Consistency as Projection

John Hooker
Carnegie Mellon University

INFORMS 2015, Philadelphia USA
Consistency as Projection

- Reconceive **consistency** in constraint programming as a form of **projection**.
  - For example, domain consistency, \(k\)-consistency, bounds consistency.
Consistency as Projection

- Define a **stronger** form of consistency, based on projection, that propagates through relaxed decision diagrams.
  - …rather than through a **domain store**.
  - **J-consistency** is achieved by **projection** onto a set $J$ of variables.
Consistency as Projection

• Find $J$-consistency algorithms for some popular global constraints.
  
  – Among, sequence, regular, SAT, alldiff.
The Big Picture

• Decision diagrams and optimization

• Relaxed decision diagrams
  – For constraint propagation, bounding, and discrete optimization

• Propagating $J$-consistency through relaxed decision diagrams

• Achieving $J$-consistency
Decision Diagrams

- Graphical encoding of a function of Boolean (or multivalued) variables
  - Historically used for circuit design & verification
    - Lee (1959), Bryant (1986)
  - Adapt to optimization and constraint programming (CP)
Example:
Stable set problem

Find max-weight subset of nonadjacent vertices
Decision Diagrams

Example:
Stable set problem

Find max-weight subset of nonadjacent vertices

Each path corresponds to a feasible solution.
Example: Stable set problem

Find max-weight subset of nonadjacent vertices

Longest path (35) corresponds to an optimal solution.
We can optimize by finding a longest path in the decision diagram, but…

- The decision diagram can grow exponentially with the problem size.
Relaxed Decision Diagrams

- Key idea: Use relaxed decision diagrams
  - Represent a superset of the feasible set
    - … with a limited-width diagram.
  - First used to strengthen propagation in CP solvers.
    - Reduced 1-million-node search trees to 1 node for graph coloring (multiple alldiff) problem

Andersen, Hadžić, JH, Tiedemann (2007)
Relaxed Decision Diagrams

Exact diagram
Width = 3
9 solutions

Relaxed diagram
Width = 2
11 solutions
• Standard CP solvers use **domain propagation**.

• **Domain consistency** = each value in a variable’s domain is part some feasible solution.
Example with 2 constraints on 0-1 variables:

\[ 2x_1 + x_2 + x_3 \leq 1 \]

nonadjacency constraint in stable set problem

Reduce domains to:

- \( x_1 \in \{0\} \)
- \( x_2 \in \{0,1\} \)
- \( x_3 \in \{0,1\} \)

Longest path over reduced domains is 25, which is optimal for both constraints.
Propagation

Example with 2 constraints on 0-1 variables:

\[2x_1 + x_2 + x_3 \leq 1\]

nonadjacency constraint in stable set problem

Reduce domains to:

- \(x_1 \in \{0\}\)
- \(x_2 \in \{0,1\}\)
- \(x_3 \in \{0,1\}\)

Longest path over reduced domains is 25, which is optimal for both constraints.

However, domain propagation can be weak…
Propagation

Another example:

\[ x_1 \leq x_2 \]

nonadjacency constraint in stable set problem

Constraints have no effect on domains.

No information transmitted from one constraint to the other
Another example:

$x_1 \leq x_2$

nonadjacency constraint in stable set problem

Constraints have no effect on domains.
No information transmitted from one constraint to the other

We can improve propagation by propagating through a relaxed decision diagram.
Relaxed Decision Diagrams

Relaxed diagram for nonadjacency constraint

Diagram after propagating $x_1 \leq x_2$

Information is now transmitted between constraints.
Relaxed Decision Diagrams

• Relaxed decision diagrams also provide **optimization bounds**.
  
  – Longest path in relaxed diagram provides a **bound** on optimal value.

  – Discrete relaxation replaces LP relaxation.

Bergman, Ciré, van Hoeve, JH (2014)
Relaxed Decision Diagrams

Exact diagram
Longest path = 35

Relaxed diagram
Longest path = 35

Bound is sharp in this case
Relaxed Decision Diagrams

- Wider diagrams yield tighter bounds
  - But take longer to build.
  - Adjust width dynamically.
Discrete Optimization

• Relaxed decision diagrams provide a general method for **discrete optimization**
  
  – Solve a **recursive** (dynamic programming) model using a **branch-and-bound** algorithm.
  
  – Branch on nodes in **last exact layer** of relaxed decision diagram.
    – …rather than branching on variables.
    – New approach to **curse of dimensionality**.

Bergman, Ciré, van Hoeve, JH (2015)
Optimization

Branching in a relaxed decision diagram

Diagram is exact down to here

Relaxed decision diagram
Optimization

Branching in a relaxed decision diagram

Branch on nodes in this layer

Relaxed decision diagram
Branching in a relaxed decision diagram

First branch

Relaxed decision diagram

Optimization
Optimization

Branching in a relaxed decision diagram

Second branch

Relaxed decision diagram
Optimization

Branching in a relaxed decision diagram

Third branch

Continue recursively

Relaxed decision diagram
Max cut on a graph

Avg. solution time vs graph density

30 vertices
Optimization

Max 2-SAT

Performance profile

30 variables
Optimization

Max 2-SAT

Performance profile

40 variables

![Graph showing the performance profile of Max 2-SAT with 40 variables solved over computation time in seconds. The graph compares MDDs and CPLEX solutions.](image_url)
Consistency

- Next step:
  - Relaxed decision diagrams transmit more than domain information.
  - So let’s define a stronger form of consistency than domain consistency.
    - And propagate the results of achieving it.
Consistency

• Key observation:

  – Existing types of consistency are forms of projection.

  – So define a stronger form of consistency based on projection.
Consistency as Projection

• Domain consistency
  – Project onto each individual variable $x_j$. 
Consistency as Projection

• Domain consistency
  – Project onto each individual variable $x_j$.

Example:

Constraint set

\[ \text{alldiff} \left( x_1, x_2, x_3 \right) \]

\[ x_1 \in \{ a, b \} \]
\[ x_2 \in \{ a, b \} \]
\[ x_3 \in \{ b, c \} \]
Consistency as Projection

- **Domain consistency**
  - Project onto each individual variable $x_j$.

**Example:**

<table>
<thead>
<tr>
<th>Constraint set</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>alldiff(x_1, x_2, x_3)</code></td>
<td>$(x_1, x_2, x_3)$</td>
</tr>
<tr>
<td>$x_1 \in {a, b}$</td>
<td>$(a, b, c)$</td>
</tr>
<tr>
<td>$x_2 \in {a, b}$</td>
<td>$(b, a, c)$</td>
</tr>
<tr>
<td>$x_3 \in {b, c}$</td>
<td></td>
</tr>
</tbody>
</table>
Consistency as Projection

• Domain consistency
  – Project onto each individual variable $x_j$.

Example:

<table>
<thead>
<tr>
<th>Constraint set</th>
<th>Solutions</th>
<th>Projection onto $x_1$</th>
<th>Projection onto $x_2$</th>
<th>Projection onto $x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{alldiff}(x_1, x_2, x_3)$</td>
<td>$(x_1, x_2, x_3)$</td>
<td>$x_1 \in {a, b}$</td>
<td>$x_2 \in {a, b}$</td>
<td>$x_3 \in {c}$</td>
</tr>
</tbody>
</table>
Consistency as Projection

- **Domain consistency**
  - Project onto each individual variable $x_j$.

**Example:**

<table>
<thead>
<tr>
<th>Constraint set</th>
<th>Solutions</th>
<th>Projection onto $x_1$</th>
<th>Projection onto $x_2$</th>
<th>Projection onto $x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{alldiff}(x_1, x_2, x_3)$</td>
<td>$(x_1, x_2, x_3)$</td>
<td>$x_1 \in {a, b}$</td>
<td>$x_2 \in {a, b}$</td>
<td>$x_3 \in {c}$</td>
</tr>
<tr>
<td>$x_1 \in {a, b}$</td>
<td>$(a, b, c)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2 \in {a, b}$</td>
<td>$(b, a, c)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3 \in {b, c}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This achieves domain consistency.
Consistency as Projection

• **Domain consistency**
  – Project onto each individual variable $x_j$.

**Example:**

<table>
<thead>
<tr>
<th>Constraint set</th>
<th>Solutions</th>
<th>Projection onto $x_1$</th>
<th>Projection onto $x_2$</th>
<th>Projection onto $x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{alldiff}(x_1, x_2, x_3)$</td>
<td>$(x_1, x_2, x_3)$</td>
<td>$x_1 \in {a, b}$</td>
<td>$x_2 \in {a, b}$</td>
<td>$x_3 \in {c}$</td>
</tr>
<tr>
<td>$x_1 \in {a, b}$</td>
<td>$(a, b, c)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2 \in {a, b}$</td>
<td>$(b, a, c)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3 \in {b, c}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This achieves domain consistency.

We will regard a projection as a **constraint set**.
Consistency as Projection

• $k$-consistency

  – Can be defined:

  – A constraint set $S$ is $k$-consistent if:
    • for every $J \subseteq \{1, \ldots, n\}$ with $|J| = k - 1$,
    • every assignment $x_J = v_J \in D_J$ for which $(x_J, x_j)$ does not violate $S$,
    • and every variable $x_j \not\in x_J$, there is an assignment $x_j = v_j \in D_j$ for which $(x_J, x_j) = (v_J, v_j)$ does not violate $S$. 

$s_J = (x_j \mid j \in J)$
Consistency as Projection

• $k$-consistency

- Can be defined:
  - A constraint set $S$ is $k$-consistent if:
    - for every $J \subseteq \{1, \ldots, n\}$ with $|J| = k - 1$,
    - every assignment $x_J = v_J \in D_j$ for which $(x_J, x_j)$ does not violate $S$,
    - and every variable $x_j \notin x_J$,
  there is an assignment $x_j = v_j \in D_j$ for which $(x_J, x_j) = (v_J, v_j)$
  does not violate $S$.

- To achieve $k$-consistency:
  - Project the constraints containing each set of $k$ variables
    onto subsets of $k - 1$ variables.
Consistency as Projection

- Consistency and backtracking:
  - Strong $k$-consistency for entire constraint set avoids backtracking…
    - if the primal graph has width $< k$ with respect to branching order.
  
  Freuder (1982)

- But strong $k$-consistency is very hard to achieve.
**J-Consistency**

- A type of consistency more directly related to projection.
  - Constraint set $S$ is **J-consistent** if it contains the **projection** of $S$ onto $x_J = (x_j | j \in J)$
    - $S$ is domain consistent if it is $\{ j \}$-consistent for each $j$.

JH (2015)
J-Consistency

• *J*-consistency and backtracking:

  – If we branch on variables $x_1, x_2, \ldots$, a natural strategy is to project out $x_n, x_{n-1}, \ldots$

    – until computational burden is excessive.
Propagating $J$-Consistency

Example:

among((x₁, x₂), {c, d}, 1, 2)
(x₁ = c) ⇒ (x₂ = d')
alldiff (x₁, x₂, x₃, x₄)
x₁, x₂ ∈ {a, b, c, d}
x₃ ∈ {a, b}
x₄ ∈ {c, d'}

Already domain consistent for individual constraints.

If we branch on $x_1$ first, must consider all 4 branches $x_1 = a, b, c, d$
Propagating $J$-Consistency

Example:

\[
\text{among}\((x_1, x_2), \{c, d\}, 1, 2) \\
(x_1 = c) \Rightarrow (x_2 = d) \\
\text{alldiff}(x_1, x_2, x_3, x_4) \\
x_1, x_2 \in \{a, b, c, d\} \\
x_3 \in \{a, b\} \\
x_4 \in \{c, d\}
\]

Suppose we propagate through a relaxed decision diagram of width 2 for these constraints.

52 paths from top to bottom represent assignments to $x_1, x_2, x_3, x_4$.
36 of these are the feasible assignments.
Propagating J-Consistency

Example:

among((x_1, x_2), {c, d}, 1, 2) (x_1 = c) ⇒ (x_2 = d)
alldiff(x_1, x_2, x_3, x_4)
x_1, x_2 ∈ \{a, b, c, d\}
x_3 ∈ \{a, b\}
x_4 ∈ \{c, d\}

Suppose we propagate through a relaxed decision diagram of width 2 for these constraints

Projection of alldiff onto x_1, x_2 is

alldiff(x_1, x_2)
atmost((x_1, x_2), \{a, b\}, 1)
atmost((x_1, x_2), \{c, d\}, 1)

52 paths from top to bottom represent assignments to x_1, x_2, x_3, x_4.
36 of these are the feasible assignments.
Propagating J-Consistency

Let’s propagate the 2\textsuperscript{nd} atmost constraint in the projected alldiff through the relaxed decision diagram.

Let the length of a path be number of arcs with labels in \{c,d\}.

For each arc, indicate length of shortest path from top to that arc.

Projection of alldiff onto \(x_1, x_2\) is

\[
\text{alldiff}(x_1, x_2) \\
\text{atmost}((x_1, x_2), \{a, b\}, 1) \\
\text{atmost}((x_1, x_2), \{c, d\}, 1)
\]
Propagating $J$-Consistency

Let’s propagate the 2\textsuperscript{nd} atmost constraint in the projected alldiff through the relaxed decision diagram.

Let the length of a path be number of arcs with labels in $\{c,d\}$.

For each arc, indicate length of shortest path from top to that arc.

Projection of alldiff onto $x_1, x_2$ is

\begin{align*}
\text{alldiff} \left( x_1, x_2 \right) \\
\text{atmost} \left( (x_1, x_2), \{a, b\}, 1 \right) \\
\text{atmost} \left( (x_1, x_2), \{c, d\}, 1 \right)
\end{align*}
Let’s propagate the 2\textsuperscript{nd} atmost constraint in the projected alldiff through the relaxed decision diagram.

Let the length of a path be number of arcs with labels in \{c,d\}.

For each arc, indicate length of shortest path from top to that arc.

Remove arcs with label > 1

Projection of alldiff onto \(x_1, x_2\) is

\[
\text{alldiff}(x_1, x_2) \\
\text{atmost}((x_1, x_2), \{a, b\}, 1) \\
\text{atmost}((x_1, x_2), \{c, d\}, 1)
\]
Propagating $J$-Consistency

Let’s propagate the $2^{nd}$ atmost constraint in the projected alldiff through the relaxed decision diagram.

Let the length of a path be number of arcs with labels in \{c,d\}.

For each arc, indicate length of shortest path from top to that arc.

Remove arcs with label $> 1$

Projection of alldiff onto $x_1, x_2$ is

\[
\text{alldiff}(x_1, x_2) = \text{atmost}((x_1, x_2), \{a, b\}, 1) \land \text{atmost}((x_1, x_2), \{c, d\}, 1)
\]
Let’s propagate the 2\textsuperscript{nd} atmost constraint in the projected alldiff through the relaxed decision diagram.

Let the length of a path be number of arcs with labels in \{c,d\}.

For each arc, indicate length of shortest path from top to that arc.

Remove arcs with label > 1

Clean up.

Projection of alldiff onto \(x_1, x_2\) is

\[
\text{alldiff} \left( x_1, x_2 \right) \atmost \left( (x_1, x_2), \{a, b\}, 1 \right) \atmost \left( (x_1, x_2), \{c, d\}, 1 \right)
\]
Let’s propagate the 2\textsuperscript{nd} atmost constraint in the projected alldiff through the relaxed decision diagram.

Let the length of a path be number of arcs with labels in \{c,d\}.

For each arc, indicate length of shortest path from top to that arc.

Remove arcs with label > 1

Clean up.

Projection of alldiff onto \(x_1, x_2\) is

\[
\text{alldiff} \left( x_1, x_2 \right) \\
\text{atmost} \left( (x_1, x_2), \{a, b\}, 1 \right) \\
\text{atmost} \left( (x_1, x_2), \{c, d\}, 1 \right)
\]
Let’s propagate the 2nd atmost constraint in the projected alldiff through the relaxed decision diagram.

We need only branch on \( a, b, d \) rather than \( a, b, c, d \)

Remove arcs with label > 1

Clean up.

Projection of alldiff onto \( x_1, x_2 \) is

\[
\text{alldiff} \left( x_1, x_2 \right) \\
\text{atmost} \left( \left( x_1, x_2 \right), \{a, b\}, 1 \right) \\
\text{atmost} \left( \left( x_1, x_2 \right), \{c, d\}, 1 \right)
\]
## Achieving $J$-consistency

<table>
<thead>
<tr>
<th>Constraint</th>
<th>How hard to project?</th>
</tr>
</thead>
<tbody>
<tr>
<td>among</td>
<td>Easy and fast.</td>
</tr>
<tr>
<td>sequence</td>
<td>More complicated but fast.</td>
</tr>
<tr>
<td>regular</td>
<td>Easy and basically same labor as domain consistency.</td>
</tr>
<tr>
<td>SAT</td>
<td>Efficient if conflict clauses are used as logic-based Benders cuts.</td>
</tr>
<tr>
<td>alldiff</td>
<td>Quite complicated but practical for small domains.</td>
</tr>
</tbody>
</table>
**J-consistency for Among**

Projection of $\text{among}((x_1, \ldots, x_n), V, t, u)$ onto $x_1, \ldots, x_{n-1}$ is among($(x_1, \ldots, x_{n-1}), V, t', u'$)

where

$$(t', u') = \begin{cases} 
((t - 1)^+, u - 1) & \text{if } D_n \subseteq V \\
(t, \min\{u, n - 1\}) & \text{if } D_n \cap V = \emptyset \\
((t - 1)^+, \min\{u, n - 1\}) & \text{otherwise}
\end{cases}$$
**J-consistency for Among**

Projection of \( \text{among}((x_1,\ldots,x_n),V,t,u) \) onto \( x_1,\ldots,x_{n-1} \) is
\[
\text{among}((x_1,\ldots,x_{n-1}),V,t',u')
\]

where
\[
(t',u') = \begin{cases} 
((t-1)^+, u-1) & \text{if } D_n \subseteq V \\
(t, \min\{u, n-1\}) & \text{if } D_n \cap V = \emptyset \\
((t-1)^+, \min\{u, n-1\}) & \text{otherwise}
\end{cases}
\]

**Example**

\[
\text{among}((x_1, x_2, x_3, x_4, x_5), \{c, d\}, t, u)
\]

\[
D_1 = \{a, b\} \\
D_2 = \{a, b, c\} \\
D_3 = \{a, d\} \\
D_4 = \{c, d\} \\
D_5 = \{d\}
\]
J-consistency for Among

Projection of $\text{among}((x_1,\ldots,x_n), V, t, u)$ onto $x_1,\ldots,x_{n-1}$ is $\text{among}((x_1,\ldots,x_{n-1}), V, t', u')$

where

$$(t', u') = \begin{cases} 
 ((t-1)^+, u-1) & \text{if } D_n \subseteq V \\
 (t, \min\{u, n-1\}) & \text{if } D_n \cap V = \emptyset \\
 ((t-1)^+, \min\{u, n-1\}) & \text{otherwise}
\end{cases}$$

Example

$D_1 = \{a, b\}$
$D_2 = \{a, b, c\}$
$D_3 = \{a, d\}$
$D_4 = \{c, d\}$
$D_5 = \{d\}$

among$((x_1, x_2, x_3, x_4, x_5), \{c, d\}, t, u)$
among$((x_1, x_2, x_3, x_4), \{c, d\}, (t-1)^+, u-1)$
**J-consistency for Among**

Projection of \( \text{among} ( (x_1, \ldots, x_n), V, t, u) \) onto \( x_1, \ldots, x_{n-1} \) is

\[
\text{among} ( (x_1, \ldots, x_{n-1}), V, t', u')
\]

where

\[
(t', u') = \begin{cases} 
((t - 1)^+, u - 1) & \text{if } D_n \subseteq V \\
(t, \min\{u, n - 1\}) & \text{if } D_n \cap V = \emptyset \\
((t - 1)^+, \min\{u, n - 1\}) & \text{otherwise}
\end{cases}
\]

**Example**

\[
D_1 = \{a, b\} \quad \text{among} ( (x_1, x_2, x_3, x_4, x_5), \{c, d\}, t, u)
\]

\[
D_2 = \{a, b, c\} \quad \text{among} ( (x_1, x_2, x_3, x_4), \{c, d\}, (t - 1)^+, u - 1)
\]

\[
D_3 = \{a, d\} \quad \text{among} ( (x_1, x_2, x_3), \{c, d\}, (t - 2)^+, u - 2)
\]

\[
D_4 = \{c, d\}
\]

\[
D_5 = \{d\}
\]
**J-consistency for Among**

Projection of $\text{among}((x_1,\ldots,x_n),V,t,u)$ onto $x_1,\ldots,x_{n-1}$ is $\text{among}((x_1,\ldots,x_{n-1}),V,t',u')$

where

$$ (t',u') = \begin{cases} 
((t-1)^+,u-1) & \text{if } D_n \subseteq V \\
(t,\min\{u,n-1\}) & \text{if } D_n \cap V = \emptyset \\
((t-1)^+\min\{u,n-1\}) & \text{otherwise}
\end{cases} $$

**Example**

$D_1 = \{a,b\}$

$D_2 = \{a,b,c\}$

$D_3 = \{a,d\}$

$D_4 = \{c,d\}$

$D_5 = \{d\}$

$\text{among}((x_1,x_2,x_3,x_4,x_5),\{c,d\},t,u)$

$\text{among}((x_1,x_2,x_3,x_4),\{c,d\},(t-1)^+,u-1)$

$\text{among}((x_1,x_2,x_3),\{c,d\},(t-2)^+,u-2)$

$\text{among}((x_1,x_2),\{c,d\},(t-3)^+\min\{u-2,2\})$
**J-consistency for Among**

Projection of \( \text{among}((x_1, \ldots, x_n), V, t, u) \) onto \( x_1, \ldots, x_{n-1} \) is \( \text{among}((x_1, \ldots, x_{n-1}), V, t', u') \)

where

\[
(t', u') = \begin{cases} 
((t - 1)^+, u - 1) & \text{if } D_n \subseteq V \\
(t, \min\{u, n - 1\}) & \text{if } D_n \cap V = \emptyset \\
((t - 1)^+, \min\{u, n - 1\}) & \text{otherwise}
\end{cases}
\]

Example

\[
D_1 = \{a, b\} \quad \text{among}((x_1, x_2, x_3, x_4, x_5), \{c, d\}, t, u) \\
D_2 = \{a, b, c\} \quad \text{among}((x_1, x_2, x_3, x_4), \{c, d\}, (t - 1)^+, u - 1) \\
D_3 = \{a, d\} \quad \text{among}((x_1, x_2, x_3), \{c, d\}, (t - 2)^+, u - 2) \\
D_4 = \{c, d\} \quad \text{among}((x_1, x_2), \{c, d\}, (t - 3)^+, \min\{u - 2, 2\}) \\
D_5 = \{d\} \quad \text{among}((x_1), \{c, d\}, (t - 4)^+, \min\{u - 2, 1\})
\]
**J-consistency for Among**

Projection of $\text{among}((x_1, \ldots, x_n), V, t, u)$ onto $x_1, \ldots, x_{n-1}$ is $\text{among}((x_1, \ldots, x_{n-1}), V, t', u')$

where

$$(t', u') = \begin{cases} 
((t-1)^+, u-1) & \text{if } D_n \subseteq V \\
(t, \min\{u, n-1\}) & \text{if } D_n \cap V = \emptyset \\
((t-1)^+, \min\{u, n-1\}) & \text{otherwise}
\end{cases}$$

**Example**

$D_1 = \{a, b\}$  \quad \text{among}((x_1, x_2, x_3, x_4, x_5), \{c, d\}, t, u)$
$D_2 = \{a, b, c\}$  \quad \text{among}((x_1, x_2, x_3, x_4), \{c, d\}, (t-1)^+, u-1)$
$D_3 = \{a, d\}$  \quad \text{among}((x_1, x_2, x_3), \{c, d\}, (t-2)^+, u-2)$
$D_4 = \{c, d\}$  \quad \text{among}((x_1, x_2), \{c, d\}, (t-3)^+, \min\{u-2, 2\})$
$D_5 = \{d\}$  \quad \text{among}((x_1), \{c, d\}, (t-4)^+, \min\{u-2, 1\})$
$D_5 = \{d\}$  \quad \text{among}((\), \{c, d\}, (t-4)^+, \min\{u-2, 0\})$
**J-consistency for Among**

Projection of \( \text{among}((x_1, \ldots, x_n), V, t, u) \) onto \( x_1, \ldots, x_{n-1} \) is
\[
\text{among}((x_1, \ldots, x_{n-1}), V, t', u')
\]
where
\[
(t', u') = \begin{cases} 
((t-1)^+, u-1) & \text{if } D_n \subseteq V \\
(t, \min\{u, n-1\}) & \text{if } D_n \cap V = \emptyset \\
((t-1)^+, \min\{u, n-1\}) & \text{otherwise}
\end{cases}
\]

**Example**

\[
D_1 = \{a, b\} \quad \text{among}((x_1, x_2, x_3, x_4, x_5),\{c,d\}, t, u)
\]
\[
D_2 = \{a, b, c\} \quad \text{among}((x_1, x_2, x_3, x_4),\{c,d\},(t-1)^+, u-1)
\]
\[
D_3 = \{a, d\} \quad \text{among}((x_1, x_2, x_3),\{c,d\},(t-2)^+, u-2)
\]
\[
D_4 = \{c, d\} \quad \text{among}((x_1, x_2),\{c,d\},(t-3)^+, \min\{u-2, 2\})
\]
\[
D_5 = \{d\} \quad \text{among}((x_1),\{c,d\},(t-4)^+, \min\{u-2, 1\}) \quad \text{among}((\),\{c,d\},(t-4)^+, \min\{u-2, 0\})
\]

Feasible if and only if \((t-4)^+ \leq \min\{u-2, 0\}\)
**J-Consistency for Sequence**

- Projection is based on an integrality property.
  - The coefficient matrix of the inequality formulation has **consecutive ones** property.
  - So projection of the convex hull of the feasible set is an **integral polyhedron**.
    - **Polyhedral projection** therefore suffices.
    - Straightforward (but tedious) application of **Fourier elimination** yields the projection.
**J-Consistency for Sequence**

- Projection is based on an integrality property.
  - The coefficient matrix of the inequality formulation has **consecutive ones** property.
  - So projection of the convex hull of the feasible set is an **integral polyhedron**.
    - Polyhedral projection therefore suffices.
    - Straightforward (but tedious) application of **Fourier elimination** yields the projection.

- Projection onto any subset of variables is a **generalized sequence constraint**.
  - Complexity of projecting out $x_k$ is $O(kq)$, where $q =$ length of the overlapping sequences.
Following standard convention, we assume without loss of generality that the sequence constraint applies to 0-1 variables \( x_1, \ldots, x_n \) [23, 46]. It enforces overlapping constraints of the form

\[
\text{among}\left((x_{\ell-q+1}, \ldots, x_\ell), \{1\}, L_\ell, U_\ell\right)
\]  

(5)

**Theorem 4.** Given any \( k \in \{0, \ldots, n\} \), the projection of the sequence constraint defined by (5) onto \( (x_1, \ldots, x_k) \) is described by a generalized sequence constraint that enforces constraints of the form

\[
\text{among}\left((x_i, \ldots, x_\ell), \{1\}, L_{\ell-i+1}^\ell, U_{\ell-i+1}^\ell\right)
\]  

(6)

where \( i = \ell - q + 1, \ldots, \ell \) for \( \ell = q, \ldots, k \) and \( i = 1, \ldots, \ell \) for \( \ell = 1, \ldots, q - 1 \). The projection of the sequence constraint onto \( (x_1, \ldots, x_{k-1}) \) is given by (6) with \( L_{\ell-i+1}^\ell \) replaced by \( \hat{L}_{\ell-i+1}^\ell \) and \( U_{\ell-i+1}^\ell \) by \( \hat{U}_{\ell-i+1}^\ell \), where

\[
\hat{L}_i^\ell = \begin{cases} 
\max\{L_i^\ell, L_{i+k-\ell}^k - U_{k-\ell}^k\}, & \text{for } i = 1, \ldots, q - k + \ell, \\
L_i^\ell, & \text{for } i = q - k + \ell + 1, \ldots, q 
\end{cases}
\]  

(7)

\[
\hat{U}_i^\ell = \begin{cases} 
\min\{U_i^\ell, U_{i+k-\ell}^k - L_{k-\ell}^k\}, & \text{for } i = 1, \ldots, q - k + \ell, \\
U_i^\ell, & \text{for } i = q - k + \ell + 1, \ldots, q 
\end{cases}
\]
$J$-Consistency for Sequence

Example \[ \text{among}((x_{t-3}, \ldots, x_t), \{1\}, 2,2), \quad t = 4,5,6 \]
\[ x_1, x_3, x_4, x_6 \in \{0,1\}, \quad x_2, x_5 \in \{1\} \]

To project out $x_6$, add constraint
\[ \text{among}((x_3, x_4, x_5), \{1\}, 1,1) \]
**J-Consistency for Sequence**

**Example**

among((x_{t-3},...,x_t),\{1\},2,2), \ t = 4,5,6

x_1, x_3, x_4, x_6 \in \{0,1\}, \ x_2, x_5 \in \{1\}

To project out x_6, add constraint

among((x_3, x_4, x_5),\{1\},1,1)

To project out x_5, add constraints

among((x_2, x_3, x_4),\{1\},1,1) \ among((x_3, x_4),\{1\},0,0)
**J-Consistency for Sequence**

**Example**

\[ \text{among}(\langle x_{t-3}, \ldots, x_t \rangle, \{1\}, 2, 2), \quad t = 4, 5, 6 \]

\[ x_1, x_3, x_4, x_6 \in \{0, 1\}, \quad x_2, x_5 \in \{1\} \]

To project out \( x_6 \), add constraint

\[ \text{among}(\langle x_3, x_4, x_5 \rangle, \{1\}, 1, 1) \]

To project out \( x_5 \), add constraints

\[ \text{among}(\langle x_2, x_3, x_4 \rangle, \{1\}, 1, 1) \]
\[ \text{among}(\langle x_3, x_4 \rangle, \{1\}, 0, 0) \]

To project out \( x_4 \), add constraints

\[ \text{among}(\langle x_1 \rangle, \{1\}, 1, 1) \]
\[ \text{among}(\langle x_1, x_2, x_3 \rangle, \{1\}, 1, 2) \]
\[ \text{among}(\langle x_2, x_3 \rangle, \{1\}, 0, 1) \]
\[ \text{among}(\langle x_3 \rangle, \{1\}, 0, 0) \]
**J-Consistency for Sequence**

**Example**

\[
\text{among}( (x_{t-3}, \ldots, x_t), \{1\}, 2, 2), \quad t = 4, 5, 6
\]

\[
x_1, x_3, x_4, x_6 \in \{0, 1\}, \quad x_2, x_5 \in \{1\}
\]

To project out \(x_6\), add constraint

\[
\text{among}( (x_3, x_4, x_5), \{1\}, 1, 1)
\]

To project out \(x_5\), add constraints

\[
\text{among}( (x_2, x_3, x_4), \{1\}, 1, 1) \quad \text{among}( (x_3, x_4), \{1\}, 0, 0)
\]

To project out \(x_4\), add constraints

\[
\text{among}( (x_1), \{1\}, 1, 1) \quad \text{among}( (x_1, x_2, x_3), \{1\}, 1, 2)
\]

\[
\text{among}( (x_2, x_3), \{1\}, 0, 1) \quad \text{among}( (x_3), \{1\}, 0, 0)
\]

To project out \(x_3\), fix \((x_1, x_2) = (1, 1)\)
J-Consistency for Regular Constraint

• Projection can be read from state transition graph.
  – Complexity of projecting onto $x_1, \ldots, x_k$ for all $k$ is $O(nm^2)$, where $n =$ number of variables, $m =$ max number of states per stage.
**J-Consistency for Regular Constraint**

- **Projection can be read from state transition graph.**
  - Complexity of projecting onto \( x_1, \ldots, x_k \) for all \( k \) is \( O(nm^2) \), where \( n = \text{number of variables} \), \( m = \text{max number of states per stage} \).

- **Shift scheduling example**
  - Assign each worker to shift \( x_i \in \{a,b,c\} \) on each day \( i = 1,\ldots,7 \).
  - Must work any given shift 2 or 3 days in a row.
  - No direct transition between shifts \( a \) and \( c \).
  - Variable domains: \( D_1 = D_5 = \{a,c\} \), \( D_2 = \{a,b,c\} \),
    \( D_3 = D_6 = D_7 = \{a,b\} \), \( D_4 = \{b,c\} \)
Deterministic finite automaton for this problem instance:

- Absorbing state

Regular language expression:

\[((aa|aaa)(bb|bbb))^*|((cc|ccc)(bb|bbb))^*\](c|(aa|aaa)|(cc|ccc))
J-Consistency for Regular Constraint

State transition graph for 7 stages
Dashed lines lead to unreachable states.

\[ D_j = \begin{cases} 
\{a, c\} & \text{for } j = 1 \\
\{a, b, c\} & \text{for } j = 2 \\
\{a, b\} & \text{for } j = 3 \\
\{b, c\} & \text{for } j = 4 \\
\{a, c\} & \text{for } j = 5 \\
\{a, b\} & \text{for } j = 6 \\
\{a, b\} & \text{for } j = 7 \\
\end{cases} \]
**J-Consistency for Regular Constraint**

State transition graph for 7 stages
Dashed lines lead to unreachable states.

\[
\begin{align*}
D_j &= \{a, c\} \quad \{a, b, c\} \quad \{a, b\} \quad \{b, c\} \quad \{a, c\} \quad \{a, b\} \quad \{a, b\} \\
D'_j &= \{a, c\} \quad \{a, c\} \quad \{b\} \quad \{b\} \quad \{a\} \quad \{a\} \quad \{a\}
\end{align*}
\]

Filtered domains
J-Consistency for Regular Constraint

To project onto $x_1, x_2, x_3$, truncate the graph at stage 4.

\[ D_j = \{a, c\} \quad \{a, b, c\} \quad \{a, b\} \quad \{b, c\} \quad \{a, c\} \quad \{a, b\} \quad \{a, b\} \]

\[ D'_j = \{a, c\} \quad \{a, c\} \quad \{b\} \quad \{b\} \quad \{a\} \quad \{a\} \quad \{a\} \]
$J$-Consistency for Regular Constraint

To project onto $x_1$, $x_2$, $x_3$, truncate the graph at stage 4.

\[ j = \begin{array}{cccc}
1 & 2 & 3 & 4 \\
D_j = \{a, c\} & \{a, b, c\} & \{a, b\} \\
D'_j = \{a, c\} & \{a, c\} & \{b\}
\end{array} \]
**J-Consistency for Regular Constraint**

To project onto $x_1$, $x_2$, $x_3$, truncate the graph at stage 4.

Resulting graph can be viewed as a constraint that describes the projection.

Constraint is easily propagated through a relaxed decision diagram.

\[
\begin{align*}
  j &= 1 & 2 & 3 & 4 \\
  D_j &= \{a, c\} & \{a, b, c\} & \{a, b\} \\
  D'_j &= \{a, c\} & \{a, c\} & \{b\}
\end{align*}
\]
J-Consistency for SAT

- Project clause set onto $J = \{x_1, x_2, x_3\}$.

Projection is

\[
\begin{align*}
    x_1 & \lor x_2 \\
    x_1 & \lor x_3 \\
\end{align*}
\]
**J-Consistency for SAT**

- **Use logic-based Benders decomposition**
  - Benders cuts describe projection onto master problem variables.

\[ x_1 \lor x_2 \]

**Diagram:**
- Current Master problem
- Benders cut from previous iteration
**J-Consistency for SAT**

- Use logic based Benders decomposition
  - Benders cuts describe projection onto master problem variables.

\[
\begin{align*}
X_1 \lor X_2 & \\
\text{solution of master} & = (0,1,0) \\
\text{Current} & \\
\text{Master problem} & \\
\text{Resulting} & \\
\text{subproblem} & \\
& \begin{align*}
& X_4 \lor X_5 \\
& X_4 \lor \overline{X}_5 \\
& X_5 \lor X_6 \\
& X_5 \lor \overline{X}_6 \\
& \overline{X}_4 \lor X_5 \\
& \overline{X}_4 \lor \overline{X}_5
\end{align*}
\]
Use logic based Benders decomposition

- Benders cuts describe projection onto master problem variables.

\[ X_1 \lor X_2 \]

Solution of master: \((x_1, x_2, x_3) = (0, 1, 0)\)

Current Master problem

Resulting subproblem

Subproblem is \textbf{infeasible}.

\((x_1, x_3) = (0, 0)\) creates infeasibility
**J-Consistency for SAT**

- Use logic based Benders decomposition
  - Benders cuts describe projection onto master problem variables.

Current Master problem

\[ x_1 \lor x_2 \]
\[ x_1 \lor x_3 \]

solution of master

\((x_1, x_2, x_3) = (0, 1, 0)\)

Benders cut

(nogood)

Subproblem is infeasible.

\((x_1, x_3) = (0, 0)\)

creates infeasibility

Resulting subproblem

\[ x_4 \lor x_5 \]
\[ x_4 \lor \overline{x}_5 \]
\[ x_5 \lor x_6 \]
\[ x_5 \lor \overline{x}_6 \]
\[ \overline{x}_4 \lor x_5 \]
\[ \overline{x}_4 \lor \overline{x}_5 \]
**J-Consistency for SAT**

- Use logic based Benders decomposition
  - Benders cuts describe projection onto master problem variables.

Current Master problem

\[
\begin{align*}
X_1 & \lor X_2 \\
X_1 & \lor X_3 \\
\end{align*}
\]

solution of master \((x_1,x_2,x_3) = (0,1,1)\)

Resulting subproblem

\[
\begin{align*}
x_4 & \lor x_5 \\
x_4 & \lor \overline{x}_5 \\
x_5 & \lor x_6 \\
x_5 & \lor \overline{x}_6 \\
\end{align*}
\]
**J-Consistency for SAT**

- Use logic based Benders decomposition
  - Benders cuts describe projection onto master problem variables.

Current Master problem

\[ \begin{align*}
x_1 & \lor x_2 \\
x_1 & \lor x_3
\end{align*} \]

solution of master \((x_1, x_2, x_3) = (0,1,1)\)

Resulting subproblem

\[ \begin{align*}
x_4 & \lor x_5 \\
x_4 & \lor \bar{x}_5 \\
\bar{x}_5 & \lor x_6 \\
\bar{x}_5 & \lor \bar{x}_6
\end{align*} \]

Subproblem is **feasible**
**J-Consistency for SAT**

- Use logic based Benders decomposition
  - Benders cuts describe projection onto master problem variables.

Current Master problem

\[
\begin{align*}
X_1 \lor X_2 \\
X_1 \lor X_3 \\
X_1 \lor \bar{X}_2 \lor \bar{X}_3
\end{align*}
\]

solution of master \((x_1, x_2, x_3) = (0,1,1)\)

Resulting subproblem

\[
\begin{align*}
x_4 \lor x_5 \\
x_4 \lor \bar{x}_5 \\
x_5 \lor x_6 \\
x_5 \lor \bar{x}_6
\end{align*}
\]

Enumerative Benders cut

Subproblem is **feasible**
J-Consistency for SAT

• Use logic based Benders decomposition
  – Benders cuts describe projection onto master problem variables.

Current Master problem

\[ X_1 \lor X_2 \]
\[ X_1 \lor X_3 \]
\[ X_1 \lor \overline{X}_2 \lor \overline{X}_3 \]

solution of master
\((x_1, x_2, x_3) = (0, 1, 1)\)

Enumerative Benders cut

Resulting subproblem

\[ x_4 \lor x_5 \]
\[ x_4 \lor \overline{x}_5 \]
\[ x_5 \lor x_6 \]
\[ x_5 \lor \overline{x}_6 \]

Continue until master is infeasible.
Black Benders cuts describe projection.

JH and Yan (1995)
JH (2012)
J-Consistency for SAT

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on $x_1$, $x_2$, $x_3$ first.
J-Consistency for SAT

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on $x_1$, $x_2$, $x_3$ first.
J-Consistency for SAT

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on $x_1$, $x_2$, $x_3$ first.

Conflict clauses

Backtrack to $x_3$ at feasible leaf nodes
J-Consistency for SAT

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on $x_1, x_2, x_3$ first.

Conflict clauses containing $x_1, x_2, x_3$ describe projection
J-Consistency for Alldiff Constraint

• Projection is inherently complicated.
  – But it can simplify for small domains.

• The result is a disjunction of constraint sets,
  – …each of which contains an alldiff constraint and some atmost constraints.
$J$-Consistency for Alldiff Constraint

Example

\[
\text{alldiff} \left( x_1, x_2, x_3, x_4, x_5 \right)
\]

\[
D_1 = \{a, b, c\}, \ D_2 = \{c, d, e\}, \ D_3 = \{d, e, f\}, \ D_4 = \{e, f, g\}, \ D_5 = \{a, f, g\}
\]
J-Consistency for Alldiff Constraint

Example

\( \text{alldiff}\left(x_1, x_2, x_3, x_4, x_5\right) \)

\( D_1 = \{a, b, c\}, D_2 = \{c, d, e\}, D_3 = \{d, e, f\}, D_4 = \{e, f, g\}, D_5 = \{a, f, g\} \)

Projecting out \( x_5 \), we get

\( \text{alldiff}\left(x_1, x_2, x_3, x_4\right), \text{atmost}\left(\left(x_1, x_2, x_3, x_4\right), \{a, f, g\}, 2\right) \)

because \( x_5 \) must take one of the values in \( \{a, f, g\} \), leaving 2 for other \( x_i \)s.
**J-Consistency for Alldiff Constraint**

**Example**

\[
\text{alldiff}(x_1, x_2, x_3, x_4, x_5)
\]

\[
D_1 = \{a, b, c\}, \quad D_2 = \{c, d, e\}, \quad D_3 = \{d, e, f\}, \quad D_4 = \{e, f, g\}, \quad D_5 = \{a, f, g\}
\]

Projecting out \(x_5\), we get

\[
\text{alldiff}(x_1, x_2, x_3, x_4), \quad \text{atmost}((x_1, x_2, x_3, x_4), \{a, f, g\}, 2)
\]

because \(x_5\) must take one of the values in \(\{a, f, g\}\), leaving 2 for other \(x_i\)s.

Projecting out \(x_4\), we note that \(x_4 \in \{a, f, g\}\) or \(x_4 \notin \{a, f, g\}\).

If \(x_4 \in \{a, f, g\}\), we get

\[
\text{alldiff}(x_1, x_2, x_3), \quad \text{atmost}((x_1, x_2, x_3), \{a, f, g\}, 1)
\]

If \(x_4 \notin \{a, f, g\}\), we get \(x_4 = e\), and we remove \(e\) from other domains.

So the projection is

\[
\begin{bmatrix}
\text{alldiff}(x_1, x_2, x_3) \\
\text{atmost}((x_1, x_2, x_3), \{a, f, g\}, 1)
\end{bmatrix}
\]

\[
\bigvee\begin{bmatrix}
D_1 = \{a, b, c\} \\
D_2 = \{c, d\} \\
D_3 = \{d, f\}
\end{bmatrix}
\]
J-Consistency for Alldiff Constraint

Example

\[ \text{alldiff}(x_1, x_2, x_3, x_4, x_5) \]

\[ D_1 = \{a, b, c\}, \ D_2 = \{c, d, e\}, \ D_3 = \{d, e, f\}, \ D_4 = \{e, f, g\}, \ D_5 = \{a, f, g\} \]

Projecting out \( x_3 \), we get simply

\[ \text{alldiff}(x_1, x_2) \]

Projecting out \( x_2 \), we get the original domain for \( x_1 \)

\[ D_1 = \{a, b, c\} \]
Algorithm 2 Given a projection of $\text{alldiff}(x^n)$ onto $x^k$, this algorithm computes a projection onto $x^{k-1}$. The projection onto $x^k$ is assumed to be a disjunction of constraint sets, each of which has the form (10). The above algorithm is applied to each disjunct, after which the disjunction of all created constraint sets forms the projection onto $x^{k-1}$.

For all $i \in I$: if $\text{atmost}(x^k, V_i, b_i)$ is redundant then remove $i$ from $I$.

For all $i \in I$:

If $D_k \cap V_i \neq \emptyset$ then

If $b_i > 1$ then

Create a constraint set consisting of $\text{alldiff}(x^{k-1})$,
$\text{atmost}(x^{k-1}, V_{i'}, b_{i'})$ for $i' \in I \setminus \{i\}$, and $\text{atmost} (x^{k-1}, V_i, b_i - 1)$.

Let $R = D_k \setminus \bigcup_{i \in I} V_i$.
If $|R| > 1$ then

Create a constraint set consisting of $\text{alldiff}(x^{k-1})$,
$\text{atmost}(x^{k-1}, V_{i'}, b_{i'})$ for $i' \in I$, and $\text{atmost} (x^{k-1}, R, |R| - 1)$.

Else if $|R| = 1$ then

Let $R = \{v\}$ and remove $v$ from $D_j$ for $j = 1, \ldots, k - 1$ and from $V_i$ for $i \in I$.

If $D_j$ is nonempty for $j = 1, \ldots, k - 1$ then

For all $i' \in I$: if $\text{atmost}(x^{k-1}, V_{i'}, b_{i'})$ is redundant then remove $i'$ from $i$.

Create a constraint set consisting of $\text{alldiff}(x^{k-1})$ and
$\text{atmost}(x^{k-1}, V_{i'}, b_{i'})$ for $i' \in I$. 
Research Program

- Investigate projection for other global constraints.
  - Design $J$-consistency maintenance algorithms (complete or incomplete)

- Add $J$-consistency maintenance to a CP solver.
  - Along with propagation through decision diagrams.
Research Program

• Investigate projection for other global constraints.
  – Design $J$-consistency maintenance algorithms (complete or incomplete)

• Add $J$-consistency maintenance to a CP solver.
  – Along with propagation through decision diagrams.

Congratulations!
You survived 98 slides!