Collaboration Opportunities for OR and Constraint Programming

John Hooker
Carnegie Mellon University

NSF CMMI Conference
July 2012
Two Ways Fields Can Collaborate

• Combine complementary techniques
  • Can yield results.
• Unify the ideas.
  • More fun intellectually, also yields results.
• Let’s look at OR/CP collaboration from this perspective.
Outline

• What is constraint programming?
• Integrating OR and CP
• A unifying principle – inference duality.
• Other areas of unification

Caveat: This is a high-level overview. Don’t worry about the technical details.
What Is Constraint Programming?

Brief intellectual history
Modeling example
Applications
Strengths and Weaknesses
First Step: Logic Programming

- Attempt to **unify** procedural and declarative modeling
  - **Procedural**: Write the algorithm (CS)
  - **Declarative**: Write the constraints (OR)
  - **Logic programming**: Propositions are also procedural goals

Example of Prolog

1. \texttt{ancestor}(X, Y) ← \texttt{parent}(X, Y).
2. \texttt{ancestor}(X, Z) ← \texttt{parent}(X, Y), \texttt{ancestor}(Y, Z).
3. \texttt{parent}(a, b).
4. \texttt{parent}(b, c).
Second Step: Constraint Logic Programming

- Interpret unification step in 1\textsuperscript{st} order logic as constraint solving
  - Extend equality constraints in logic to more general constraints.
  - Constraints accumulate in a constraint store at each leaf node.
  - Node is feasible ("succeeds") if constraints have a solution.
Third Step: Constraint Programming

- Drop the logic programming framework.
- View each line of the program as specifying both a **constraint** and a **procedure**.
  - **Constraints** are high-level global constraints
  - The **procedure** removes infeasible values from variable domains (**filtering, domain consistency maintenance**)
  - Passes reduced domains to next constraint (**constraint propagation**).
Modeling example: Employee scheduling

• Schedule four nurses in 8-hour shifts.
• A nurse works at most one shift a day, at least 5 days a week.
• Same schedule every week.
• No shift staffed by more than two different nurses in a week.
• A nurse cannot work different shifts on two consecutive days.
• A nurse who works shift 2 or 3 must do so at least two days in a row.
Two ways to view the problem

Assign nurses to shifts

<table>
<thead>
<tr>
<th></th>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift 1</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Shift 2</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Shift 3</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Assign shifts to nurses

<table>
<thead>
<tr>
<th></th>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nurse A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Nurse B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Nurse C</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Nurse D</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

0 = day off
Use **both** formulations in the same model!

First, assign nurses to shifts.

Let $w_{sd} =$ nurse assigned to shift $s$ on day $d$

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \text{ all } d$$

Schedule 3 different nurses on each day
Use both formulations in the same model!

First, assign nurses to shifts.

Let $w_{sd} =$ nurse assigned to shift $s$ on day $d$

\[
\text{alldiff} \left( w_{1d}, w_{2d}, w_{3d} \right), \quad \text{all } d
\]

\[
\text{cardinality} \left( w \mid (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6) \right)
\]

Each nurse works at least 5 and at most 6 days a week
Use both formulations in the same model!

First, assign nurses to shifts.

Let $w_{sd} =$ nurse assigned to shift $s$ on day $d$

$$\text{alldiff} \left( w_{1d}, w_{2d}, w_{3d} \right), \text{ all } d$$

$$\text{cardinality} \left( w \mid (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6) \right)$$

$$\text{nvalues} \left( w_{s,\text{Sun}}, ..., w_{s,\text{Sat}} \mid 1, 2 \right), \text{ all } s$$

At least 1 and at most 2 nurses work any given shift.
Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let $y_{id} =$ shift assigned to nurse $i$ on day $d$

\[ \text{alldiff } (y_{1d}, y_{2d}, y_{3d}), \text{ all } d \]

Assign a different nurse to each shift on each day.

Redundant, but speeds solution.
Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let $y_{id} =$ shift assigned to nurse $i$ on day $d$

$$\text{alldiff}(y_{1d}, y_{2d}, y_{3d}), \text{ all } d$$

$$\text{stretch}(y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \ | \ (2,3),(2,2),(6,6), P), \text{ all } i$$

Shift 2 or 3 must be worked at least 2 days in a row.
Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let $y_{id} =$ shift assigned to nurse $i$ on day $d$

$$\text{alldiff} \left( y_{1d}, y_{2d}, y_{3d} \right), \text{ all } d$$

$$\text{stretch} \left( y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \left| (2,3),(2,2),(6,6),P \right. \right), \text{ all } i$$

Pattern constraint: $P = \{(s,0),(0,s) \mid s = 1,2,3\}$

Don’t switch shifts without taking at least one day off.
Connect the $w_{sd}$ variables to the $y_{id}$ variables.

Use **channeling constraints**: 

$$w_{y_{id}d} = i, \text{ all } i,d$$

$$y_{w_{sd}d} = s, \text{ all } s,d$$

Channeling constraints increase propagation and make the problem easier to solve.
The complete model is:

\[
\text{alldiff}\left( w_{1d}, w_{2d}, w_{3d} \right), \text{ all } d
\]
\[
\text{cardinality}\left( w \mid (A,B,C,D),(5,5,5,5),(6,6,6,6) \right)
\]
\[
\text{nvalues}\left( w_{s,\text{Sun}},...,w_{s,\text{Sat}} \mid 1,2 \right), \text{ all } s
\]
\[
\text{alldiff}\left( y_{1d}, y_{2d}, y_{3d} \right), \text{ all } d
\]
\[
\text{stretch}\left( y_{i,\text{Sun}},...,y_{i,\text{Sat}} \mid (2,3),(2,2),(6,6),P \right), \text{ all } i
\]
\[
w_{y_{id}d} = i, \text{ all } i,d
\]
\[
y_{w_{sd}d} = s, \text{ all } s,d
\]
Early commercial successes

- Circuit design (Siemens)
- Container port scheduling (Hong Kong and Singapore)
- Real-time control (Siemens, Xerox)
Applications

• Job shop scheduling
• Assembly line smoothing and balancing
• Cellular frequency assignment
• Nurse scheduling
• Shift planning
• Maintenance planning
• Airline crew rostering and scheduling
• Airport gate allocation and stand planning
Applications

• Production scheduling
  chemicals
  aviation
  oil refining
  steel
  lumber
  photographic plates
  tires

• Transport scheduling (food, nuclear fuel)

• Warehouse management

• Course timetabling
## CP and Mathematical Programming

### Comparison

<table>
<thead>
<tr>
<th>CP</th>
<th>MP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic processing</td>
<td>Numerical calculation</td>
</tr>
<tr>
<td>Inference (filtering, constraint propagation)</td>
<td>Relaxation</td>
</tr>
<tr>
<td>High-level modeling (global constraints)</td>
<td>Atomistic modeling (linear inequalities)</td>
</tr>
<tr>
<td>Branching</td>
<td>Branching</td>
</tr>
<tr>
<td>Constraint-based processing</td>
<td>“Independence” of model and algorithm</td>
</tr>
</tbody>
</table>
Integrating OR and CP

Complementary strengths
How to integrate
Computational advantages
Applications & Software
Complementary Strengths

- **CP:**
  - Inference methods
  - Modeling
  - Exploits local structure
  - Good at scheduling

- **OR:**
  - Relaxation methods
  - Duality theory
  - More robust
  - Good with continuous variables

Let’s bring them together!
How to Integrate

• Constraint propagation + relaxation
  – Propagation reduces search space.
  – Relaxation bounds prune the search
• CP-based column generation
  – In branch-and-price methods
  – CP accommodates complex constraints on columns
• Decomposition methods
  – Distinguish master problem and subproblem
  – MILP solves one, CP the other.
• Use CP-style modeling
## Computational Advantage of Integrating CP and OR

### Using CP + relaxation from OR

<table>
<thead>
<tr>
<th>Problem</th>
<th>Speedup</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lesson timetabling (LP + red. cost var. fixing)</td>
<td>2 to 50 times faster than CP</td>
<td>Focacci, Lodi, Milano (1999)</td>
</tr>
<tr>
<td>Piecewise linear costs (LP relaxation)</td>
<td>2 to 120 times faster than MILP</td>
<td>Refalo (1999), Yunes, Aron &amp; Hooker (2010)</td>
</tr>
<tr>
<td>Flow shop scheduling, etc. (LP relaxation + cuts)</td>
<td>4 to 150 times faster than MILP</td>
<td>Hooker &amp; Osorio (1999)</td>
</tr>
</tbody>
</table>
### Computational Advantage of Integrating CP and OR

**Using CP + relaxation from OR**

<table>
<thead>
<tr>
<th><strong>Problem</strong></th>
<th><strong>Speedup</strong></th>
<th><strong>Reference</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Product configuration</td>
<td>30 to 40 times faster than CP, MILP</td>
<td>Thorsteinsson &amp; Ottosson (2001)</td>
</tr>
<tr>
<td>(LP relaxation + cuts)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic recording</td>
<td>1 to 10 times faster than CP, MILP</td>
<td>Sellmann &amp; Fahle (2001)</td>
</tr>
<tr>
<td>(Lagrangean relaxation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stable set problem</td>
<td>Better than CP in less time</td>
<td>Van Hoeve (2001)</td>
</tr>
<tr>
<td>(semidefinite relaxation)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Computational Advantage of Integrating CP and OR

Using CP + relaxation from OR

<table>
<thead>
<tr>
<th>Problem</th>
<th>Speedup</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural design (nonlinear) (LP quasi-relaxation + logic cuts)</td>
<td>Up to 600 times faster than MILP.</td>
<td>Bollapragada, Ghattas &amp; Hooker (2001)</td>
</tr>
<tr>
<td>Radiation therapy planning (Lagrangian relaxation)</td>
<td>10 times faster than CP, MILP</td>
<td>Cambazard, O'Mahony and O'Sullivan (2010)</td>
</tr>
</tbody>
</table>
## Computational Advantage of Integrating CP and OR

Using CP-based Branch and Price

<table>
<thead>
<tr>
<th>Problem</th>
<th>Speedup</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban transit crew scheduling</td>
<td>Schedules 210 trips, vs. 120 for traditional branch and price</td>
<td>Yunes, Moura &amp; de Souza (1999)</td>
</tr>
<tr>
<td>Airline crew rostering</td>
<td>Incorporates complicated work rules</td>
<td>Fahle et al. (2002)</td>
</tr>
<tr>
<td>Traveling tournament scheduling</td>
<td>First to solve 8-team instance</td>
<td>Easton, Nemhauser &amp; Trick (2002)</td>
</tr>
</tbody>
</table>
Computational Advantage of Integrating CP and OR

Using CP/MILP Benders methods

<table>
<thead>
<tr>
<th>Problem</th>
<th>Speedup</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min-cost planning &amp; scheduling</td>
<td>20 to 1000 times faster than CP, MILP</td>
<td>Jain &amp; Grossmann (2001), Thorsteinsson (2001)</td>
</tr>
<tr>
<td>Min-cost planning &amp; scheduling</td>
<td>Solved some MIP-intractable instances in &lt;1 sec</td>
<td>Yunes, Aron &amp; Hooker (2010)</td>
</tr>
<tr>
<td>Polypropylene batch scheduling at BASF</td>
<td>Solved previously insoluble problem in 10 min</td>
<td>Timpe (2002)</td>
</tr>
</tbody>
</table>
Computational Advantage of Integrating CP and OR
Using CP/MILP Benders methods

<table>
<thead>
<tr>
<th>Problem</th>
<th>Speedup</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call center scheduling</td>
<td>Solved twice as many instances as traditional Benders</td>
<td>Benoist, Gaudin, Rottembourg (2002)</td>
</tr>
<tr>
<td>Min-cost, min-makespan planning &amp; cumulative scheduling</td>
<td>100-1000 times faster than CP, MILP</td>
<td>Hooker (2004)</td>
</tr>
<tr>
<td>Min tardiness planning &amp; cumulative scheduling</td>
<td>10-1000 times faster than CP, MILP</td>
<td>Hooker (2005)</td>
</tr>
</tbody>
</table>
Computational Advantage of Integrating CP and OR

Using CP/MILP Benders methods

<table>
<thead>
<tr>
<th>Problem</th>
<th>Speedup</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sports scheduling</td>
<td>Several orders of magnitude speedup vs MILP</td>
<td>Rasmussen &amp; Trick (2007)</td>
</tr>
<tr>
<td>Single-facility scheduling</td>
<td>Schedules several times as many jobs as MILP. Faster and/or more robust than CP.</td>
<td>Coban &amp; Hooker (2012)</td>
</tr>
</tbody>
</table>
### Applications of Integrated CP and OR

Using CP + relaxation from OR

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orthogonal Latin squares</td>
<td>Appa, Magos &amp; Mourtos (2002)</td>
</tr>
<tr>
<td>Truss structure design</td>
<td>Bollapragada, Gattas &amp; Hooker (2001)</td>
</tr>
<tr>
<td>Chemical processing network design</td>
<td>Grossmann et al. (1994), Hooker &amp; Osorio (1999)</td>
</tr>
<tr>
<td>Multiple machine scheduling</td>
<td>Bockmayr &amp; Pisaruk (2003)</td>
</tr>
<tr>
<td>Shuttle transit routing</td>
<td>Quadrifoglio, Dessouky &amp; Ordóñez (2008)</td>
</tr>
<tr>
<td>Boat party scheduling</td>
<td>Hooker &amp; Osorio (1999)</td>
</tr>
</tbody>
</table>
## Applications of Integrated CP and OR

Using CP + relaxation from OR

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multidimensional knapsack problem</td>
<td>Osorio &amp; Glover (2001)</td>
</tr>
<tr>
<td>Factory retrofit planning</td>
<td>Sawaya &amp; Grossmann (2005)</td>
</tr>
<tr>
<td>Strip packing</td>
<td>Sawaya &amp; Grossmann (2005)</td>
</tr>
<tr>
<td>Transport &amp; production problems with piecewise linear costs</td>
<td>Refalo (1999), Ottosson et al. (1999)</td>
</tr>
<tr>
<td>TSP with time windows</td>
<td>Milano &amp; van Hoeve (2002)</td>
</tr>
<tr>
<td>Product configuration</td>
<td>Milano &amp; van Hoeve (2002)</td>
</tr>
</tbody>
</table>
# Applications of Integrated CP and OR

Using CP + relaxation from OR

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network design</td>
<td>Cronholm &amp; Ajili (2004)</td>
</tr>
<tr>
<td>Automatic digital recording</td>
<td>Sellmann &amp; Fahle (2001)</td>
</tr>
<tr>
<td>Traveling tournament problem</td>
<td>Benoist, Laburthe &amp; Rottembourg (2001)</td>
</tr>
<tr>
<td>Resource-constrained shortest path problem</td>
<td>Gellermann, Sellmann &amp; Wright (2005)</td>
</tr>
<tr>
<td>Radiation therapy planning</td>
<td>Cambazard, O’Mahony &amp; O’Sullivan (2010)</td>
</tr>
<tr>
<td>CP domain filtering</td>
<td>Khemmoudj, Bennaceur &amp; Nagih (2005)</td>
</tr>
</tbody>
</table>
Applications of Integrated CP and OR

Using CP-based branch and price

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft scheduling</td>
<td>Grönqvist (2003)</td>
</tr>
<tr>
<td>Bus crew scheduling</td>
<td>Yunes, Moura &amp; de Souza (2005)</td>
</tr>
<tr>
<td>Network design</td>
<td>Chabrier (2003)</td>
</tr>
<tr>
<td>Employee timetabling</td>
<td>Demassey, Pesant &amp; Rousseau (2005)</td>
</tr>
<tr>
<td>Physician scheduling</td>
<td>Gendron, Lebbah &amp; Pesant (2005)</td>
</tr>
<tr>
<td>Radiation therapy planning</td>
<td>Cambazard, O’Mahony &amp; O’Sullivan (2010)</td>
</tr>
</tbody>
</table>
Applications of Integrated CP and OR

Using CP/MILP Benders methods

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic circuit verification</td>
<td>Hooker &amp; Yan (1995)</td>
</tr>
<tr>
<td>Steel production scheduling</td>
<td>Harjunkoski &amp; Grossmann (2001)</td>
</tr>
<tr>
<td>Computer processor scheduling</td>
<td>Cambazard et al. (2004), Benini et al. (2005, 2008),</td>
</tr>
</tbody>
</table>
## Applications of Integrated CP and OR
Using CP/MILP Benders methods

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location-allocation</td>
<td>Fazel-Zarandi &amp; Beck (2009)</td>
</tr>
<tr>
<td>Transport network design</td>
<td>Peterson &amp; Trick (2009)</td>
</tr>
<tr>
<td>Queuing design &amp; control</td>
<td>Terekhov, Beck &amp; Brown (2007)</td>
</tr>
</tbody>
</table>
Software for Integrated Methods

- **ECLiPSe**
  - Exchanges information between ECLiPSe solver, Xpress-MP
- **OPL Studio (IBM)**
  - Combines CPLEX and ILOG CP Optimizer with script language
- **Mosel (FICO)**
  - Combines Xpress-MP, Xpress-Kalis with low-level modeling
- **SIMPL (CMU)**
  - Full integration with high-level modeling (prototype)
- **SCIP (ZIB)**
  - Combines MILP and CP-based propagation
- **G12 (NICTA)**
  - Converts generic model to one that invokes cooperating solvers
- **BARON (global optimization)**
  - Combines convexification and interval propagation
A Unifying Principle – Inference Duality

Optimization duals
Constraint-directed search
Example: Conflict clauses in SAT
Application to new problems
Unifying Theories

- **Synthesis** – 19\(^{th}\) century
  - Maxwell’s equations
  - Evolution by natural selection

- **Analysis** – 20\(^{th}\) century
  - Except in reductive schemes (DNA, TOE in physics)
  - Abstraction is **not** synthesis (Bourbaki school)

- **Return** to synthesis in 21\(^{st}\) century?

- **Duality** – a unifying principle for OR/CP/AI.
Duality in Mathematics

• Most mathematical duals are symmetric
  – Dual of dual = original

• Polar duality of polytopes
  – Facets / vertices

• Duality in projective geometry
  – Points / lines

• Duality of vector spaces
  – Row rank / column rank

Fano plane
Optimization Duals

- Generally not symmetric:
  - Surrogate dual
  - Lagrangean dual
  - Superadditive dual
  - LP dual is an exception

- All are inference duals and relaxation duals.
Inference Dual

- Primal problem:
  - Find a feasible solution

- Dual problem:
  - Find a proof of optimality

- Dual is defined by logical inference method
  - Nonnegative linear combination + domination → surrogate dual
  - Nonnegative linear combination + different type of domination → Lagrangean dual
  - LP dual is special case of both
Linear Programming Duality

An LP can be viewed as an inference problem...

\[
\begin{align*}
\min \quad & cx \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

\[
\begin{align*}
\max \quad & v \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

Dual problem: Find the tightest lower bound on the objective function that is implied by the constraints.
Linear Programming Duality

An LP can be viewed as an inference problem...

\[ \min cx = \max \nu \]

\[ Ax \geq b \]

\[ x \geq 0 \]

From Farkas Lemma,

\[ Ax \geq b \Rightarrow cx \geq \nu \]

iff \[ \lambda Ax \geq \lambda b \text{ dominates } cx \geq \nu \]

for some \[ \lambda \geq 0 \]

\[ \lambda A \leq c \text{ and } \lambda b \geq \nu \]
Linear Programming Duality

An LP can be viewed as an inference problem...

\[
\begin{align*}
\min \ cx &= \max \ v & \quad \Rightarrow \\
Ax \geq b &\quad Ax \geq b \Rightarrow cx \geq v \\
x \geq 0 &\quad \lambda A \leq c \\
\end{align*}
\]

We get classical LP dual

From Farkas Lemma:

\[
\begin{align*}
Ax \geq b &\quad \Rightarrow cx \geq v \\
\lambda A \geq \lambda b &\quad \text{dominates} \quad cx \geq v \\
\lambda A \leq c &\quad \text{and} \quad \lambda b \geq v
\end{align*}
\]
Constraint-based Search

- **Inference dual** is the basis for a wide variety of constraint-based search methods
  - Branching and dynamic backtracking.
  - Benders decompositions and its generalizations.
  - Satisfiability solvers.
Constraint-based Search

• Most combinatorial methods search over partial solutions.
  – Nodes of a branching tree.
  – Solutions of Benders master problem.
  – Partial solution defines a subproblem.
• Dual solution (proof) of subproblem defines a nogood constraint.
  – A form of learning.
  – What can deduced from the same proof schema if the premises change?
  – That is, for different partial solutions?
Constraint-based Search

- **Nogood constraints:**
  - Rule out infeasible partial solutions.
  - Or bound value of future partial solutions.

- **Example: Benders cuts**
  - Classical cuts from LP dual of subproblem.

- **Example: Conflict clauses in SAT**
  - Based on unit resolution proof at leaf node.
Conflict clauses in SAT

Example:

\[
\begin{align*}
  x_1 & \lor x_5 \lor x_6 \\
  x_1 & \lor x_5 \lor \neg x_6 \\
  x_2 & \lor \neg x_5 \lor x_6 \\
  x_2 & \lor \neg x_5 \lor \neg x_6 \\
  \neg x_1 & \lor x_3 \lor x_4 \\
  \neg x_2 & \lor x_3 \lor x_4 \\
  \neg x_1 & \lor \neg x_3 \\
  \neg x_1 & \lor \neg x_4 \\
  \neg x_2 & \lor \neg x_3 \\
  \neg x_2 & \lor \neg x_4 
\end{align*}
\]
Conflict clauses in SAT

SAT problem + partial assignment \((x_1, \ldots, x_5) = (F, F, F, F, F)\) is infeasible
SAT problem + partial assignment \((x_1, \ldots, x_5) = (F,F,F,F,F)\) is infeasible.

Solution of inference dual is a unit resolution proof of infeasibility.
Conflict clauses in SAT

SAT problem + partial assignment \((x_1, \ldots, x_5) = (F, F, F, F, F)\) is infeasible

Solution of inference dual is a unit resolution proof of infeasibility.

Proof remains valid when only \(x_1\) and \(x_5\) are fixed to F.
Conflict clauses in SAT

SAT problem + partial assignment \((x_1,\ldots,x_5) = (F,F,F,F,F)\) is infeasible

Solution of inference dual is a unit resolution proof of infeasibility.

Proof remains valid when only \(x_1\) and \(x_5\) are fixed to \(F\).

So we have the conflict clause \(x_1 \lor x_5\)
Conflict clauses in SAT

Next partial assignment \((x_1, \ldots, x_5) = (F, F, F, F, T)\) must satisfy nogood
Next partial assignment \((x_1, x_2) = (F, T)\) must satisfy previous 2 nogoods and their resolvent \(x_1 \lor x_2\)
At this point the accumulated nogoods are unsatisfiable and the search is exhaustive.

Conflict clauses in SAT
Conflict Clauses in SAT

- In practice…
  - Conflict clauses are obtained from **implication graph**.
- A case of constraint-directed branching
  - Long used in AI (backjumping, dynamic backtracking)
  - Recently used in **MIP solvers**
Application to New Problems

• Algorithmic payoff
  – In any optimization subproblem, inference dual can give rise to nogood constraints
  – Leading to a new constraint-based search method.

• Example: Logic-based Benders
  – Applications and speedups mentioned earlier.
Other Possibilities for Unification
Cutting Planes as Logical Inference

- Previous work
  - Resolution as a cutting plane method.
  - Cutting plane proof theory
- Cutting planes and CP
  - How do cutting planes reducing backtracking... by helping to achieve domain consistency or $k$-consistency?
  - Almost completely unexplored
Dependency Graphs and Induced Width

- Unites ideas from several areas
  - Nonserial dynamic programming, Markov trees, join trees, belief logics, Bayesian networks, database theory, $k$-trees, pseudoboolean optimization, bucket elimination, location theory
- Provides:
  - Measure of complexity
  - Recursive solution algorithms
  - Relaxation and relaxation duals.
Graphical Representations of Feasible Sets

- And/or graphs, binary decision diagrams.
- Provide tools for restriction, relaxation
  - Restriction provides a primal heuristic
  - Relaxation provides bounds, etc.
- Structural similarity to continuous relaxation
  - Bounds, pseudocosts, separating cuts
  - Example: MDD relaxations
Exhaustive and Heuristic Search

• **Goal:** Move gracefully from one to the other
• **Unifying idea:**
  – Special cases of the same *restrict-infer-relax* paradigm
  – Example: tabu search as constraint-directed search
• **Incorporate** inference and relaxation mechanisms into metaheuristics
  – Simulated annealing, evolutionary algorithms, ant colony optimization, particle swarm optimization.
Stochastic Optimization, DP, and Learning

• Next talk…