</td>
</tr>
<tr>
<td>$s = 2$</td>
<td>$G_{21}$</td>
<td>$G_{22}$</td>
<td>$G_{23}$</td>
<td>$G_{24}$</td>
<td>$G_{25}$</td>
</tr>
<tr>
<td>$s = 3$</td>
<td>$G_{31}$</td>
<td>$G_{32}$</td>
<td>$G_{33}$</td>
<td>$G_{34}$</td>
<td>$G_{35}$</td>
</tr>
</tbody>
</table>
**Index Symmetry**

- Weeks and slots have **index symmetry**: variable symmetry from permuting rows/columns of matrix.
- **Idea**: use same trick to break index symmetries by fixing an ordering over rows and over columns.

<table>
<thead>
<tr>
<th></th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$w = 3$</th>
<th>$w = 4$</th>
<th>$w = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 1$</td>
<td>$G_{11}$</td>
<td>$G_{12}$</td>
<td>$G_{13}$</td>
<td>$G_{14}$</td>
<td>$G_{15}$</td>
</tr>
<tr>
<td>$s = 2$</td>
<td>$G_{21}$</td>
<td>$G_{22}$</td>
<td>$G_{23}$</td>
<td>$G_{24}$</td>
<td>$G_{25}$</td>
</tr>
<tr>
<td>$s = 3$</td>
<td>$G_{31}$</td>
<td>$G_{32}$</td>
<td>$G_{33}$</td>
<td>$G_{34}$</td>
<td>$G_{35}$</td>
</tr>
</tbody>
</table>
Lexicographic Ordering

- **Lexicographic ordering**: way of ordering vectors
  - Intuition: compare first coordinate, and in a tie try next one
  - See additional code (sports.py) for my implementation

- We say \([x_1, x_2, x_3, \ldots] <_{\text{lex}} [y_1, y_2, y_3, \ldots]\) iff:
  - \(x_1 < y_1\), or
  - \(x_1 = y_1\), and \([x_2, x_3, \ldots] <_{\text{lex}} [y_2, y_3, \ldots]\)

- **Ex:** \([c, o, n, s, t, r, a, i, n, t] <_{\text{lex}} [c, o, n, s, t, r, u, c, t, s]\)
  - Because \(a < u\)

- We say \([x_i] \leq_{\text{lex}} [y_i]\) iff \([x_i] <_{\text{lex}} [y_i]\) or \([x_i] = [y_i]\)
The DoubleLex Scheme

- Therefore, to break row and column symmetries:
  - Rows: \( G_{1w} \leq_{\text{lex}} G_{2w} \leq_{\text{lex}} G_{3w} \)
  - Cols: \( G_{s1} \leq_{\text{lex}} G_{s2} \leq_{\text{lex}} G_{s3} \leq_{\text{lex}} G_{s4} \leq_{\text{lex}} G_{s5} \)
  - Theorem: since matrix entries are all different, lexicographical ordering all dimensions breaks all symmetries!

<table>
<thead>
<tr>
<th>( w = 1 )</th>
<th>( w = 2 )</th>
<th>( w = 3 )</th>
<th>( w = 4 )</th>
<th>( w = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s = 1 )</td>
<td>( G_{11} )</td>
<td>( G_{12} )</td>
<td>( G_{13} )</td>
<td>( G_{14} )</td>
</tr>
<tr>
<td>( s = 2 )</td>
<td>( G_{21} )</td>
<td>( G_{22} )</td>
<td>( G_{23} )</td>
<td>( G_{24} )</td>
</tr>
<tr>
<td>( s = 3 )</td>
<td>( G_{31} )</td>
<td>( G_{32} )</td>
<td>( G_{33} )</td>
<td>( G_{34} )</td>
</tr>
</tbody>
</table>
The DoubleLex Scheme

● If matrix entries are not all different, then in general DoubleLex may not break all symmetries
  ○ In practice, though, it still breaks many!

● Counterexample:
  ○ These matrices are symmetric (swap $C_1, C_2$ and swap $R_1, R_3$)...
  ○ But they are both lex ordered!

\[
\begin{array}{ccc}
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 0 & 1 \\
\end{array}
\begin{array}{ccc}
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 0 & 1 \\
\end{array}
\]
Simplifying Lex Constraints

- Most lex constraints don't really need to include all the variables – they can be simplified
- Ex: \([x_1, x_2, x_3] \leq_{\text{lex}} [x_3, x_2, x_1]\)
  \([x_1, x_3] \leq_{\text{lex}} [x_3, x_1]\)

- O1: if \(\sigma(x_i) = x_i\), then that pair of variables always have the same value, so they can be removed
Most lex constraints don't really need to include all the variables – they can be simplified.

Ex: \[ [x_1, x_3] \leq_{\text{lex}} [x_3, x_1] \]
\[ [x_1] \leq_{\text{lex}} [x_3] \]

O2: if \( \sigma(x_i) = x_{i+1} \) and \( \sigma(x_{i+1}) = x_i \), then the second pair is only considered if the first pair has the same value – but then so does the second pair.
   • So the second pair can be removed.
Generalizing to K-D Arrays

- \( x_{swp} \) is 3-D array of vars
- How to generalize to higher dimensional matrices?
- Instead of rows and columns, we have slices
- Just “flatten” slices into a vector and order them lexicographically

\[
\begin{array}{ccccc}
\text{\( w = 1 \)} & \text{\( w = 2 \)} & \text{\( w = 3 \)} & \text{\( w = 4 \)} & \text{\( w = 5 \)} \\
\hline
\text{\( s = 1 \)} & T1 & T1 & T3 & T2 & T4 \\
\text{\( s = 2 \)} & T3 & T2 & T2 & T3 & T1 \\
\text{\( s = 3 \)} & T5 & T4 & T1 & T1 & T2 \\
\end{array}
\]

\[
\begin{array}{ccccc}
\text{\( w = 1 \)} & \text{\( w = 2 \)} & \text{\( w = 3 \)} & \text{\( w = 4 \)} & \text{\( w = 5 \)} \\
\hline
\text{\( s = 1 \)} & T2 & T3 & T6 & T6 & T5 \\
\text{\( s = 2 \)} & T4 & T5 & T4 & T5 & T6 \\
\text{\( s = 3 \)} & T6 & T6 & T5 & T4 & T3 \\
\end{array}
\]
Generalizing Further

- Not all variable symmetries are index symmetries
- Can we generalize our symmetry breaking scheme to arbitrary variable symmetries?
  - ...and is this a good idea?
The Lex-Leader Scheme

- Method of breaking **all** variable symmetries
- For each variable symmetry $\sigma$, add the constraint:
  $$[x_1, x_2, ..., x_n] \leq_{\text{lex}} [\sigma(x_1), \sigma(x_2), ..., \sigma(x_n)]$$
- **Intuition:** for each set of solutions which can be obtained by permuting by a symmetry, prefer the lexicographically least of these
- **Issue:** might be exponentially many symmetries!
  - DoubleLex is less prohibitive – adds linear num. of constraints
Ex: The Lex-Leader Scheme

- **Variables:** $x_1, x_2, x_3 \in \{1, 2, 3\}$
- **Constraints:** \(x_1 + x_2 + x_3 = 5\)
- **Symmetries:** all 6 permutations of $x_1, x_2, x_3$
- **Lex-Leader constraints:**

\[
\begin{align*}
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_1, x_3, x_2] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_2, x_1, x_3] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_2, x_3, x_1] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_3, x_1, x_2] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_3, x_2, x_1] \\
\end{align*}
\]

- Simplify

\[
\begin{align*}
[x_2] & \leq_{\text{lex}} [x_3] \\
[x_1] & \leq_{\text{lex}} [x_2] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_2, x_3, x_1] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_3, x_1, x_2] \\
[x_1, x_2, x_3] & \leq_{\text{lex}} [x_3, x_2, x_1] \\
\end{align*}
\]

- Highly redundant – all we really needed was $x_1 \leq x_2 \leq x_3$!
Symmetry Breaking Caveats

- Many symmetry breaking schemes have been studied using traditional CP solvers (vs. CP-SAT)
- Even DoubleLex will sometimes give more overhead than speedup, esp. for CP-SAT
- Symmetry breaking is often (anecdotally) more effective on infeasible instances
  - Also very helpful when enumerating many solutions
- Haven’t really discussed value symmetry – if interested, read about simple generalization of LexLeader [here](#)
Symmetry Breaking Advice

- Symmetry breaking is (like much of this) sometimes more art than science
- Look for easy, problem-specific symmetries that can be broken with few/basic constraints (e.g. coloring)
- DoubleLex can be hit-or-miss, but if you write it once it's pretty easy to “copy and paste” to another problem
References

