Multivalued Decision Diagrams
and What They Can Do for You

J. N. Hooker
Carnegie Mellon University

Workshop on Constraint Reasoning and Graphical Structures
CP 2010
Two Roles for Graphical Structures

- Problem-solving device
Two Roles for Graphical Structures

- Problem-solving device

- The solution.
  - Transform the problem to a more **transparent** data structure.
MDDs

- MDDs can play both roles.
  - An aid to constraint solving and optimization
  - A transparent representation of the solution space

[Diagram of a graph]
MDDs can play both roles.
- An aid to constraint solving and optimization
- A transparent representation of the solution space

This is a conceptual overview.
- By no means complete.
Outline

- Introduction to MDDs
- MDDs as Propagators
- MDDs as Relaxations
- MDDs as Restrictions
- MDDs as Transparent Data Structures
- Cost-Bounded MDDs
- Nonserial MDDs and Dynamic Programming
Introduction to MDDs
BDDs and MDDs

- A binary decision diagram (BDD) represents a boolean function $f(x_1, \ldots, x_n)$.
  - With binary $x_j$. 
BDDs and MDDs

- A binary decision diagram (BDD) represents a boolean function \( f(x_1, \ldots, x_n) \).
  - With binary \( x_j \).
- Multivalued decision diagram (MDD).
  - General discrete \( x_j \)
BDDs and MDDs

- Every path to 1 represents assignments to \( x_0, \ldots, x_{n-1} \) for which \( f(x_0, \ldots, x_{n-1}) = 1 \).
BDDs and MDDs

- Every path to 1 represents assignments to $x_0, \ldots, x_{n-1}$ for which $f(x_0, \ldots, x_{n-1}) = 1$.
  - Can be viewed as feasible solutions of a constraint.
BDDs and MDDs

- Every path to 1 represents assignments to $x_0, \ldots, x_{n-1}$ for which $f(x_0, \ldots, x_{n-1}) = 1$.
  - Can be viewed as feasible solutions of a constraint.
  - In this case,
    $$2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$
BDDs and MDDs

- Every path to 1 represents assignments to $x_0, \ldots, x_{n-1}$ for which $f(x_0, \ldots, x_{n-1}) = 1$.
  - Can be viewed as feasible solutions of a constraint.
  - In this case,
    $2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$

$x = (1,*,1,*)$
BDDs and MDDs

- Every path to 1 represents assignments to \( x_0, \ldots, x_{n-1} \) for which \( f(x_0, \ldots, x_{n-1}) = 1 \).
  - Can be viewed as feasible solutions of a constraint.
  - In this case,
    \[
    2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7
    \]

\[
x = (1,*,1,*), (1,*,0,1)
\]
BDDs and MDDs

- Every path to 1 represents assignments to $x_0, \ldots, x_{n-1}$ for which $f(x_0, \ldots, x_{n-1}) = 1$.
  - Can be viewed as feasible solutions of a constraint.
  - In this case,
    \[2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7\]

$x = (1,*,1,*)$, $(1,*,0,1)$, $(0,1,1,*)$
BDDs and MDDs

- Every path to 1 represents assignments to $x_0, \ldots, x_{n-1}$ for which $f(x_0, \ldots, x_{n-1}) = 1$.
  - Can be viewed as feasible solutions of a constraint.
  - In this case,
    $$2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$

$x = (1,*,1,*), (1,*,0,1), (0,1,1,*), (0,1,0,1)$
Reduced MDDs

- There is a **unique reduced** MDD for any given constraint.
  - Once the variable ordering is specified.

- The reduced MDD can be viewed as a branching tree with redundancy removed.
  - Superimpose isomorphic subtrees.
  - Remove redundant nodes.
Branching tree for 0-1 inequality

$2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$

1 indicates feasible solution, 0 infeasible
Branching tree for 0-1 inequality

\[ 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7 \]

Remove redundant nodes…
Superimpose identical subtrees…
Superimpose identical subtrees…
Superimpose identical leaf nodes...
as generated by software
Optimizing with MDDs

• Optimal solution corresponds to a shortest path to 1 in the MDD.
  – Provided the objective function is additively separable

\[ g(x) = \sum_j g_j(x_j) \]
\[
\min 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to} \quad 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7
\]

Edge lengths reflect terms of objective function
\[ \min 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to} \quad 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7 \]

\[ 2 \cdot 1 + \min_{x_i \in \{0,1\}} \{-3x_i\} \]
\[
\begin{align*}
\text{min } 2x_0 - 3x_1 + 4x_2 + 6x_3 & \quad \text{subject to } 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7 \\
2 \cdot 1 + \min_{x_1 \in \{0, 1\}} \{-3x_1\} & \rightarrow -1 \\
4 \cdot 1 + \min_{x_3 \in \{0, 1\}} \{6x_3\} & \rightarrow 4 \\
\end{align*}
\]
\[
\text{min } 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to } 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7
\]

Shortest path has length 1
Optimal solution:
\((x_0, x_1, x_2, x_3) = (0,1,1,0)\)

Set to minimizing value
Combine constraints by conjoining MDDs

\[ 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7 \quad x_0 + x_1 + x_2 + x_3 \leq 2 \]
Combine constraints by conjoining MDDs

\[ 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7 \quad x_0 + x_1 + x_2 + x_3 \leq 2 \]
Combine constraints by conjoining MDDs

Complexity $\leq$ product of MDD node counts

\[ 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7 \quad x_0 + x_1 + x_2 + x_3 \leq 2 \]
BDDs and MDDs

• MDD can grow exponentially with number of variables
  – ...but is compact in some important cases.
  – Size can be sensitive to variable ordering.

• MDD doesn't care whether the problem is linear or nonlinear, convex or nonconvex.
  – Functions can take any form (polynomial, etc.).
  – But objective function must be separable.
Constructing MDDs

• One can generate an MDD by enumerating search tree.
  – Intelligent caching to identify reduced form.
  – Known bound on optimal value can prune the search.

• Most BDD software computes BDD for an expression by combining BDDs for subexpressions.
Constructing MDDs

- One **can** generate an MDD by enumerating search tree.
  - Intelligent caching to identify reduced form.
  - Known bound on optimal value can prune the search.

- Most BDD software computes BDD for an expression by combining BDDs for subexpressions.

- Several ways to **limit size** of MDD:
  - Use cost bounding to exclude bad solutions when combining.
  - Create **relaxed** or **restricted** MDD of limited width.
Constructing MDDs

- BDD for a knapsack constraint can be surprisingly small...
Constructing MDDs

- BDD for a knapsack constraint can be surprisingly small...

The 0-1 inequality

\[
300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6 + 230x_7 + 190x_8 + 200x_9 + \\
400x_{10} + 200x_{11} + 400x_{12} + 200x_{13} + 400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \leq 2700
\]

has 117,520 minimal feasible solutions (or minimal covers)

But its reduced BDD has only 152 nodes...
Historical Applications of BDDs

• Circuit checking.
  – Want to implement a boolean function on a chip.
  – Generate reduced BDDs for the function and the chip and compare.
  – BDD size varies.
    • Adders are polynomial.
    • Multipliers are exponential.
Historical Applications of BDDs

• Circuit checking.
  – Want to implement a boolean function on a chip.
  – Generate reduced BDDs for the function and the chip and compare.
  – BDD size varies.
    • Adders are polynomial.
    • Multipliers are exponential.

• Configuration
  – Encode possible design configurations in a BDD.
  – Quickly deduce consequences of fixing some variables.
  – Transparency of BDD – easily queried.
MDDs as Propagators
MDDs as Propagators

• We can replace the **domain store** in CP with a **relaxed MDD**.
  – All feasible solutions are paths in the MDD
  – But not all paths are feasible solutions.
MDDs as Propagators

- We can replace the **domain store** in CP with a **relaxed MDD**.
  - All feasible solutions are paths in the MDD
  - But not all paths are feasible solutions.

- **Limit the width** of the relaxed MDD to avoid blowup.
  - Width of 1 = traditional domain store
MDDs as Propagators

• We can replace the **domain store** in CP with a **relaxed MDD**.
  – All feasible solutions are paths in the MDD
  – But not all paths are feasible solutions.

• **Limit the width** of the relaxed MDD to avoid blowup.
  – Width of 1 = traditional domain store

• **Achieve balance** between search and inference
  – More **information shared** between constraints.
  – It pays to enumerate fewer nodes and spend more time processing each node.
Consider the MDD

Remove paths to 0
Consider the MDD

Each path represents a set of solutions

\[\{1\} \times \{2,3\} \times \{3\} = \{(1,2,3), (1,3,3)\}\]
MDD Relaxation

Original MDD

Relaxation of width 1
= traditional domain store

8 solutions

3 \times 3 \times 3 = 27 solutions
MDD Relaxation

Original MDD

Relaxation of width 2

8 solutions
14 solutions
MDD-Guided Branching

\[
x_1 \in \{1,2,3\}
\]

\[
x_2 \in \{1,2,3\}
\]

Relaxation of width 2
MDD-Guided Branching

\[
x_1 \in \{1,2,3\} \\
\{1\}
\]

\[
x_2 \in \{1,2,3\} \\
\{1\}
\]

\[
x_3 \in \{1,2\}
\]

Relaxation of width 2

\[
\begin{align*}
&u_1 \\
&u_2 \\
&u_3 \\
&u_4 \\
&u_5 \\
&1
\end{align*}
\]

\[
\{1\} \quad \{2\} \quad \{3\} \\
\{1\} \quad \{1,2\} \\
{2,3} \\
\{2,3\} \\
{3} \\
\{3\}
\]
MDD-Guided Branching

$x_1 \in \{1,2,3\}$

$x_2 \in \{1,2,3\}$

$x_3 \in \{3\}$

$x_3 \in \{1,2\}$

Relaxation of width 2

$u_2 \in \{1\}$

$u_4 \in \{1\}$

$u_5 \in \{1,2\}$

$u_1 \in \{1\}$

$u_3 \in \{3\}$

$u_1 \in \{2\}$

$u_3 \in \{1,2\}$

$u_1 \in \{3\}$

$u_3 \in \{1,2\}$
MDD-Guided Branching

And so forth.

Less branching than with domain store.

Relaxation of width 2
MDDs as Propagators

• To propagate a constraint through an MDD:
  – **Filter** edge domains
  – Refine the MDD by **node splitting**
  – Do not exceed max width
MDDs as Propagators

• To propagate a constraint through an MDD:
  – **Filter** edge domains
  – Refine the MDD by **node splitting**
  – Do not exceed max width

• **Example:**
  – Propagate **alldiff** in an MDD relaxation of width 3
Propagation in an MDD

Current MDD relaxation

\[
\begin{align*}
    u_5 &
    \quad 
    \begin{cases}
        u_1 & \{1\} \\
        u_2 & \{1,2\} \\
        u_3 & \{2\} \\
        u_4 & \{1\} \\
        u_6 & \{3\} \\
        u_6 & \{1,2,3\}
    \end{cases}
\end{align*}
\]
Propagation in an MDD

First filter edge domains using alldiff
First filter edge domains using alldiff

Propagation in an MDD

Diagram:
- $u_1$: {1}
- $u_2$: {2}
- $u_3$: {1}
- $u_4$: {1,2}
- $u_5$: {1,2}
- $u_6$: {3}
To split $u_5$:
Identify equivalence classes of incoming edges
To split $u_5$:  
Identify equivalence classes of incoming edges  

These are equivalent for alldiff because they lead to the same set of feasible paths.
These are equivalent for alldiff because they lead to the same set of feasible paths.

Easy to check equivalence for alldiff – all incoming paths use same values
To split $u_5$:
Identify equivalence classes of incoming edges.
Split $u_5$ to receive $\leq 3$ equivalence classes.
Propagation in an MDD

Duplicate outgoing edges.
Propagation in an MDD

Duplicate outgoing edges.
Propagation in an MDD

Filter domains.
Filter domains.
Propagation in an MDD

Alldiff has now been propagated.
MDDs as Propagators

- Computational results – \textit{alldiffs}
  - Conventional domain store – about a million search tree nodes
  - MDD constraint store (width 5) – one node
  - MDD solution about 30 times faster
MDDs as Propagators

- Computational results – **alldiffs**
  - Conventional domain store – about a million search tree nodes
  - MDD constraint store (width 5) – one node
  - MDD solution about 30 times faster

- Computational results – **among**s
  - CP talk Wednesday
MDDs as Relaxations
MDDs as Relaxations

- Shortest path in MDD relaxation provides a bound on optimal value
  - Use as in conventional branch and bound.
MDDs as Relaxations

- Shortest path in MDD relaxation provides a bound on optimal value
  - Use as in conventional branch and bound.

- Strengthen bound by removing infeasible shortest paths.
  - Until shortest path becomes feasible, in which case problem is solved.
  - Or until MDD becomes too large.
  - Roughly analogous to adding Gomory cuts to an IP.
MDDs as Relaxations

- Shortest path in MDD relaxation provides a bound on optimal value
  - Use as in conventional branch and bound.

- Strengthen bound by removing infeasible shortest paths.
  - Until shortest path becomes feasible, in which case problem is solved.
  - Or until MDD becomes too large.
  - Roughly analogous to adding Gomory cuts to an IP.

- MDD can provide **pseudocosts** to guide branching.
  - Test effect on shortest path length of fixing a variable
MDDs as Relaxations

Original MDD

\[ \min x_1 - 2x_2 + 3x_3 \]

8 solutions
MDDs as Relaxations

\[ \min x_1 - 2x_2 + 3x_3 \]

Optimal value = 2
(Shortest path length)

Original MDD

8 solutions
Tightening the MDD Relaxation

$$\min x_1 - 2x_2 + 3x_3$$

Relaxation of width 2

14 solutions
Tightening the MDD Relaxation

\[ \min x_1 - 2x_2 + 3x_3 \]

Shortest path length = −1

We would like a tighter bound (closer to 2)
Tightening the MDD Relaxation

\[
\min x_1 - 2x_2 + 3x_3
\]

Shortest path length = −1

We would like a tighter bound (closer to 2)

Compute possible path lengths at each node.
Tightening the MDD Relaxation

\[
\min x_1 - 2x_2 + 3x_3
\]

Shortest path length = -1

We would like a tighter bound (closer to 2)

Compute possible path lengths at each node.

Construct MDD of paths with length -1

Relaxation of width 2
Tightening the MDD Relaxation

\[ \text{min } x_1 - 2x_2 + 3x_3 \]

Shortest path length = −1

We would like a tighter bound (closer to 2)

Compute possible path lengths at each node.

Construct MDD of paths with length −1

This path is infeasible, so we have new upper bound of 1. Continue.
Pseudocosts

\[ \min x_1 - 2x_2 + 3x_3 \]

Shortest path length = \(-1\)

**Pseudocost** = effect on optimal value of relaxation of fixing a variable.

Relaxation of width 2

14 solutions
min $x_1 - 2x_2 + 3x_3$

Shortest path length = −1

**Pseudocost** = effect on optimal value of relaxation of fixing a variable.

Branch on $x_1 = 1$.
Optimal value increases by 3 (−1 to 2).
Pseudocosts

\[
\min x_1 - 2x_2 + 3x_3
\]

Shortest path length = −1

**Pseudocost** = effect on optimal value of relaxation of fixing a variable.

Branch on \( x_1 = 1 \).
Optimal value increases by 3 (−1 to 2).

Branch on \( x_1 = 2 \).
Optimal value increases by 0.
Pseudocosts

\[ \text{min } x_1 - 2x_2 + 3x_3 \]

Shortest path length = −1

**Pseudocost** = effect on optimal value of relaxation of fixing a variable.

Branch on \( x_1 = 1 \).
Optimal value increases by 3 (-1 to 2).

Branch on \( x_1 = 2 \).
Optimal values increases by 0.

Branch on \( x_1 = 3 \).
Optimal value increases by 3
Pseudocosts

\[ \min x_1 - 2x_2 + 3x_3 \]

Shortest path length = −1

**Pseudocost** = effect on optimal value of relaxation of fixing a variable.

Branch on \( x_1 = 1 \).
Optimal value increases by 3
(−1 to 2).

Branch on \( x_1 = 2 \).
Optimal values increases by 0.

Branch on \( x_1 = 3 \).
Optimal value increases by 3
MDDs as Restrictions
MDDs as Restrictions

- A **restricted** MDD represents a subset of the feasible solutions.

- Restricted MDDs provide a basis for a **feasibility heuristic** in optimization problems.
  - Shortest paths in the restricted MDD provide good feasible solutions.
MDDs as Restrictions

- To generate a restriction of limited width:
  - Create relaxed MDD for **negation** of the constraint and **complement** the MDD
  - Complementation adds at most one to width of the MDD

- Or use a **constraint-specific** algorithm.
To generate a restricted MDD for the set covering problem:

\[ x_1 + x_2 + x_3 \geq 1 \]
\[ x_1 + x_4 + x_5 \geq 1 \]
\[ x_2 + x_4 + x_6 \geq 1 \]

Find collection of sets that cover elements A, B, C:

Sets

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

52 feasible solutions.

Minimum cover of 2, e.g. \( x_1, x_2 \)
To generate a restricted MDD for the set covering problem

\[ x_1 + x_2 + x_3 \geq 1 \]
\[ x_1 + x_4 + x_5 \geq 1 \]
\[ x_2 + x_4 + x_6 \geq 1 \]

Build a relaxed MDD for its negation

\( (x_1 + x_2 + x_3 < 1) \)
\( \lor (x_1 + x_4 + x_5 < 1) \)
\( \lor (x_2 + x_4 + x_6 < 1) \)

using method described earlier.

Find collection of sets that cover elements A, B, C

<table>
<thead>
<tr>
<th>Sets</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

52 feasible solutions.

Minimum cover of 2, e.g. \( x_1, x_2 \)
To generate a restricted MDD for the set covering problem

\[
\begin{align*}
    x_1 + x_2 + x_3 & \geq 1 \\
    x_1 + x_4 + x_5 & \geq 1 \\
    x_2 + x_4 + x_6 & \geq 1
\end{align*}
\]

Build a relaxed MDD for its negation

\[
\begin{align*}
    (x_1 + x_2 + x_3 < 1) & \lor (x_1 + x_4 + x_5 < 1) & \lor (x_2 + x_4 + x_6 < 1)
\end{align*}
\]

using method described earlier.
This is a restricted MDD for the set covering problem.
41 feasible solutions (< 52)
Complement of relaxed MDD
Width 4

Several shortest paths have length 2.
All are minimum covers.

This is a restricted MDD for the set covering problem.
41 feasible solutions (< 52)
Complement of relaxed MDD
Width 4

Several shortest paths have length 2.
All are minimum covers.
In this case, feasibility heuristic delivers optimal solutions.

This is a restricted MDD for the set covering problem.
41 feasible solutions (< 52)
MDDs as Transparent Data Structures
Transparent Data Structure

- An alternate paradigm for problem solving:
  - Rather than generate a solution, transform the problem to a more transparent data structure.
  - MDDs can serve this purpose.
Transparent Data Structure

- An alternate paradigm for problem solving:
  - Rather than generate a solution, transform the problem to a more transparent data structure.
  - MDDs can serve this purpose.

- In particular, use MDDs for postoptimality analysis / explanation.
  - Rapid processing of queries.
Postoptimality Analysis

- Reliability networks
  - Minimize cost subject to a bound on reliability
  - System of 5 bridges:

\[
R = R_1 R_2 + (1 - R_2) R_3 R_4 + (1 - R_1) R_2 R_3 R_4 + R_1 (1 - R_2) (1 - R_3) R_4 R_5 + (1 - R_1) R_2 R_3 (1 - R_4) R_5
\]
The problem:

\[ \min \sum_j c_j x_j \]

Number of links \( j \)

\[ R \geq R_{\text{min}} \]

\[ R = R_1 R_2 + (1 - R_2) R_3 R_4 + (1 - R_1) R_2 R_3 R_4 \]
\[ + R_1 (1 - R_2)(1 - R_3) R_4 R_5 + (1 - R_1) R_2 R_3 (1 - R_4) R_5 \]

\[ R_j = 1 - (1 - r_j)^{x_j}, \text{ all } j \]

\[ x_j \in \{0,1,2,3\} \]

Set \( R_{\text{min}} = 60 \) in all examples
Cost-based domain analysis

89 nodes in MDD
1.2 seconds to compile MDD

\[ r = (0.9, 0.85, 0.8, 0.9, 0.95) \]
\[ c = (25, 35, 40, 10, 60) \]

<table>
<thead>
<tr>
<th>( c_{opt} + \Delta )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>50: 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>60: 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.2</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>85: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>90: 2</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>95: 2</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>100:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>95</td>
</tr>
<tr>
<td>120:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>125: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>155: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>98</td>
</tr>
<tr>
<td>170:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>180:</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>230:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
7 bridges
Cost-based domain analysis

3126 nodes in MDD
9.0 seconds to compile MDD

<table>
<thead>
<tr>
<th>$c_{opt} + \Delta$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>72.2</td>
</tr>
<tr>
<td>11:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12:</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>82.9</td>
</tr>
<tr>
<td>14:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>15:</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>84.6</td>
</tr>
<tr>
<td>17:</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>95.2</td>
</tr>
<tr>
<td>18:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>19:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>20:</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>97.2</td>
</tr>
<tr>
<td>23:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>27:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.2</td>
</tr>
<tr>
<td>34:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.4</td>
</tr>
<tr>
<td>40:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.6</td>
</tr>
<tr>
<td>43:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.7</td>
</tr>
<tr>
<td>47:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.8</td>
</tr>
<tr>
<td>54:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.9</td>
</tr>
</tbody>
</table>
12 bridges
Cost-based domain analysis

36,301 nodes in MDD
1980 seconds to compile MDD
1.8 seconds query time to build this table

<table>
<thead>
<tr>
<th>$c_{opt} + \Delta$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
<th>$x_{10}$</th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>185</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82</td>
</tr>
<tr>
<td>190</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>83</td>
</tr>
<tr>
<td>195</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>86</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>88</td>
</tr>
<tr>
<td>205</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>220</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>225</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>230</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>235</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>245</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>265</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>270</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>275</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>280</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>305</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>310</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>315</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>320</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>340</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>360</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>365</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>380</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>430</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>485</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cost-Bounded MDDs
Cost-Bounded MDDs

• **Original MDD**
  – Represents entire feasible set
  – Can be very large.
Cost-Bounded MDDs

• **Original** MDD
  – Represents entire feasible set
  – Can be very large.

• **Near-optimal** MDD
  – Exactly represents solutions within a given distance from optimal value.
  – Can be even larger than the original MDD!
Cost-Bounded MDDs

• **Original** MDD
  – Represents entire feasible set
  – Can be very large.

• **Near-optimal** MDD
  – Exactly represents solutions within a given distance from optimal value.
  – Can be even larger than the original MDD!

• **Compressed** MDD
  – Includes all near-optimal solutions, plus some others.
  – Much smaller than near-optimal MDD.
  – Constructed by pruning and contraction.
Cost-Bounded MDDs

All feasible solutions

Large MDD
Cost-Bounded MDDs

- All feasible solutions
- Near-optimal solutions
- Possibly even larger MDD
Cost-Bounded MDDs

- All feasible solutions
- Simplification of near-optimal set
- Near-optimal solutions
- Compressed MDD
Pruning and Contracting

- Heuristic methods for generating **small compressed** MDDs when conjoining constraints:
  - Pruning edges
  - Contracting nodes
Pruning

- Delete all edges that belong only to paths longer than optimal value + tolerance.
Pruning

- Delete all edges that belong only to paths longer than optimal value + tolerance.
Pruning

- Delete all edges that belong only to paths longer than optimal value + tolerance.
Pruning

- Delete all edges that belong only to paths longer than optimal value + tolerance.

Delete it, too.
Pruning

- Delete all edges that belong only to paths longer than optimal value + tolerance.

And simplify the BDD.
Pruning

- Delete all edges that belong only to paths longer than optimal value + tolerance.

And simplify the BDD.
Contracting

- Remove a node if this creates no new paths shorter than optimal value + tolerance.
Experimental Results

- Solve the 0-1 problem

\[
\min cx \\
Ax \geq b \\
x \in \{0,1\}^n
\]

\(A_{ij}\) drawn uniformly from \([0,r]\)
20 variables, 5 constraints

20-5-50-3 $c_{\text{min}}=101$, $c_{\text{max}}=588$

Max value of objective function

Size of MDD

Exact near-optimal MDD

Compressed MDD

Best solution

Worst solution

16,285 nodes

8566 nodes

Max value of objective function
20 variables, 5 constraints

Max value of objective function

Size of MDD

Exact near-optimal MDD

Compressed MDD

Best solution

Worst solution

At tolerance = 80%, only 524 nodes in contracted MDD

16,285 nodes

8566 nodes

141 218 296 393 491 588

Max value of objective function
Max value of objective function

Size of MDD

Exact near-optimal MDD

Compressed MDD

Best solution

Worst solution

556 solutions

72,896 solutions

449,102 solutions

929,260 solutions

20-5-50-3 c_{min}=101, c_{max}=588
Experimental results

- MDD representation of all **optimal** solutions
- MIPLIB instances

<table>
<thead>
<tr>
<th>MIPLIB instance</th>
<th>Size of exact MDD for all feasible solutions</th>
<th>Size of exact MDD for all optimal solutions</th>
<th>Size of compressed MDD for all optimal solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>lseu</td>
<td>?</td>
<td>99</td>
<td>19</td>
</tr>
<tr>
<td>p0033</td>
<td>375</td>
<td>41</td>
<td>21</td>
</tr>
<tr>
<td>p0201</td>
<td>310,420</td>
<td>737</td>
<td>84</td>
</tr>
<tr>
<td>stein27</td>
<td>25,202</td>
<td>6260</td>
<td>4882</td>
</tr>
<tr>
<td>stein45</td>
<td>5,102,257</td>
<td>1765</td>
<td>1176</td>
</tr>
</tbody>
</table>
Nonserial MDDs and Dynamic Programming
Nonserial Dynamic Programming

- Independently(?) rediscovered many times:
  - Nonserial DP (1972)
  - Constraint satisfaction (1981)
  - Database queries (1983)
  - $k$-trees (1985)
  - Belief logics (1986)
  - Bucket elimination (1987)
  - Bayesian networks (1988)
  - Pseudoboolean optimization (1990)
  - Location analysis (1994)
Set Partitioning example

Find collection of sets that partition elements A, B, C, D

<table>
<thead>
<tr>
<th>Sets</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>
Set Partitioning example

Find collection of sets that partition elements A, B, C, D

Sets

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

For example…
Set Partitioning example

Find collection of sets that partition elements A, B, C, D

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Or...
Set Partitioning example

Find collection of sets that partition elements A, B, C, D

Sets

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

0-1 formulation

\[
\begin{align*}
    x_1 + x_2 + x_3 &= 1 \\
    x_2 + x_4 &= 1 \\
    x_3 + x_5 + x_6 &= 1 \\
    x_4 + x_6 &= 1
\end{align*}
\]

\[x_j = 1 \quad \Rightarrow \quad \text{set } j \text{ selected}\]
Set Partitioning example

Dependency graph

\[ x_1 \quad x_3 \quad x_5 \]
\[ x_2 \quad x_4 \quad x_6 \]

0-1 formulation

\[
\begin{align*}
x_1 + x_2 + x_3 &= 1 \\
x_2 + x_4 &= 1 \\
x_3 + x_5 + x_6 &= 1 \\
x_4 + x_6 &= 1
\end{align*}
\]

\[ x_j = 1 \quad \Rightarrow \quad \text{set } j \text{ selected} \]
Set Partitioning example

Dependency graph

Enumeration order

\[x_2\]
\[x_3\]
\[x_4\]
\[x_1\]
\[x_5\]
\[x_6\]
Set Partitioning example

Dependency graph

\[ x_2 \rightarrow x_3 \rightarrow x_5 \]
\[ x_1 \rightarrow x_3 \rightarrow x_5 \]
\[ x_2 \rightarrow x_4 \rightarrow x_6 \]
\[ x_1 \rightarrow x_4 \rightarrow x_6 \]

Enumeration order

\[ x_2, x_3, x_4, x_1, x_5, x_6 \]
Set Partitioning example

Dependency graph

x₁ ← x₂ → x₃ ← x₄ → x₅ ← x₆

Enumeration order

x₂
x₃
x₄
x₅
x₁
x₆
Set Partitioning example

Dependency graph

$x_1 \rightarrow x_2 \rightarrow x_4 \rightarrow x_6$

$x_1 \rightarrow x_3 \rightarrow x_5$

$x_3 \rightarrow x_4 \rightarrow x_5$

x2

Enumeration order

$x_2$

x3

x4

x1

x5

x6
Set Partitioning example

Dependency graph

Enumeration order

x₁ → x₃ → x₅
x₂ → x₄ → x₆

x₂ → x₃ → x₄ → x₁
x₃ → x₄ → x₁
x₄ → x₁
x₁ → x₅ → x₆
x₅ → x₆
x₆
Set Partitioning example

Dependency graph

$x_1 \leftrightarrow x_3 \rightarrow x_5$
$x_2 \rightarrow x_4 \rightarrow x_6$

Enumeration order

$x_2 \xrightarrow{} x_3 \xrightarrow{} x_4 \xrightarrow{} x_5 \xrightarrow{} x_6 \xrightarrow{} x_1$
Set Partitioning example

Dependency graph

Enumeration order
Set Partitioning example

Dependency graph

\[ x_1 \leftarrow x_3 \rightarrow x_5 \]
\[ x_2 \rightarrow x_4 \rightarrow x_6 \]

Induced width = 3 (max in-degree)

Enumeration order

\[ x_2 \quad x_3 \quad x_4 \quad x_1 \quad x_5 \quad x_6 \]
Set Partitioning example

Solution by nonserial DP

Enumeration order
Set Partitioning example

Solution by nonserial DP

Sets

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Set Partitioning example

Feasible solution

Sets

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

x_2

x_2x_3

x_3x_4

x_1

x_3x_4x_5

x_6
Set Partitioning example

**Feasible solution**

<table>
<thead>
<tr>
<th>Sets</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

**Sets**

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Variables**

- $x_2$
- $x_2x_3$
- $x_3x_4$
- $x_1$
- $x_3x_4x_5$
- $x_6$
Set Partitioning example

Feasible solution

<table>
<thead>
<tr>
<th>Sets</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x_1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Set Partitioning example

Solution by nonserial DP

Serialized DP
Set Partitioning example

Feasible solutions

\[ x_2 \]
\[ x_2 x_3 \]
\[ x_3 x_4 \]
\[ x_1 \]
\[ x_3 x_4 x_5 \]
\[ x_6 \]

Feasible solutions

\[ x_2 \]
\[ x_2 x_3 \]
\[ x_3 x_3 x_4 \]
\[ x_1 x_2 x_3 x_4 \]
\[ x_3 x_4 x_5 \]
\[ x_6 \]
BDD vs. DP Solution

BDD

Serialized DP
BDD vs. DP Solution

BDD

\[ x_2 \]
\[ x_2 x_3 \]
\[ x_3 x_4 \]
\[ x_1 \]
\[ x_3 x_4 x_5 \]
\[ x_6 \]

BDD

\[ x_2 \]
\[ x_2 x_3 \]
\[ x_3 x_4 \]
\[ x_1 \]
\[ x_3 x_4 x_5 \]
\[ x_6 \]

Serialized DP

\[ x_2 \]
\[ x_2 x_3 \]
\[ x_3 x_3 x_4 \]
\[ x_1 x_2 x_3 x_4 \]
\[ x_3 x_4 x_5 \]
\[ x_6 \]

Deleted

010 011 110 000 001
BDD vs. DP Solution

BDD

Serialized DP

Merged
Set Partitioning example

Solution by nonserial DP

\( x_2 \)

\( x_2 x_3 \)

\( x_3 x_4 \)

\( x_1 \)

\( x_3 x_4 x_5 \)

\( x_6 \)
Set Partitioning example

Solution by nonserial DP

Nonserial BDD
Constructing the Join Tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

Clique in the dependency graph

$x_1x_2x_3$
Constructing the Join Tree

\[ x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6 \]

Clique in the dependency graph

\[ x_1 x_2 x_3 \]

\[ x_2 x_3 x_4 \]
Constructing the Join Tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

Clique in the dependency graph

$x_1 x_2 x_3$

$x_2 x_3 x_4$

$x_3 x_4 x_5 x_6$
Constructing the Join Tree

\[ x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6 \]

Dependency graph

Join graph

Connect nodes with common variables
Constructing the Join Tree

\[ x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6 \]

Dependency graph

Join graph

\[ x_j \text{ occurs along every path connecting } x_j \text{ with } x_j \]
Constructing the Join Tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

Dependency graph

Join graph

This can be viewed as the **constraint dual**

Binary constraints equate common variables in subproblems
Constructing the Join Tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

Some edges may be redundant when equating variables
Constructing the Join Tree

$$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$$

Dependency graph

Join tree

Removing redundant edges yields **join tree**
Constructing the Join Tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

Dependency graph

Join tree

$X_1 X_2 X_3$
$X_2 X_3 X_4$
$X_3 X_4$
$X_3 X_4 X_5 X_6$

Max node cardinality is $\text{tree width} + 1 = 3 + 1$
Constructing the Join Tree

\[ x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6 \]

Dependency graph

Join tree

\[ x_1x_2x_3 \]
\[ x_2x_3 \]
\[ x_2x_3x_4 \]
\[ x_3x_4 \]
\[ x_3x_4x_5x_6 \]

Induced width = tree width = 3
Designing the Nonserial BDD

BDD design

$x_2$

Join tree

$x_2x_3x_1$

$x_2x_3$

$x_2x_3x_4$

$x_3x_4$

$x_3x_4x_5x_6$
Designing the Nonserial BDD

BDD design

Join tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

$x_2 \ x_3 \ x_1$

$x_2 x_3 x_4$

$x_3 x_4$

$x_3 x_4 x_5 x_6$
Designing the Nonserial BDD

\[ x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6 \]

BDD design

Join tree

\[ x_2x_3x_1 \]
\[ x_2x_3x_4 \]
\[ x_3x_4 \]
\[ x_3x_4x_5x_6 \]
Designing the Nonserial BDD

BDD design

Join tree

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$
Designing the Nonserial BDD

$x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6$

BDD design

Join tree
Designing the Nonserial BDD

BDD design

\[ x_2 \ x_3 \ x_4 \ x_1 \ x_5 \ x_6 \]

Join tree
Designing the Nonserial BDD

BDD design

$X_2 X_3 X_4 X_1 X_5 X_6$

Nonserial BDD
Another Variable Ordering

\[ x_3 \ x_6 \ x_2 \ x_5 \ x_1 \ x_4 \]

Dependency graph

Join graph

Induced width = 2
Constructing the Join Tree

\[ x_3 \ x_6 \ x_2 \ x_5 \ x_1 \ x_4 \]

**Dependency graph**

**Join tree**

Induced width = 2

Tree width = 2
Designing the BDD

$x_3 \ x_6 \ x_2 \ x_5 \ x_1 \ x_4$

BDD design

$x_3$

Join tree

$x_3x_2x_1$

$x_3x_6$

$x_3x_6x_5$

$x_3x_6x_2$

$x_6x_2$

$x_6x_2x_4$

Tree width = 2
Designing the BDD

BDD design

\[ x_3 \quad x_6 \quad x_2 \quad x_5 \quad x_1 \quad x_4 \]

Join tree

Tree width = 2
Designing the BDD

$X_3 \, X_6 \, X_2 \, X_5 \, X_1 \, X_4$

BDD design

Join tree

Tree width = 2
Designing the BDD

BDD design

Tree width = 2

Join tree

$x_3 \ x_6 \ x_2 \ x_5 \ x_1 \ x_4$
Nonserial BDD

BDD design

\[
x_3, x_6, x_2, x_5, x_1, x_4
\]
Constraints with DP Structure

- Constraints with serial DP structure
  - Stretch
  - Regular
  - Sequence
- Constraints with nonserial DP structure…