Cost-Bounded Binary Decision Diagrams for 0-1 Programming

Tarik Hadzic
IT University of Copenhagen

John Hooker
Carnegie Mellon University

INFORMS 2007, Seattle

Motivation

- Perform **postoptimality analysis** for 0-1 programming.
- **Problem**: It is hard to reason about the entire solution space.
- **Solution**: Represent the set of near-optimal solutions as a Binary Decision Diagram.
  - This was done in previous work (H&H 2006).

Motivation

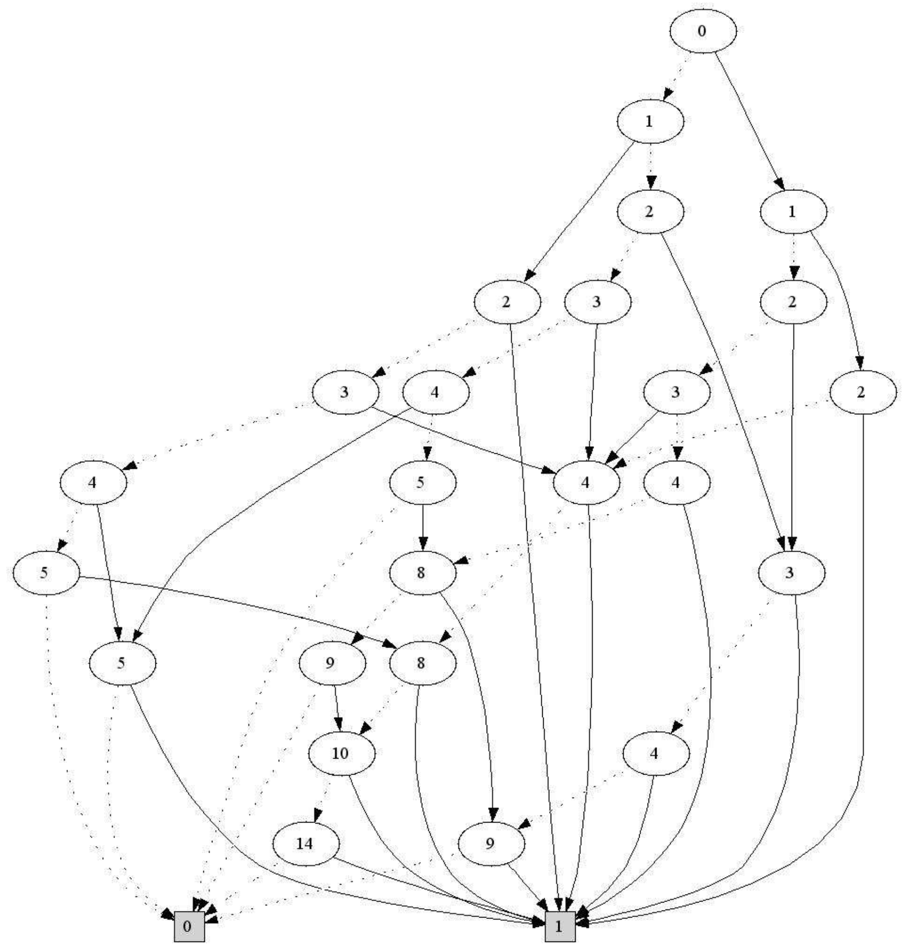
- Today's focus: **Scalability**
  - How large do BDDs grow with problem size?
  - How can we minimize the growth?
- We introduce **sound** BDDs.
  - Much **smaller** than the full BDD.
  - Provide **exact** postoptimality analysis for near-optimal solutions.

Types of Analysis

- Characterization of optimal or near-optimal solutions.
  - How much freedom is there to alter solution without much sacrifice in solution quality?
- Sensitivity analysis.
  - Which problem data significantly affect the solution?
- Online what-if queries.
  - What if I fix certain variables?

Basic Idea

- Use **reduced ordered binary decision diagrams** (BDDs) as a compact representation of the set of feasible or near-optimal solutions.
  - We can extract information from BDDs in real time.
  - Although exponentially large in the worst case, BDDs can be compact for important constraints.

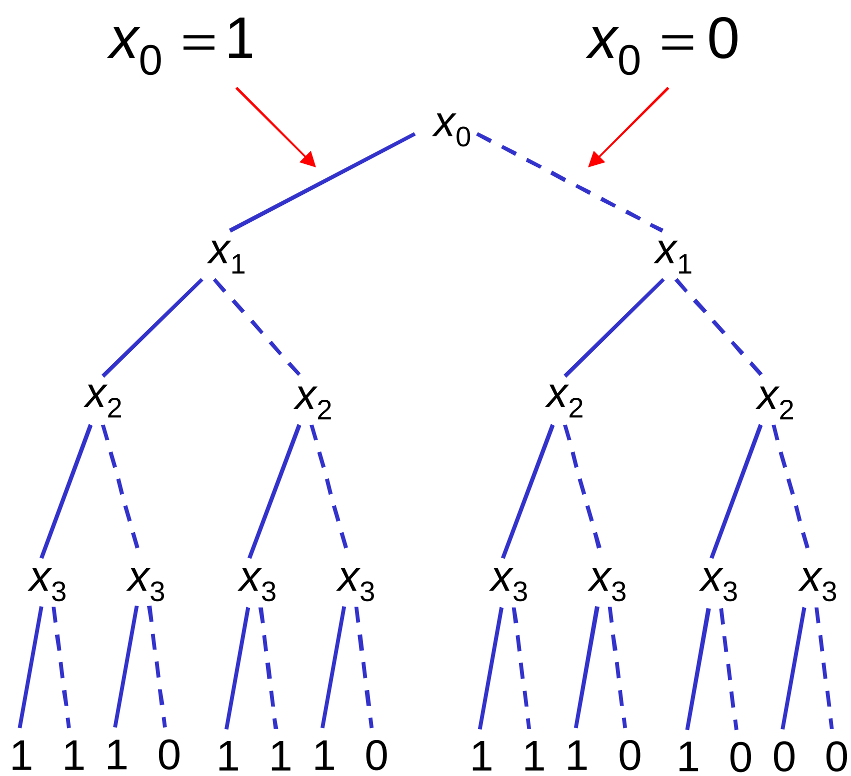


Binary Decision Diagrams

- A reduced ordered BDD for a constraint set is a **compact representation of the branching tree** for a given branching order.
  - If both branches from a node lead to identical subtrees, remove the node.
  - If two subtrees are identical, superimpose them.

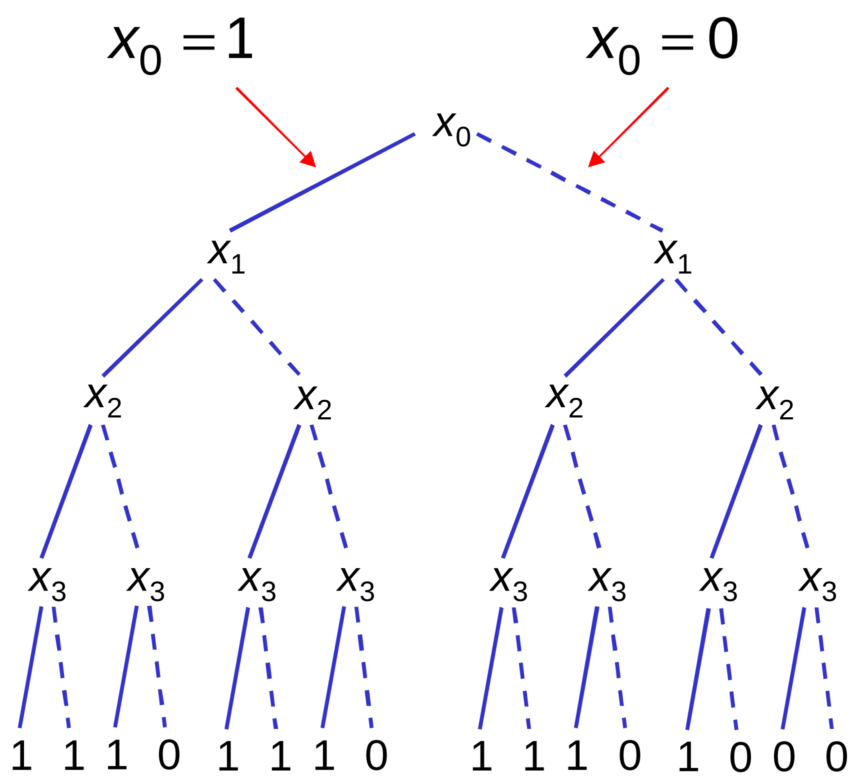
Branching tree for 0-1 inequality

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

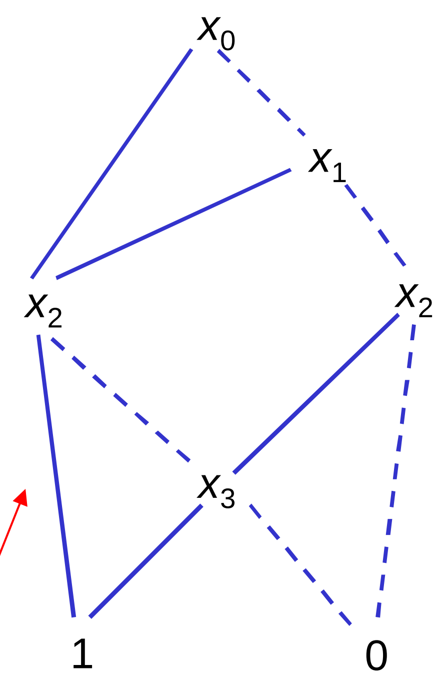


Branching tree for 0-1 inequality

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



Reduced ordered BDD

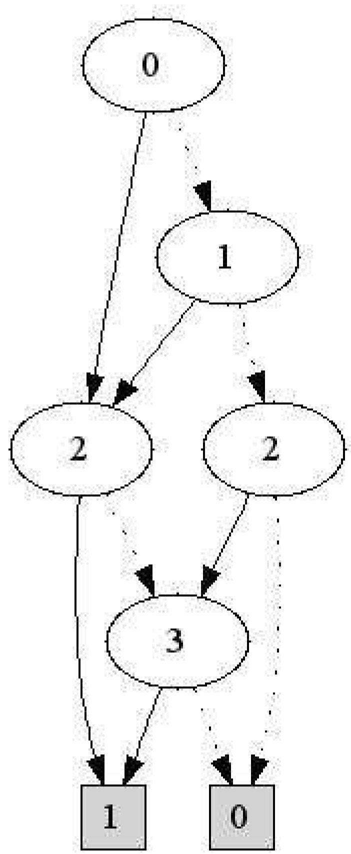


1

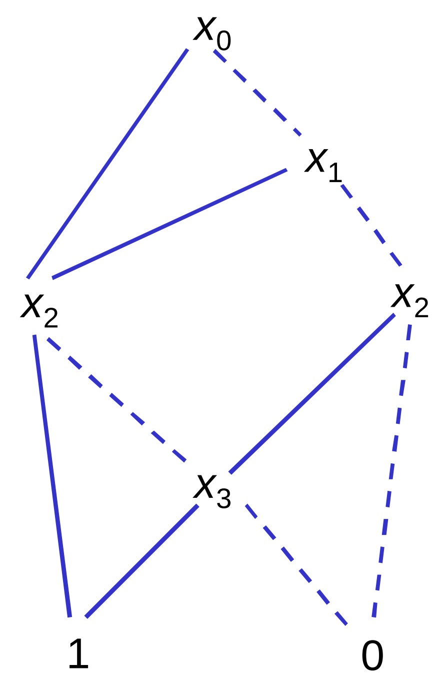
If an edge skips a variable, both assignments are allowed ($x_3 = 0$ and $x_3 = 1$)

In practice, BDDs are generated **bottom-up**.

First construct a BDD for every constraint and then conjoin the BDDs.



as generated by software
(CLab, BuDDy)



- The BDD for a knapsack constraint can be surprisingly small…

The 0-1 inequality

$$300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6 + 230x_7 + 190x_8 + 200x_9 + 400x_{10} + 200x_{11} + 400x_{12} + 200x_{13} + 400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \geq 2701$$

has 117,520 minimal feasible solutions

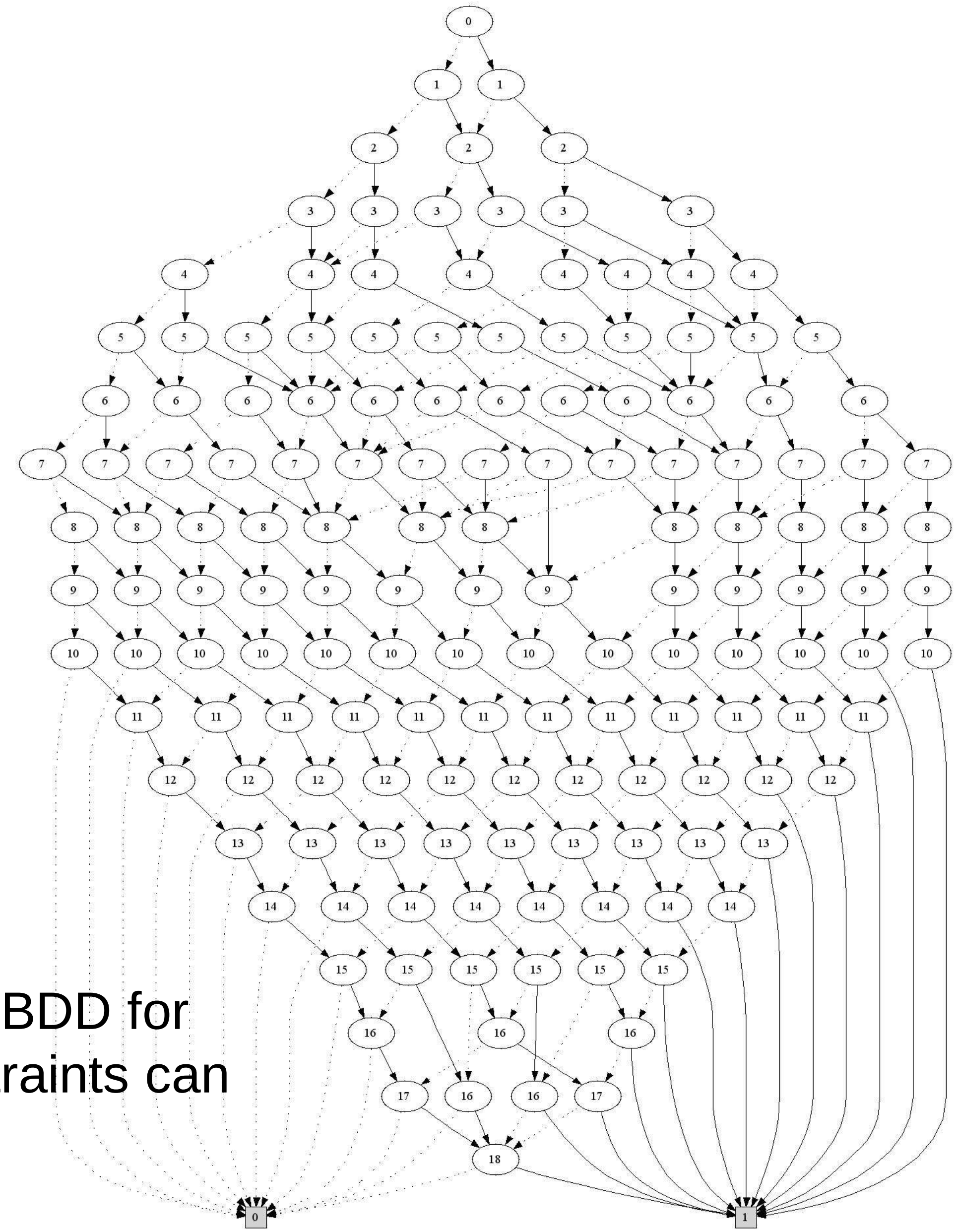
Or equivalently,

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

has 117,520 minimal covers

But its reduced BDD has only 152 nodes…

* However, the BDD for multiple constraints can explode.



Optimization Over BDDs

- We want to solve a 0-1 programming model with an additively separable objective function

$$\min \sum_{j=1}^{n} c_j(x_j)$$
$$g_i(x) \ge b_i, \quad i = 1,\ldots,m$$
$$x_j \in \{0,1\}, \quad j = 1,\ldots,n$$

Can be straightforwardly extended to general integer programming

Optimization Over BDDs

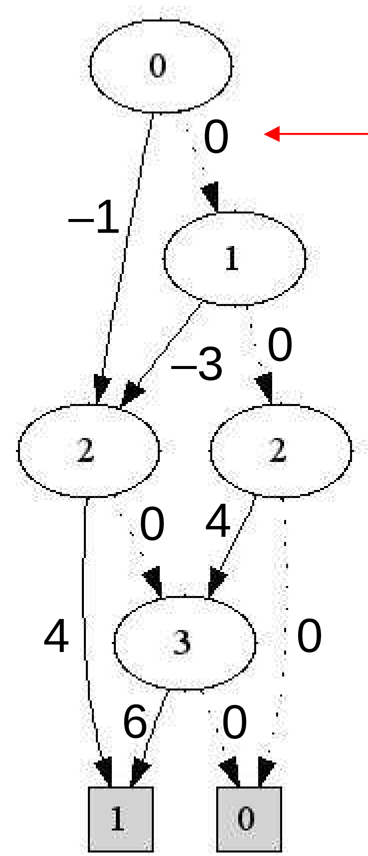
- We want to solve a 0-1 programming model with an additively separable objective function

$$\begin{aligned} &\min \sum_{j=1}^{n} c_j(x_j) \\ &g_i(x) \ge b_i, \quad i = 1,\ldots,m \\ &x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned}$$

Can be straightforwardly extended to general integer programming

- If we represent the constraints $g_i(x)$ **≥** b_i as a BDD, then we can solve the problem by **finding a shortest path** in the BDD with appropriate edge lengths…

$$\min 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to} \quad 2x_0+3x_1+5x_2+5x_3 \geq 7$$

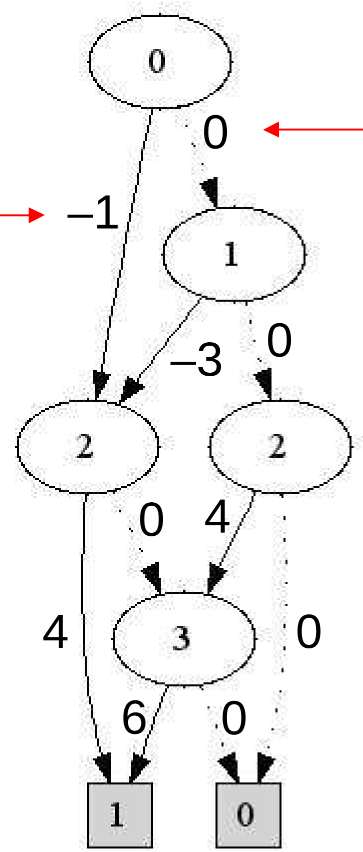


$2 \cdot 0$

Edge lengths reflect coefficients in the objective function

$$\min 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to} \quad 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$

$2 \cdot 1 + \min_{x_1 \in \{0,1\}} \{-3x_1\}$



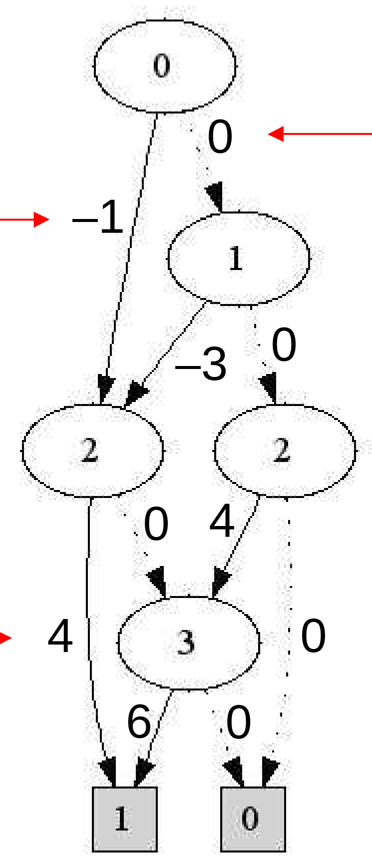
$2 \cdot 0$

Edge lengths reflect coefficients in the objective function

$$\min 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to} \quad 2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$

$$2 \cdot 0$$

$$2 \cdot 1 + \min_{x_1 \in \{0,1\}} \{-3x_1\}$$



$$4 \cdot 1 + \min_{x_3 \in \{0,1\}} \{6x_3\}$$

Edge lengths reflect coefficients in the objective function

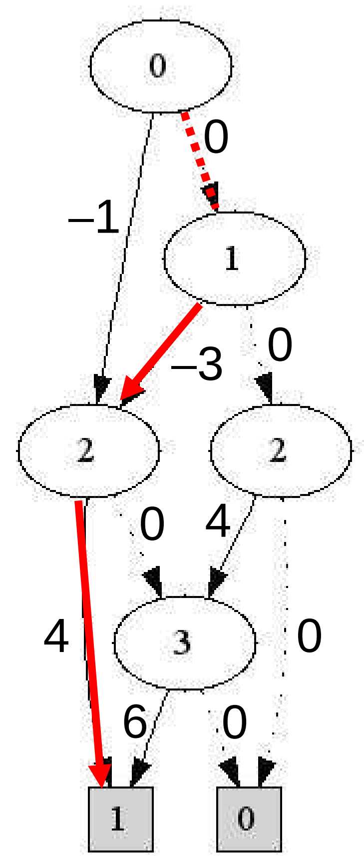
$$\min\ 2x_0 - 3x_1 + 4x_2 + 6x_3 \quad \text{subject to} \quad 2x_0+3x_1+5x_2+5x_3 \geq 7$$

Shortest path has length 1

Optimal solution:

$$(x_0, x_1, x_2, x_3) = (0,1,1,0)$$

Set to minimizing value



We conduct postoptimality analysis by analyzing shortest and near-shortest paths

Cost-Based Domain Analysis

- Consider again

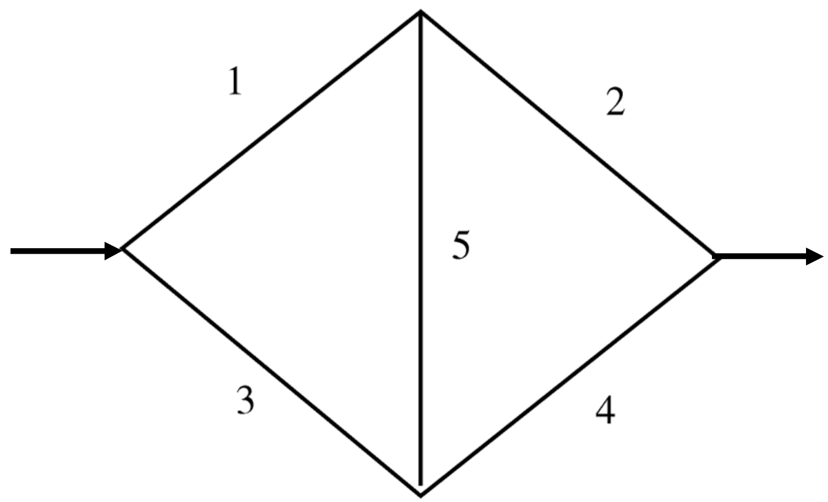
$$\begin{aligned}&\min \sum_{j=1}^{n} c_j(x_j)\\ &g_i(x) \geq b_i, \quad i = 1,\ldots,m\\ &x_j \in \{0,1\}, \quad j = 1,\ldots,n\end{aligned}$$

- What values can x_j take without forcing the objective function value above $\boxed{c_{\text{opt}}} + \Delta$?

Optimal value

Example: Network Reliability

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



$$R = R_1R_2 + (1-R_2)R_3R_4 + (1-R_1)R_2R_3R_4 + R_1(1-R_2)(1-R_3)R_4R_5 + (1-R_1)R_2R_3(1-R_4)R_5$$

The problem:

$$\min \sum_j c_j x_j$$

Number of links j

$$R \geq R_{\min}$$

$$R = R_1R_2 + (1-R_2)R_3R_4 + (1-R_1)R_2R_3R_4$$
$$+ R_1(1-R_2)(1-R_3)R_4R_5 + (1-R_1)R_2R_3(1-R_4)R_5$$

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

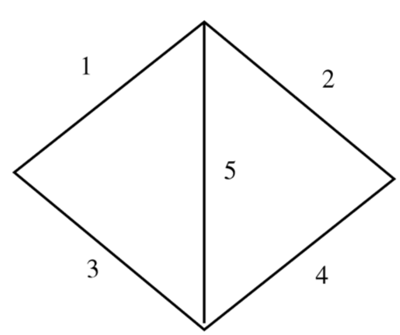
$$x_j \in \{0,1,2,3\}$$

Set $R_{\min}$ = 60 in all examples

Cost-based domain analysis

308 nodes in BDD

1.1 seconds
to compile BDD



$r = (0.9, 0.85, 0.8, 0.9, 0.95)$

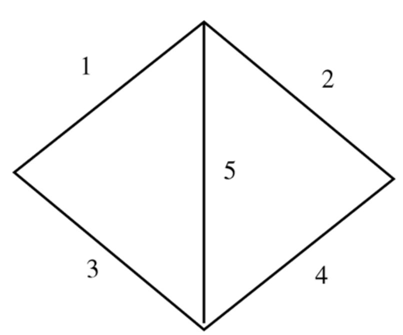
$c = (25, 35, 40, 10, 60)$

$c_{opt} + \Delta$	x_1	x_2	x_3	x_4	x_5	R
50:	0	0	1	1	0	72
60:	1	1	0	0,2		79
85:	2					84
90:			2	3		86
95:		2			1	88
100:						95
120:						97
125:	3					
155:		3			2	
160:						98
170:						99
180:			3			
230:					3	

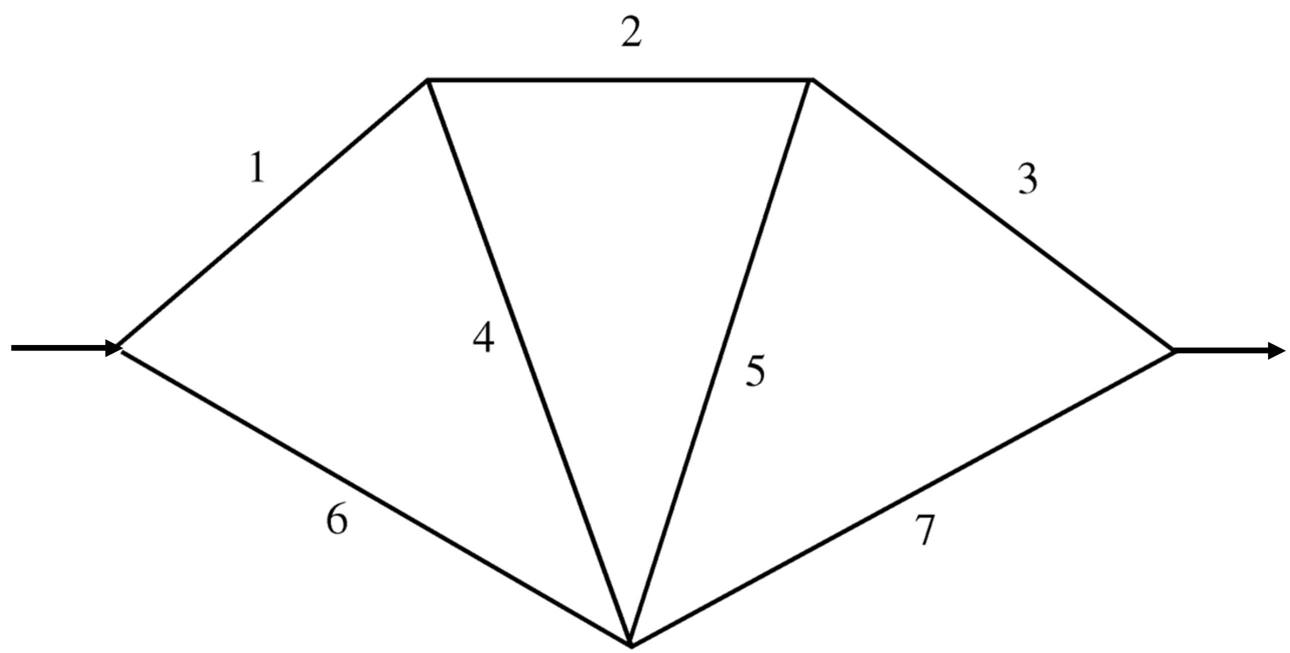
Domain analysis with respect to *R*

R	x_1	x_2	x_3	x_4	x_5
99:	1,2	1,2	1,2	1,2	0,1,2,3
98:		0,3	0,3		
97:				0,3	
95:	0,3				

Same BDD as before



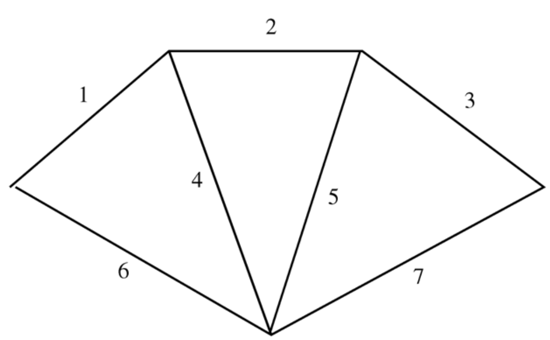
7 bridges



Cost-based domain analysis

1779 nodes in BDD

14.8 seconds
to compile BDD

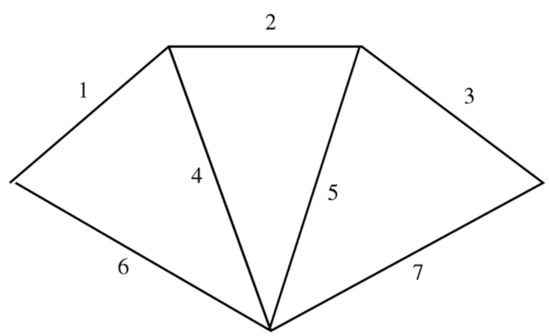


$c_{opt} + \Delta$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	R
9:	0	0	0	0	0	1	1	72.2
11:			1		1		0	
12:	1			1		0		
13:		1				2		82.9
14:					2		2	
15:			2	2				
16:	2							
17:					3	3		84.6
18:		2		3				95.2
19:			3				3	
20:	3							
22:								97.2
23:		3						
27:								99.2
34:								99.4
40:								99.6
43:								99.7
47:								99.8
54:								99.9

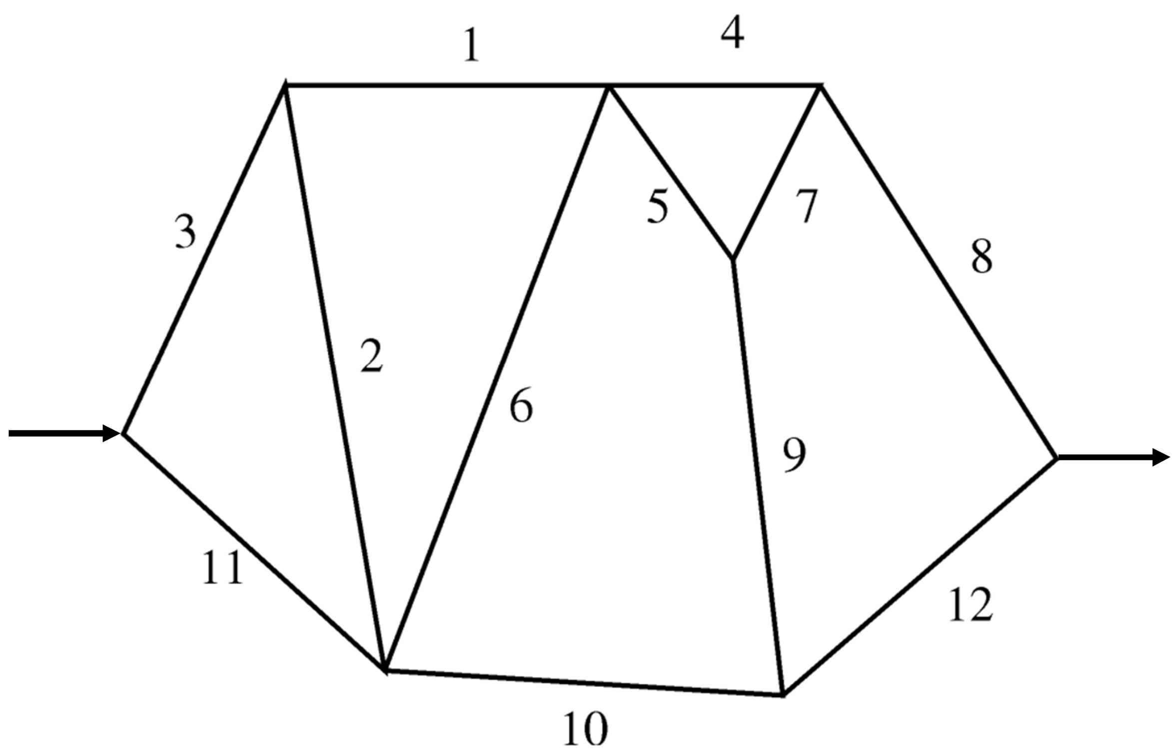
Domain analysis with respect to *R*

R	x_1	x_2	x_3	x_4	x_5	x_6	x_7
99.9	2,3	0,1,2,3	1,2,3	0,1,2,3	0,1,2,3	2,3	2,3
99.8	1						1
99.5	0		0			1	
99.1							0
97.2						0	

Same BDD as before



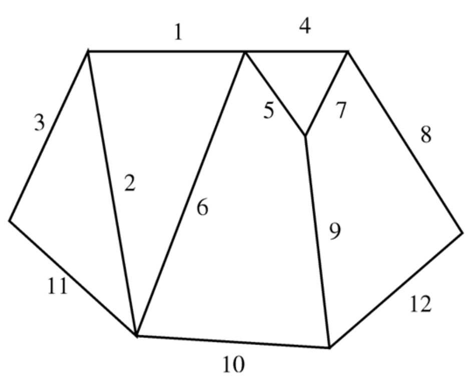
12 bridges



Cost-based domain analysis

69,457 nodes in BDD

2933 seconds
to compile BDD

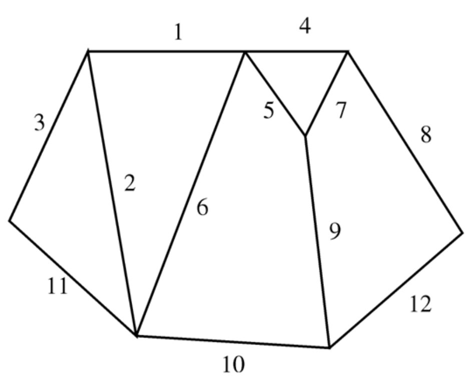


$c_{opt} + \Delta$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	R
180	1	0	2	3	0	0	0	2	0	0	0	0	80
185			3	2									82
190								3					
195													83
200										1			
205													86
210	2						1						
215													88
220										2			
225					1				1				
230	0		0			1,2					1		
235			1										
240		1		1			2			3	2	1	
250				0				0,1				2	
255													91
260	3								2				
265													93
270					2	3	3						
290											3		
300		2											
305									3				
310												3	
315					3								94
340													95
360		3											
365													96
380													97
430													98
485													99

Domain analysis with respect to R

R	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
99:	0..3	0..3	1..3	0..3	0..3	0..3	0..3	1..3	0..3	0..3	1..3	1..3
98:			0					0				0
96:											0	

Same BDD as before



Reducing BDD Growth

- It suffices to use a BDD with the same near-optimal solutions as the original BDD. We assume $\Delta \leq \Delta_{max}$.

$B_{\Delta_{max}}$

Reducing BDD Growth

- It suffices to use a BDD with the same near-optimal solutions as the original BDD. We assume $\Delta \le \Delta_{max}$.

$$Sol = \left\{ \begin{matrix} \text{feasible} \\ \text{solutions} \end{matrix} \right\} \qquad B = \begin{matrix} \text{original} \\ \text{BDD} \end{matrix}$$

$$Sol_{\Delta_{max}} = \left\{ \begin{matrix} \text{feasible solutions} \\ \text{with value} \le c_{opt} + \Delta_{max} \end{matrix} \right\} \qquad B_{\Delta_{max}} = \begin{matrix} \text{BDD that} \\ \text{represents} \\ Sol_{\Delta_{max}} \end{matrix}$$

$$B_{\Delta_{max}}$$

Reducing BDD Growth

- It suffices to use a BDD with the same near-optimal solutions as the original BDD. We assume $\Delta \le \Delta_{\max}$.

$$Sol = \left\{ \begin{array}{c} \text{feasible} \\ \text{solutions} \end{array} \right\} \qquad B = \begin{array}{c} \text{original} \\ \text{BDD} \end{array}$$

$$Sol_{\Delta_{\max}} = \left\{ \begin{array}{c} \text{feasible solutions} \\ \text{with value} \le c_{\text{opt}} + \Delta_{\max} \end{array} \right\} \qquad B_{\Delta_{\max}} = \begin{array}{c} \text{BDD that} \\ \text{represents} \\ Sol_{\Delta_{\max}} \end{array}$$

- Unfortunately, $B_{\Delta_{\max}}$ can be exponentially larger than B.
  - Even though it represents a smaller set of solutions.

Reducing BDD Growth

- It suffices to use a BDD with the same near-optimal solutions as the original BDD. We assume $\Delta \leq \Delta_{max}$.

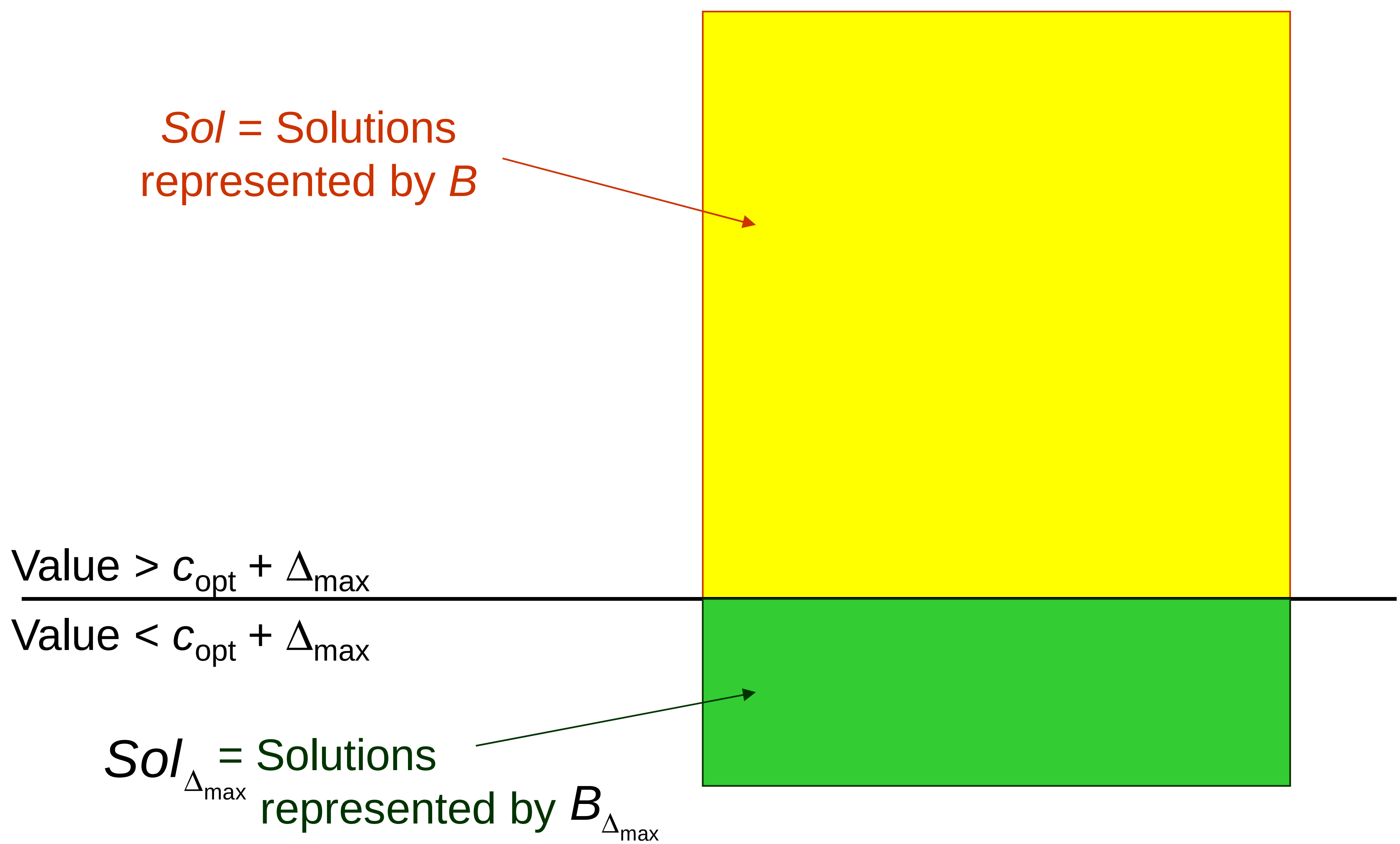
$$Sol = \left\{ \begin{array}{c} \text{feasible} \\ \text{solutions} \end{array} \right\} \qquad B = \begin{array}{c} \text{original} \\ \text{BDD} \end{array}$$

$$Sol_{\Delta_{max}} = \left\{ \begin{array}{c} \text{feasible solutions} \\ \text{with value} \leq c_{opt} + \Delta_{max} \end{array} \right\} \qquad B_{\Delta_{max}} = \begin{array}{c} \text{BDD that} \\ \text{represents} \\ Sol_{\Delta_{max}} \end{array}$$

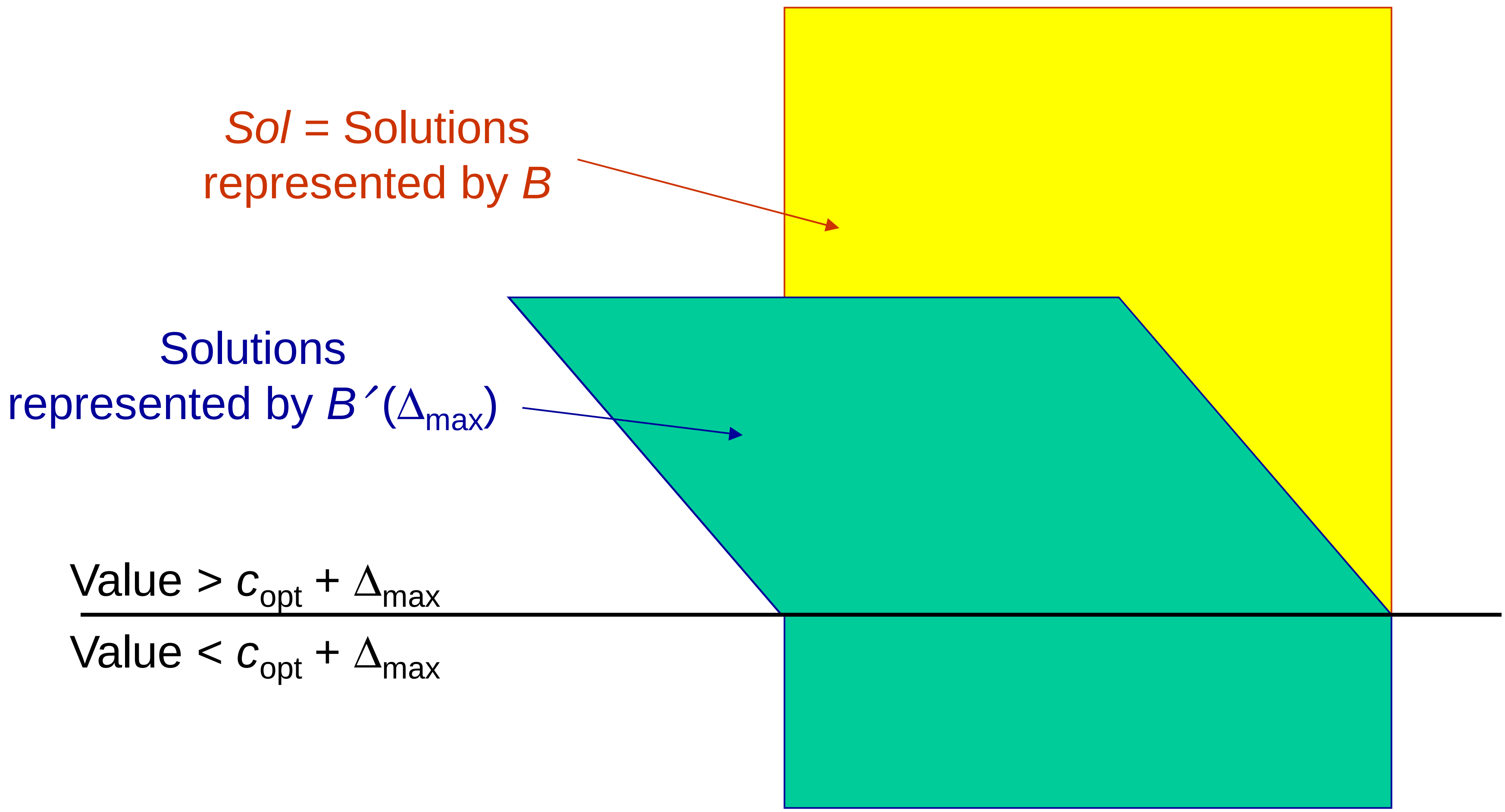
- We will construct a **smaller** BDD $B'(\Delta_{max})$ that is **sound**: $B'(\Delta_{max})_{\Delta_{max}} = B_{\Delta_{max}}$
  - It has the same near optimal solutions as B.

Reducing BDD Growth

Reducing BDD Growth



Reducing BDD Growth



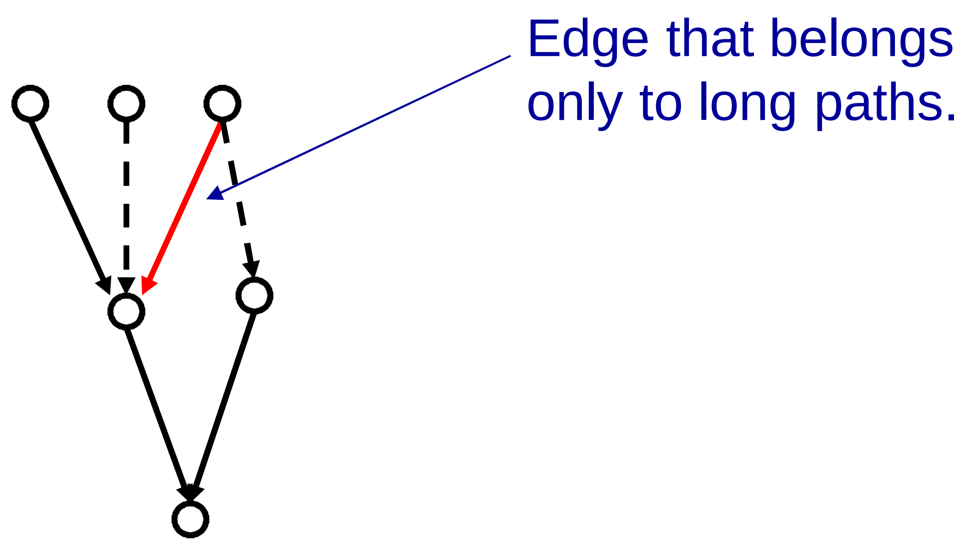
Pruning and Contracting

- We are unaware of a polytime exact method for constructing the **smallest sound** BDD.

- We use two heuristic methods for generating **small sound** BDDs during compilation:
  - Pruning edges
  - Contracting nodes

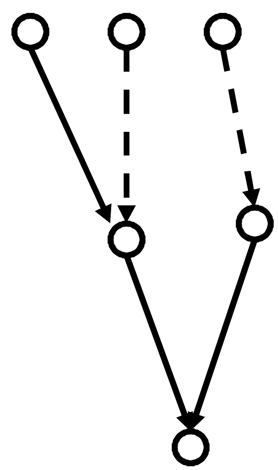
Pruning

- Delete all edges that belong only to paths longer than $c_{opt} + \Delta_{max}$.



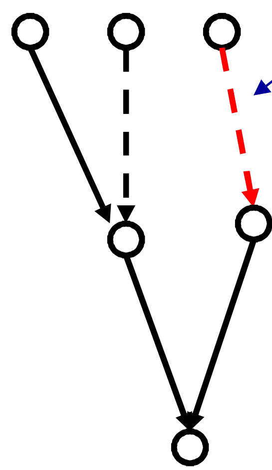
Pruning

- Delete all edges that belong only to paths longer than $c_{opt} + \Delta_{max}$.



Pruning

- Delete all edges that belong only to paths longer than $c_{opt} + \Delta_{max}$.

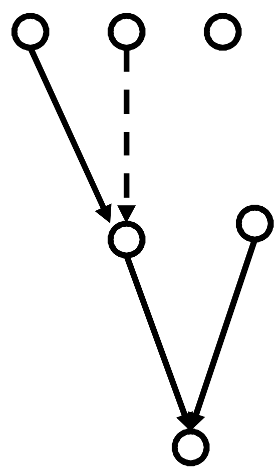


If another edge now belongs only to long paths

Pruning

- Delete all edges that belong only to paths longer than $c_{opt} + \Delta_{max}$.

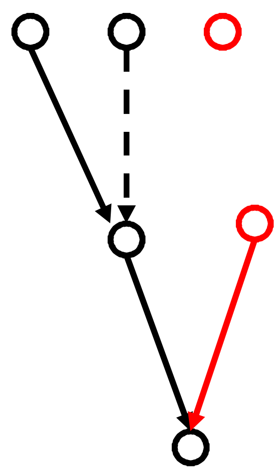
Delete it, too.



Pruning

- Delete all edges that belong only to paths longer than $c_{opt} + \Delta_{max}$.

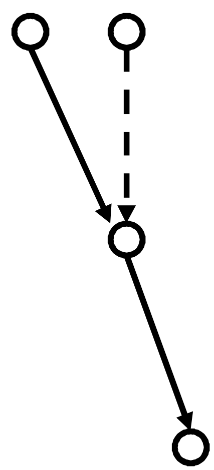
And simplify the BDD.



Pruning

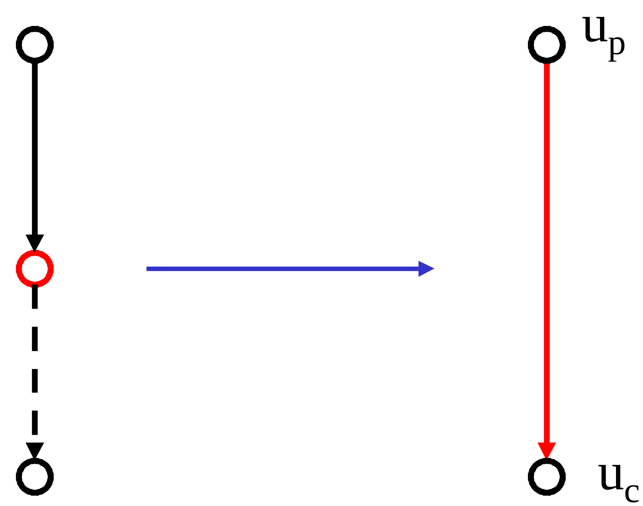
- Delete all edges that belong only to paths longer than $c_{opt} + \Delta_{max}$.

And simplify the BDD.



Contracting

- Remove a node if this creates no new paths shorter than $c_{opt} + \Delta_{max}$



Experimental Results

- We solve the 0-1 problem

$$\begin{aligned} &\min cx \\ &Ax \geq b \\ &x \in \{0,1\}^n \end{aligned}$$

$$b_i = \alpha \sum_j A_{ij}$$

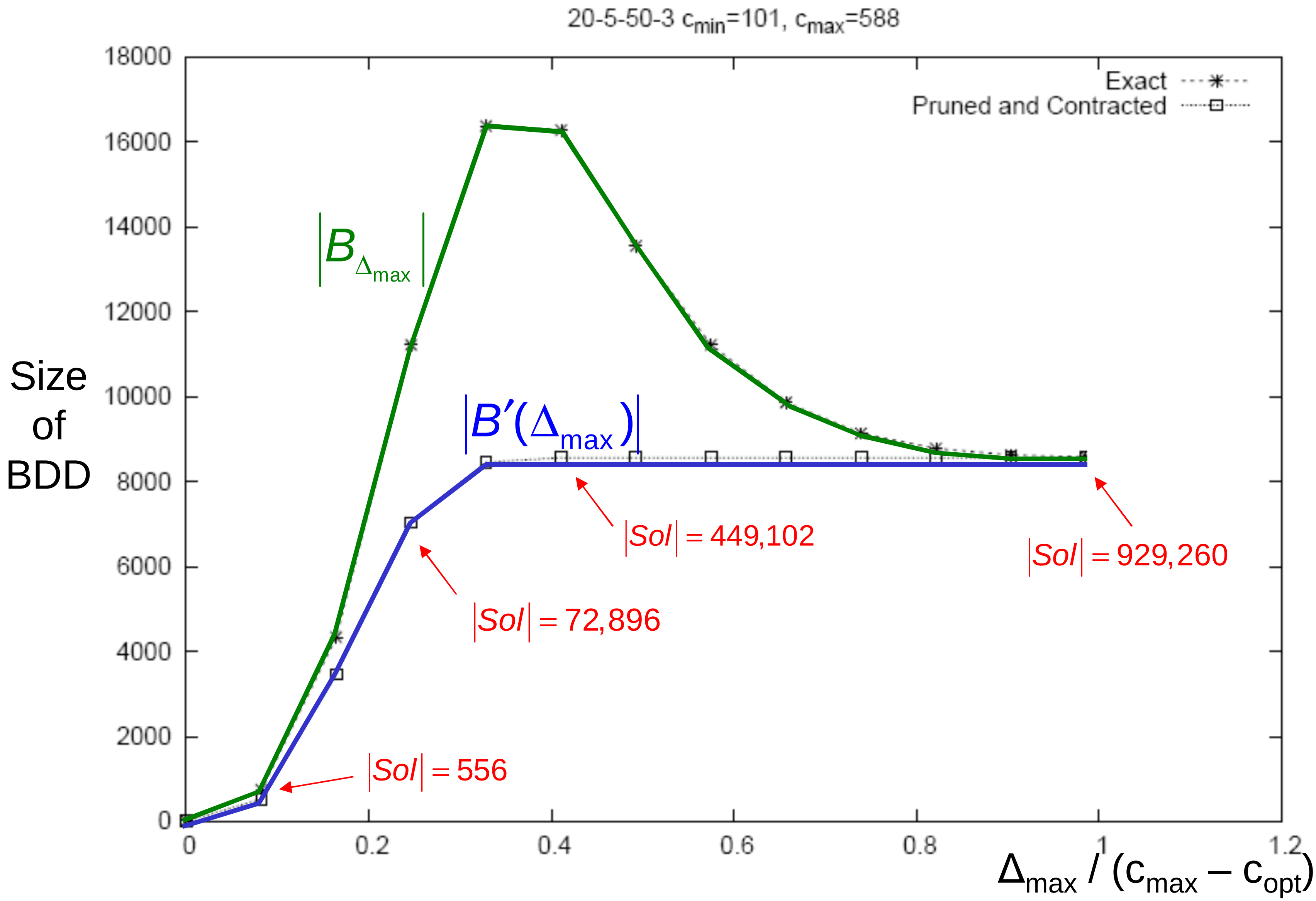
A_{ij} drawn uniformly from $[0,r]$

Experimental Results

- 20 variables, 5 constraints
- $c_{opt} = 101$, $c_{max} = 588$

Sound BDD

Δ_{max}	$\lvert B \rvert$	$\lvert B_{\Delta max} \rvert$	$\lvert B'(\Delta_{max}) \rvert$
0	8,566	20	5
40		742	524
80		4,388	3,456
120		11,217	7,034
200		16,285	8,563
240		13,557	8,566



20-5-50-3 c_{min}=101, c_{max}=588
Size of BDD
Exact
Pruned and Contracted
$|B_{\Delta_{max}}|$
$|B'(\Delta_{max})|$
$|Sol| = 556$
$|Sol| = 72,896$
$|Sol| = 449,102$
$|Sol| = 929,260$
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
0.2
0.4
0.6
0.8
1
1.2
$\Delta_{max} / (c_{max} - c_{opt})$

Experimental Results

- 30 variables, 6 constraints
- $c_{opt} = 36$, $c_{max} = 812$

Δ_{max}	$\lvert B \rvert$	$\lvert B_{\Delta max} \rvert$	$\lvert B'(\Delta_{max}) \rvert$
0	925,610	30	10
50		3,428	2,006
150		226,683	262,364
200		674,285	568,863
250		1,295,465	808,425
300		1,755,378	905,602

Experimental Results

- 40 variables, 8 constraints
- $c_{opt} = 110$, $c_{max} = 1241$

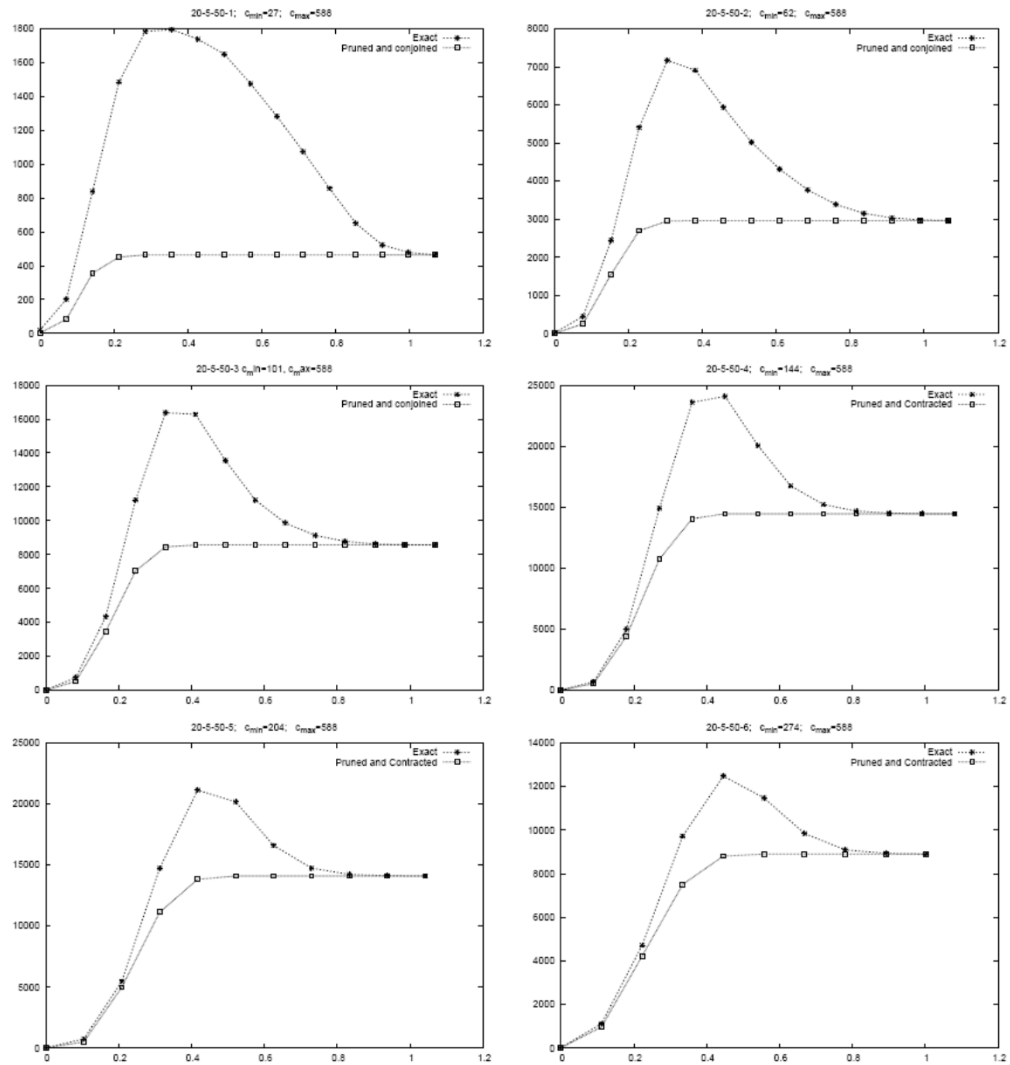
Δ_{max}	$\lvert B \rvert$	$\lvert B_{\Delta max} \rvert$	$\lvert B'(\Delta_{max}) \rvert$
0	?	40	12
15		1,143	402
35		3,003	1,160
70		11,040	7,327
100		404,713	223,008
140		?	52,123

Experimental Results

- 60 variables, 10 constraints
- c_{min} = 67, c_{max} = 3179

Δ_{max}	$\lvert B \rvert$	$\lvert B_{\Delta max} \rvert$	$\lvert B'(\Delta_{max}) \rvert$
0	?	60	7
50		5,519	1,814
100		111,401	78,023

Experimental Results



Results when tightness α is gradually reduced

Experimental Results

- MIPLIB instances
- $\Delta_{max} = 0$ (BDD represents all optimal solutions)

Instance	$\|B\|$	$\|B_{\Delta max}\|$	$\|B'(\Delta_{max})\|$
lseu	?	99	19
p0033	375	41	21
p0201	310,420	737	84
stein27	25,202	6,260	4,882
stein45	5,102,257	1,765	1,176

Conclusions and Future Work

- Cost-bounded BDDs provide reasonable scalabililty for BDD-based postoptimality analysis in 0-1 linear programming.

Future work:

- Tests on nonlinear, nonconvex 0-1 problems
  - Nonlinearity, nonconvexity should not be a major factor.
- Extension to general integer problems.
  - Straightforward; a matter of implementation.
- Extension to MILP.
  - ??