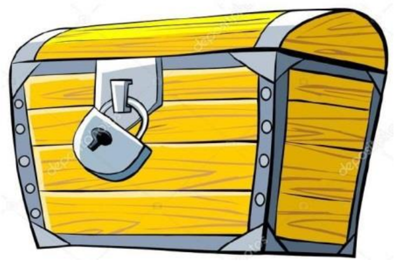
Compact Representation of Near-Optimal Integer Programming Solutions

John Hooker
Carnegie Mellon University
Thiago Serra
Mitsubishi Electric Research Laboratories

ISMP 2018
Bordeaux, France

Two Perspectives on Optimization

Traditional



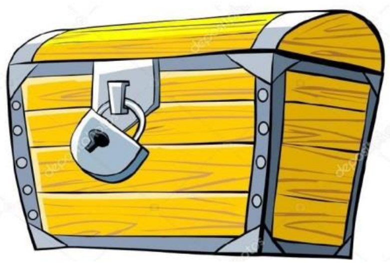
Problem formulation
(nontransparent)

Solver

A few solutions
revealed

Two Perspectives on Optimization

Traditional

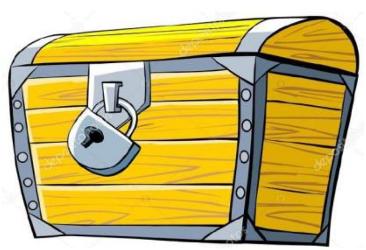


Problem formulation
(nontransparent)

Solver

A few solutions
revealed

Proposed



Problem formulation
(nontransparent)

Solver

Transparent
data structure

Two Perspectives on Optimization

Traditional

Nontransparent data structure

$$Ax \geq b$$

Solver →

Optimal solution, or list of (near-optimal) solutions

This wastes a wealth of information collected for the model, perhaps at great expense

Two Perspectives on Optimization

Traditional

Nontransparent data structure

$Ax \geq b$

Solver →

Optimal solution, or list of (near-optimal) solutions

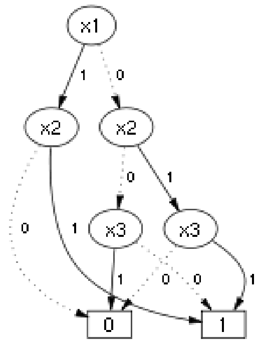
Proposed

Nontransparent data structure

$Ax \geq b$

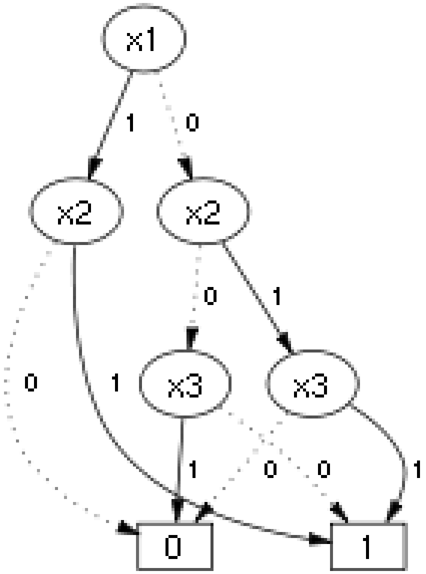
Solver →

Transparent data structure



Postoptimality Analysis

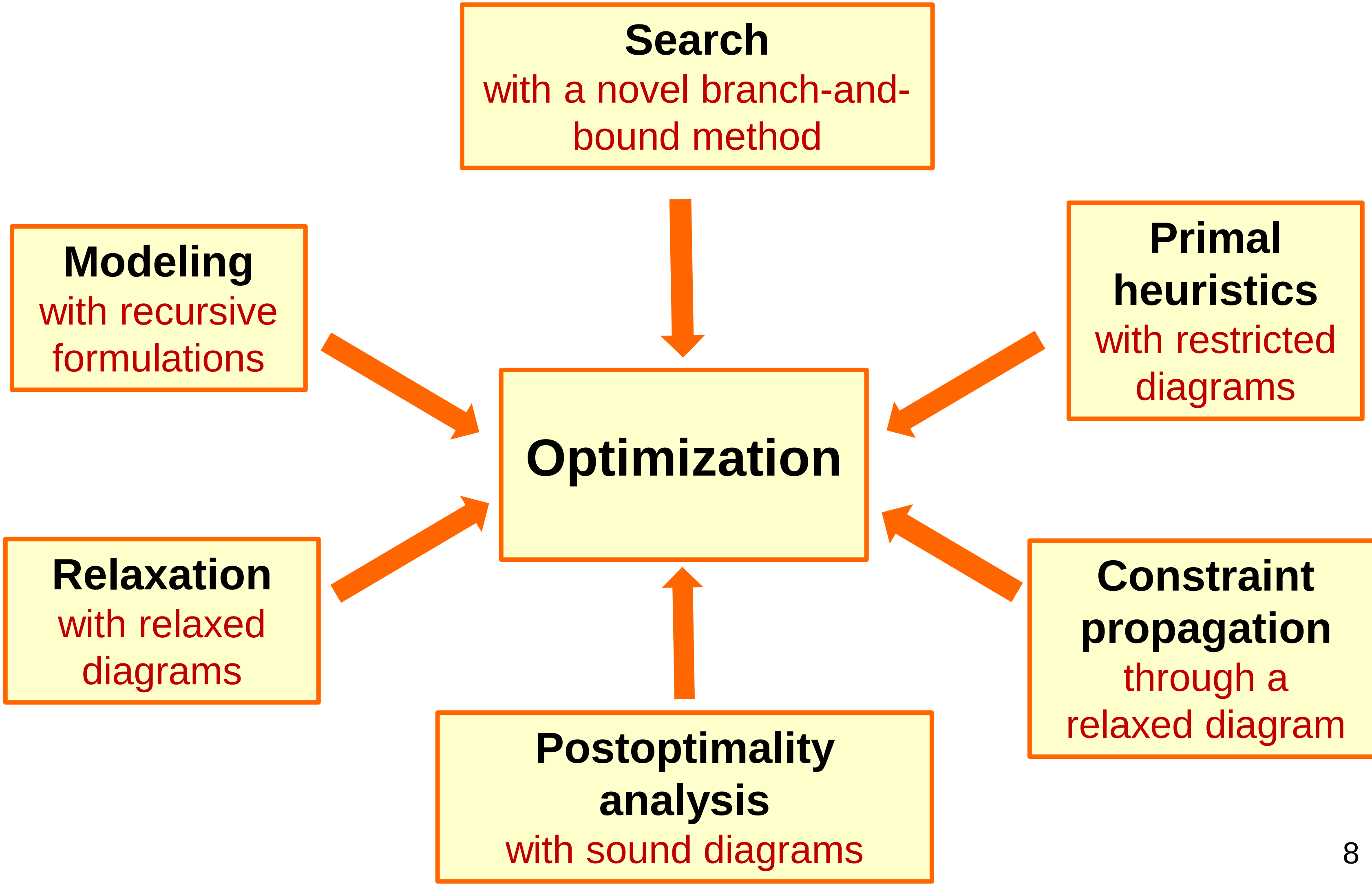
- **Decision diagrams** provide a transparent data structure
  - Can compactly represent **all near-optimal solutions** (within Δ of optimum).
  - Open the door to more comprehensive postoptimality analysis
  - Can be **efficiently queried** with what-if questions.



Decision Diagrams

- Used in **computer science** and **AI** for decades
  - Logic circuit design
  - Product configuration
- A **new perspective** on optimization
  - An alternative **data structure**
  - A **new tool** to do many of the things we do in optimization.

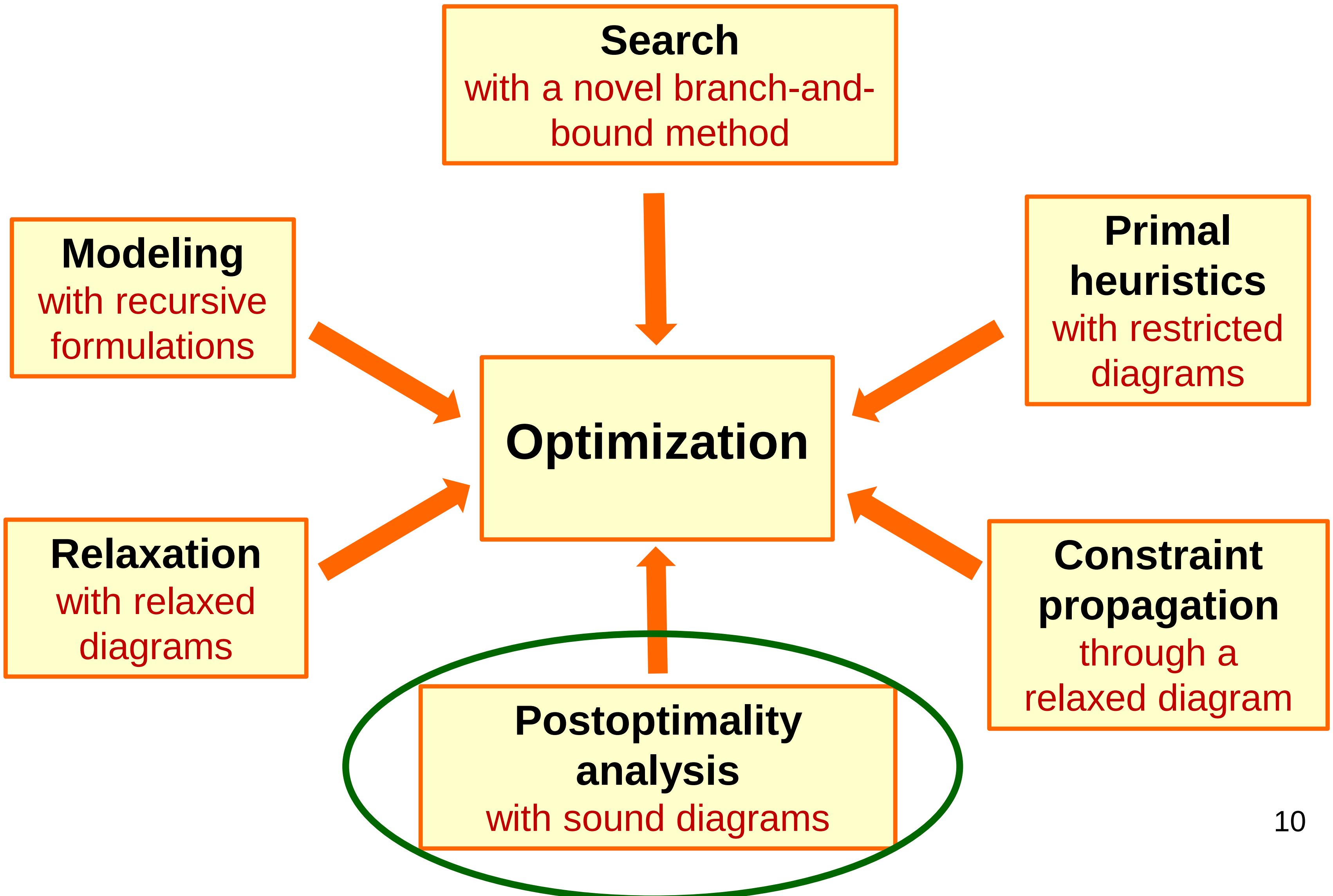
DDs in Optimization



Decision Diagrams

- Some advantages:
  - No need for **inequality** formulations.
  - No need for **linear** or **convex** relaxations.
  - Exploits **recursive structure** in the problem, but…
  - Solves **dynamic programming** models **without state space enumeration**.
  - Effective **parallel** computation.
  - Ideal for **postoptimality** analysis

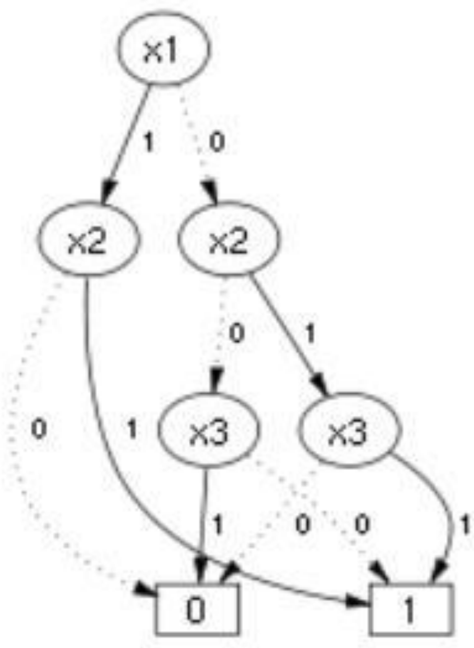
DDs in Optimization



Decision Diagram Basics

- Binary decision diagrams encode Boolean functions

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



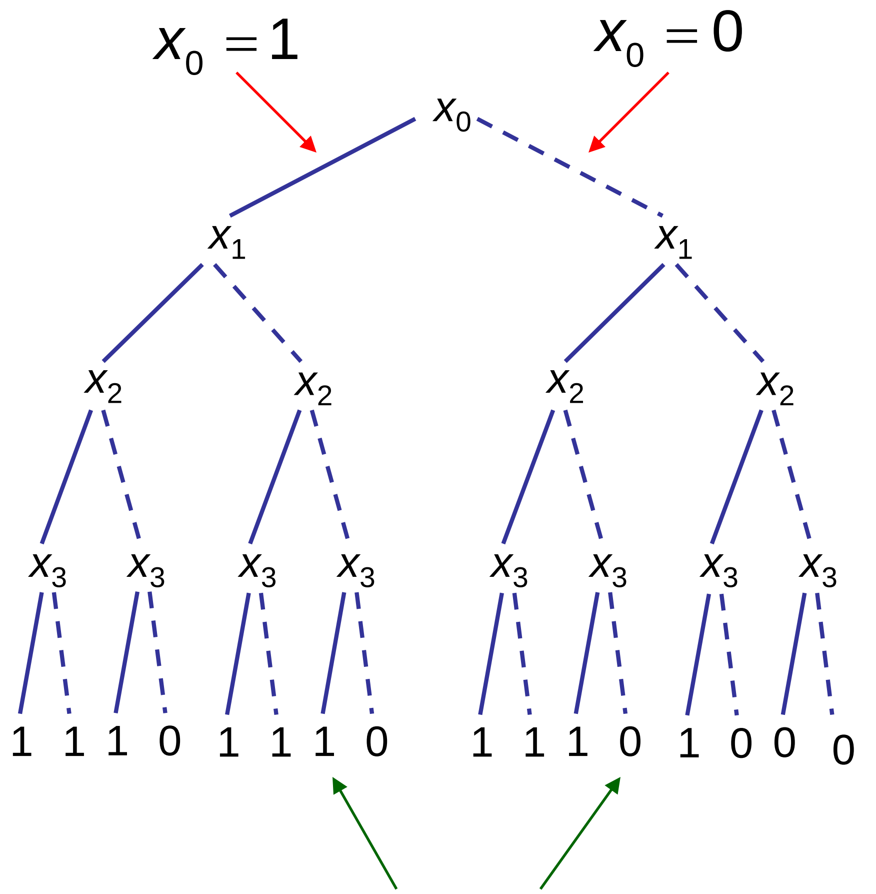
Boole (1847), Shannon (1937), Lee (1959), Akers (1978), Bryant (1986)

Reduced Decision Diagrams

- There is a **unique reduced** DD representing any given function.
  - Once the variable ordering is specified.

Bryant (1986)

- The reduced DD can be viewed as a branching tree with **redundancy** removed.
  - Superimpose isomorphic subtrees.
  - Remove redundant nodes.

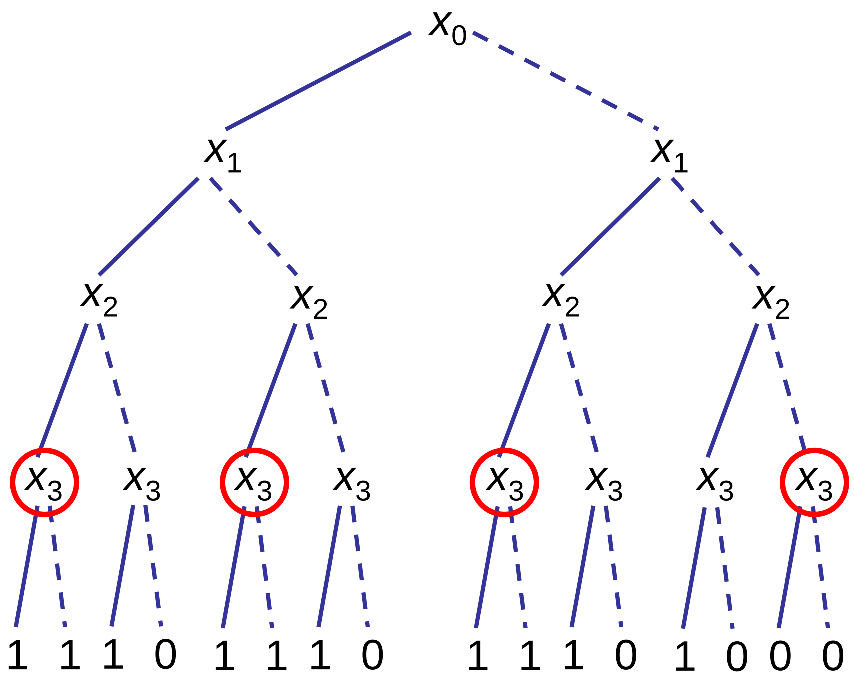


1 indicates feasible solution,
0 infeasible

Branching tree for 0-1 inequality

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

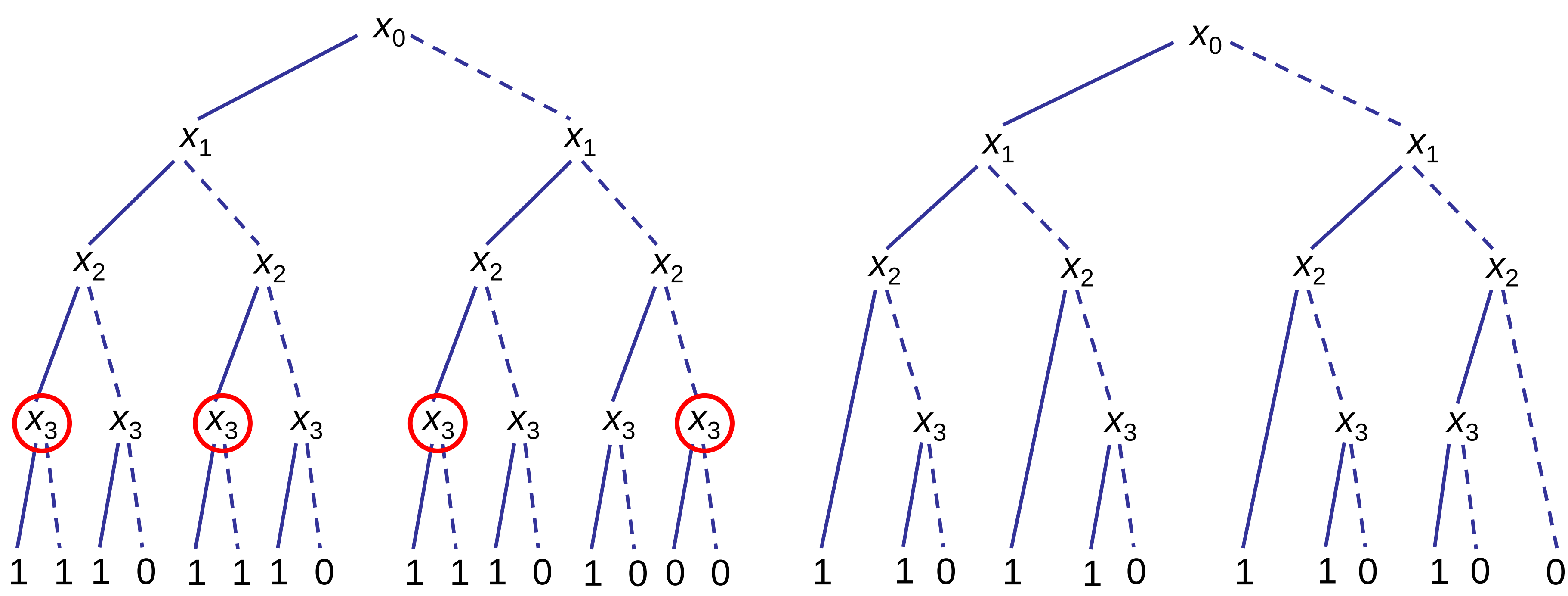
1



Branching tree for 0-1 inequality

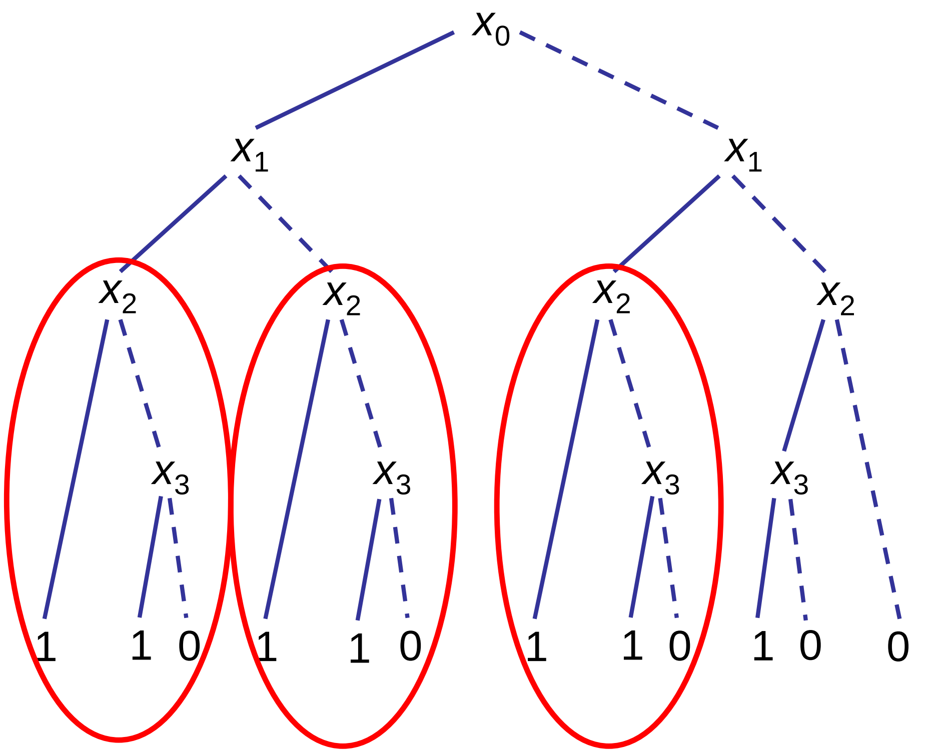
$$2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$

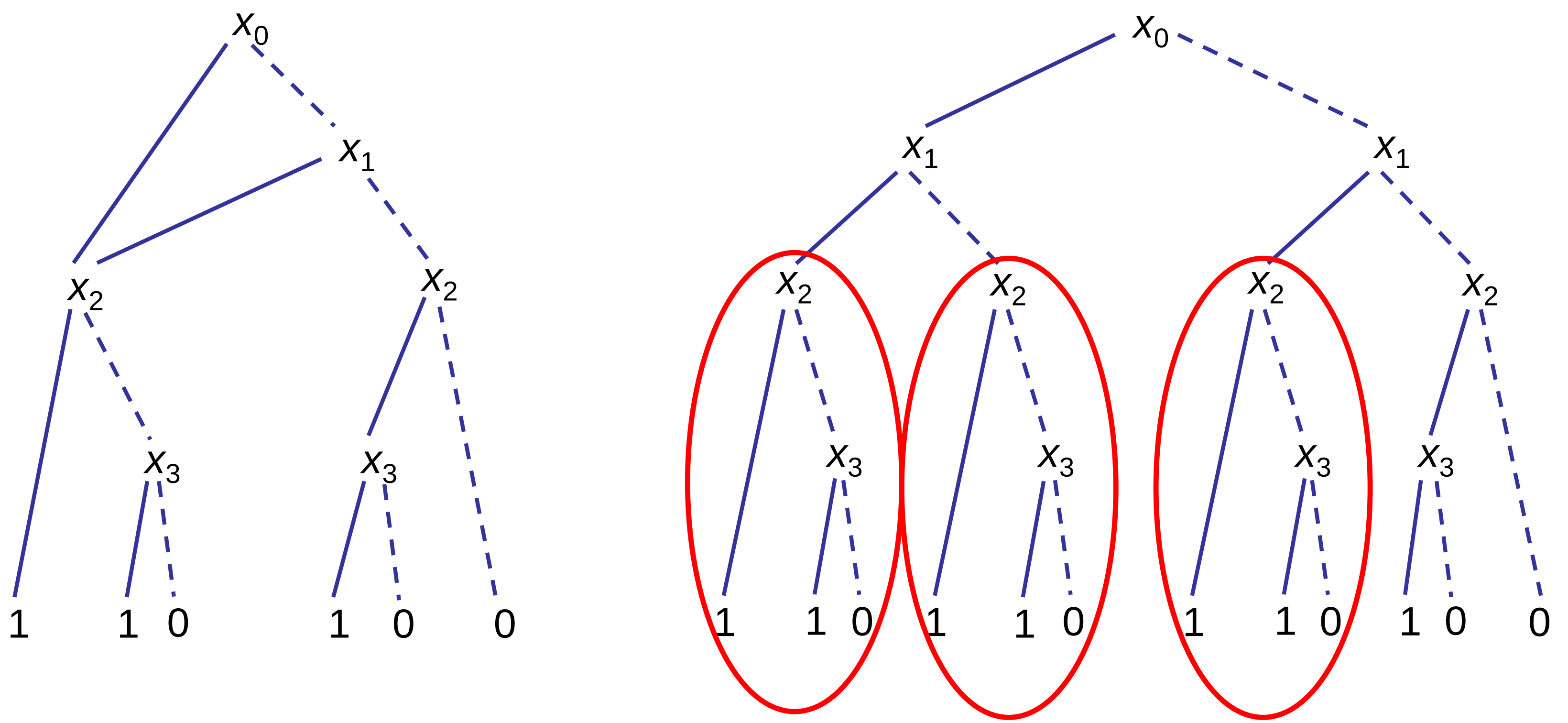
Remove redundant nodes…

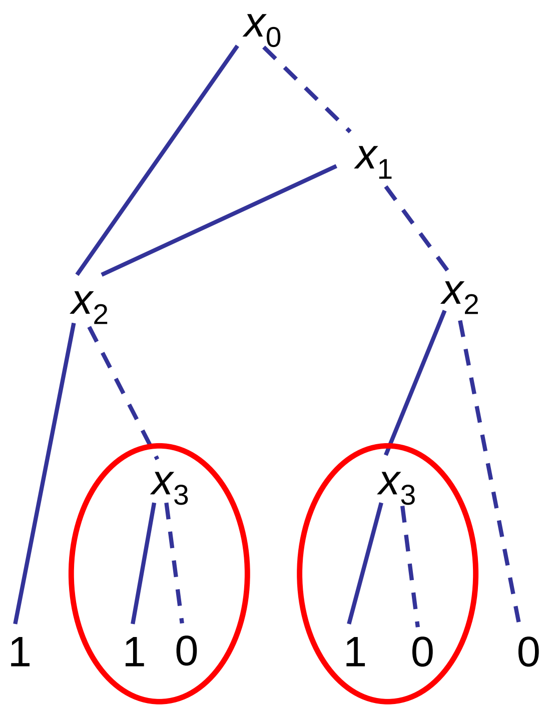


x_0
x_1 x_1
x_2 x_2 x_2 x_2
x_3 x_3 x_3 x_3 x_3 x_3 x_3 x_3
1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0
x_0
x_1 x_1
x_2 x_2 x_2 x_2
x_3 x_3 x_3 x_3
1 1 0 1 1 0 1 1 0 1 0 0

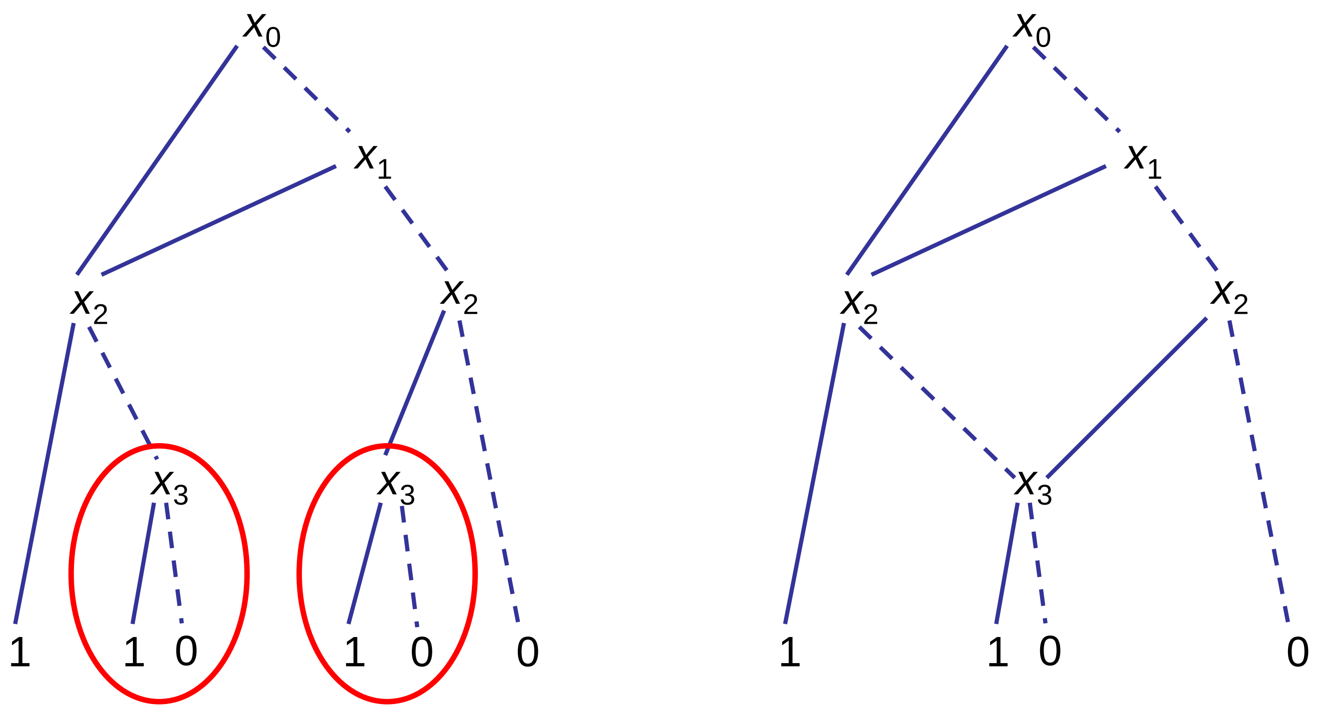
Superimpose identical subtrees…




x_0
x_1
x_2 x_2
x_3 x_3
1 1 0 1 0 0
x_0
x_1 x_1
x_2 x_2 x_2 x_2
x_3 x_3 x_3 x_3
1 1 0 1 1 0 1 1 0 1 0 0

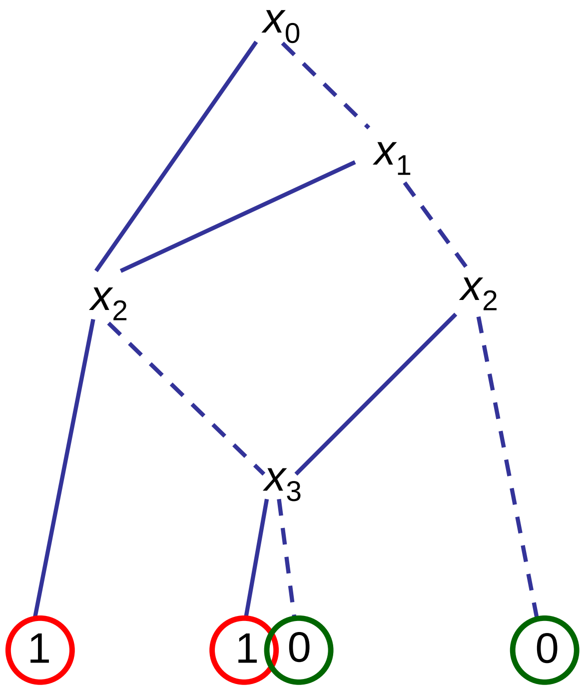


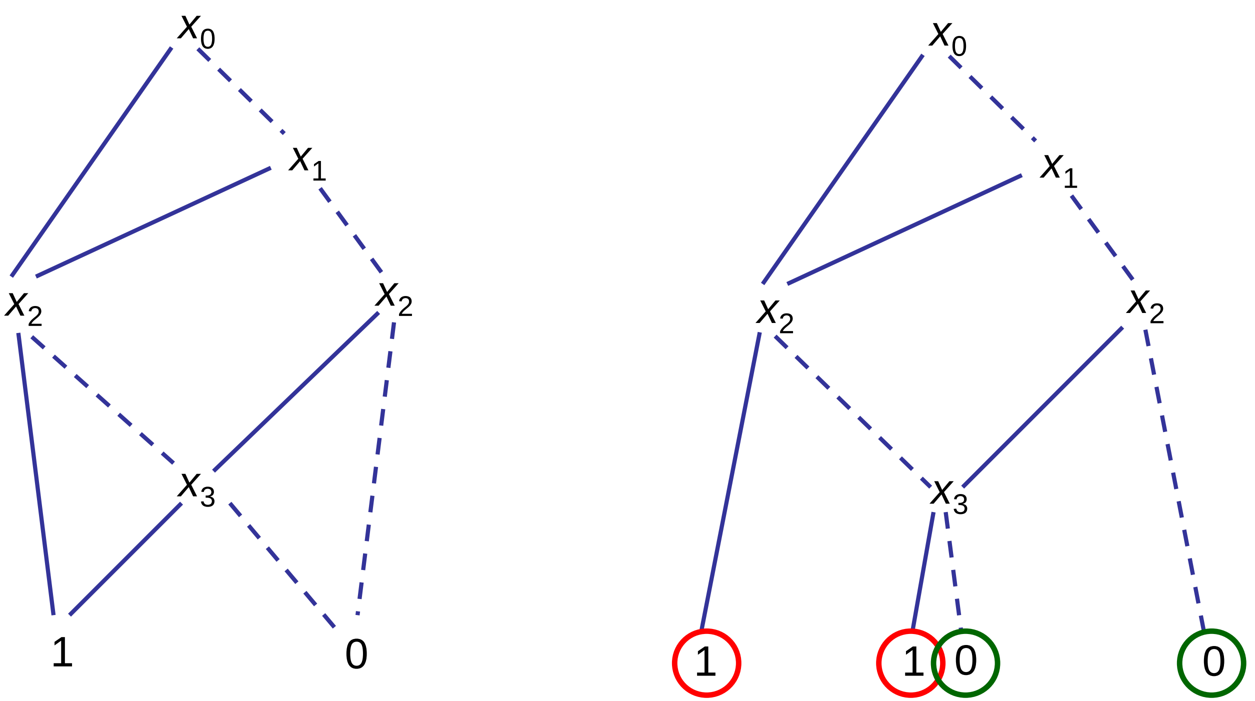
Superimpose identical subtrees…



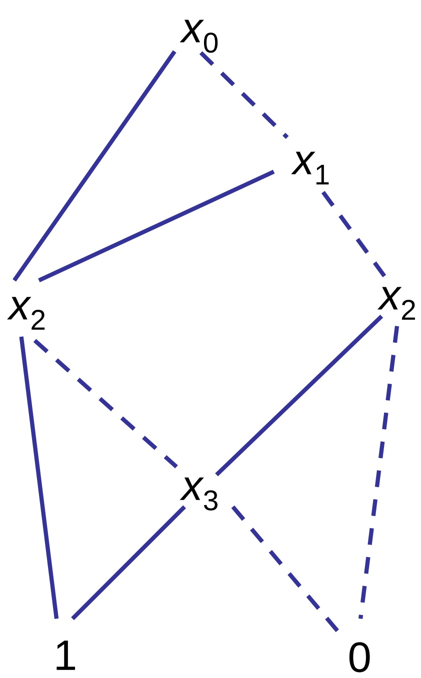
x_0
x_1
x_2
x_2
x_3
x_3
1
1
0
1
0
0
x_0
x_1
x_2
x_2
x_3
1
1
0
0

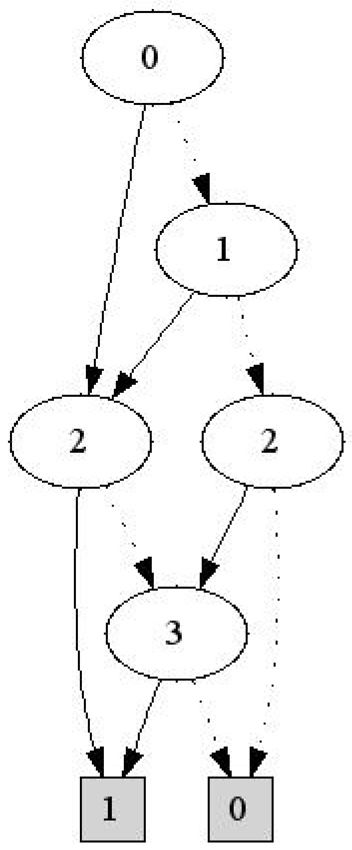
Superimpose identical leaf nodes…





x0
x1
x2
x2
x3
1
0
x0
x1
x2
x2
x3
1
1
0
0

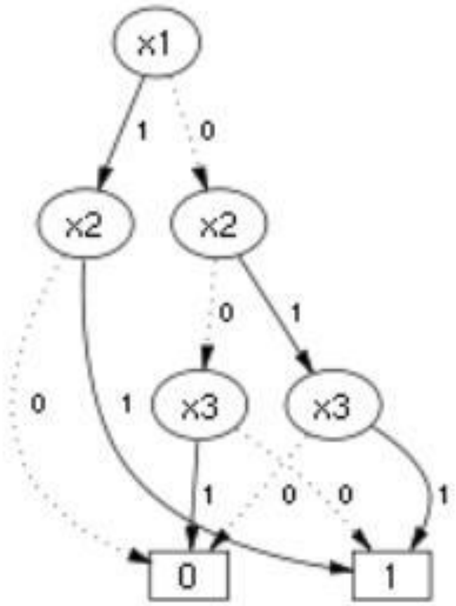




as generated by software

Optimization with Exact Decision Diagrams

- Decision diagrams can represent feasible set
  - Remove paths to 0.
  - Paths to 1 are feasible solutions.
  - Associate costs with arcs.
  - Reduces **optimization** to a **shortest path** problem

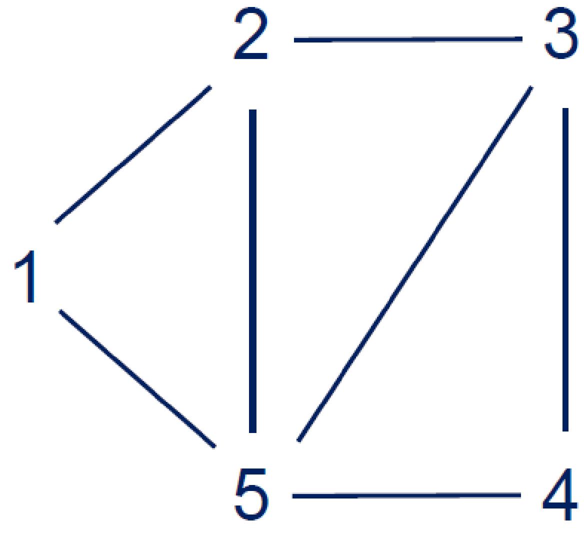


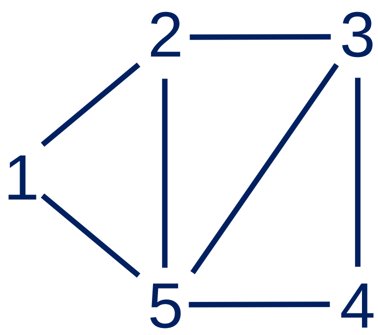
Hadžić and JH (2006, 2007)

Stable Set Problem

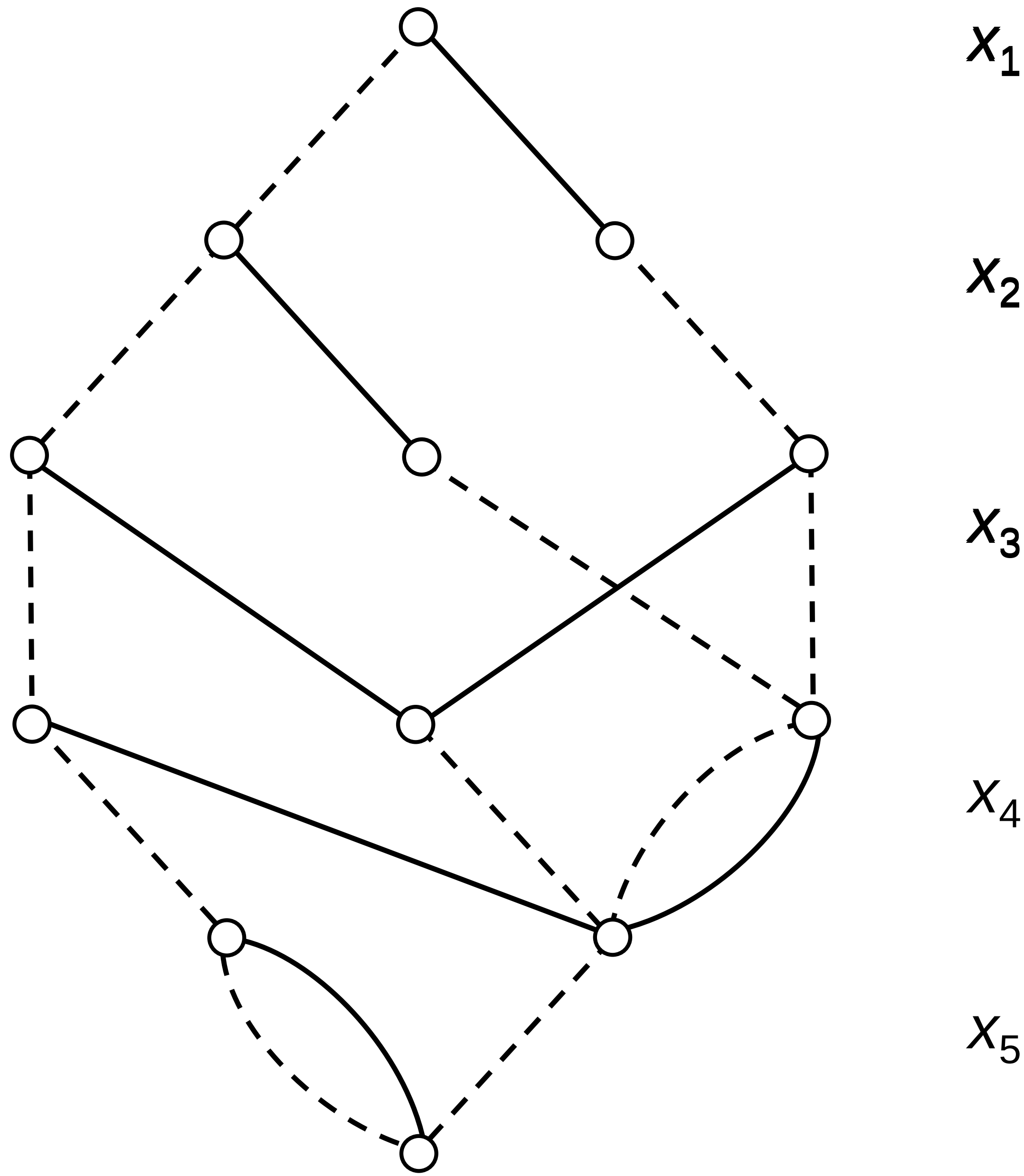
Let each vertex have weight w_i

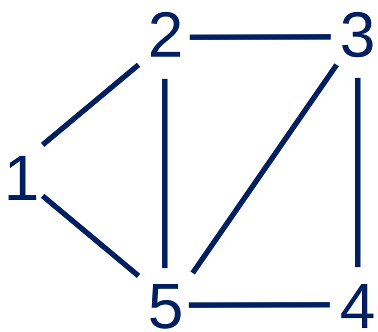
Select nonadjacent vertices to maximize $\sum_i w_i x_i$

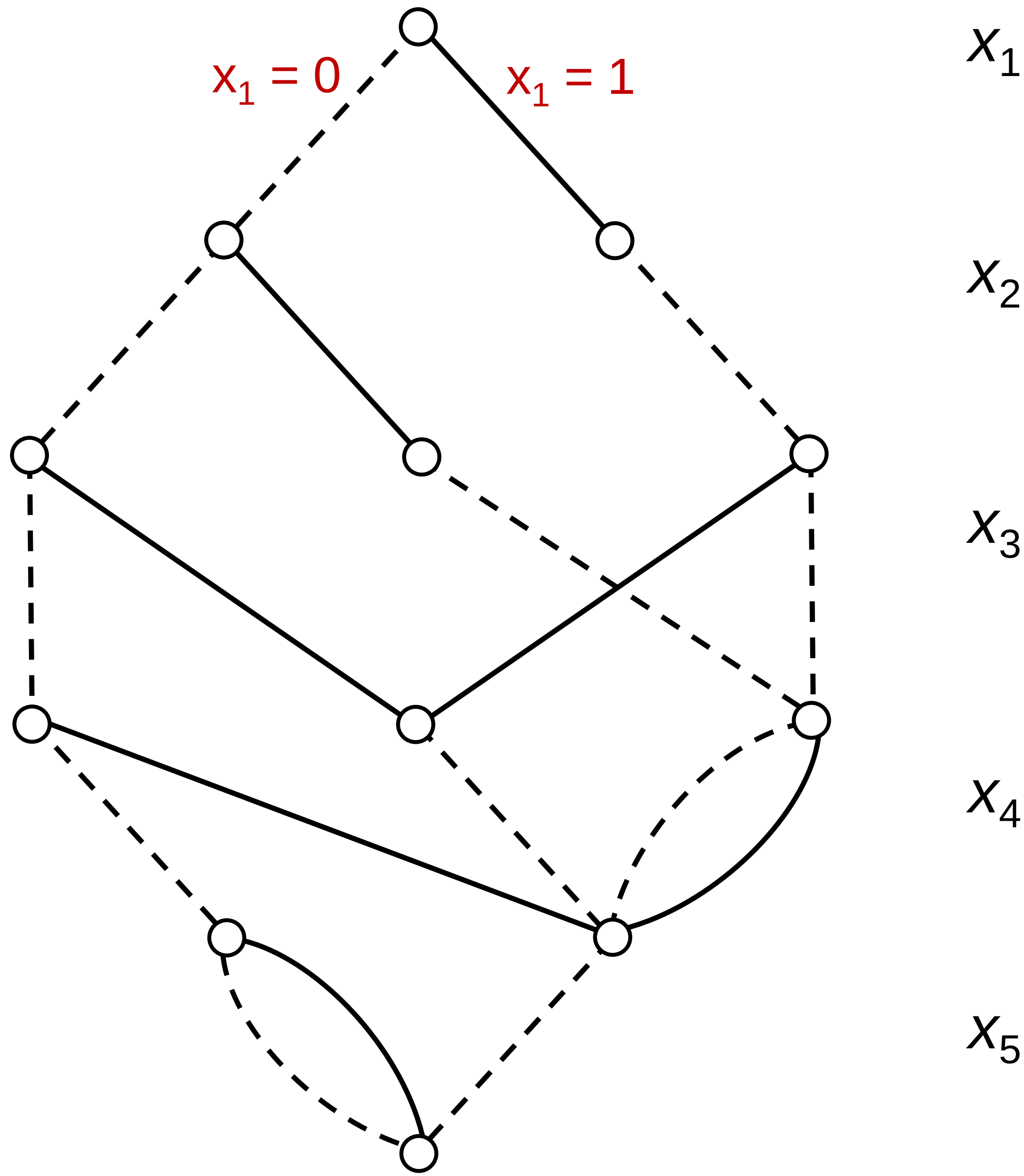




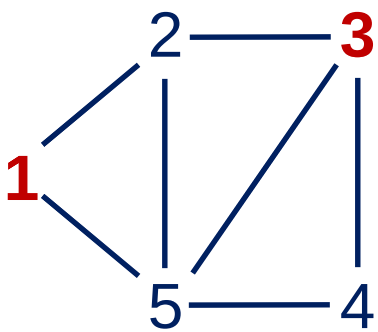
Exact DD for stable set problem



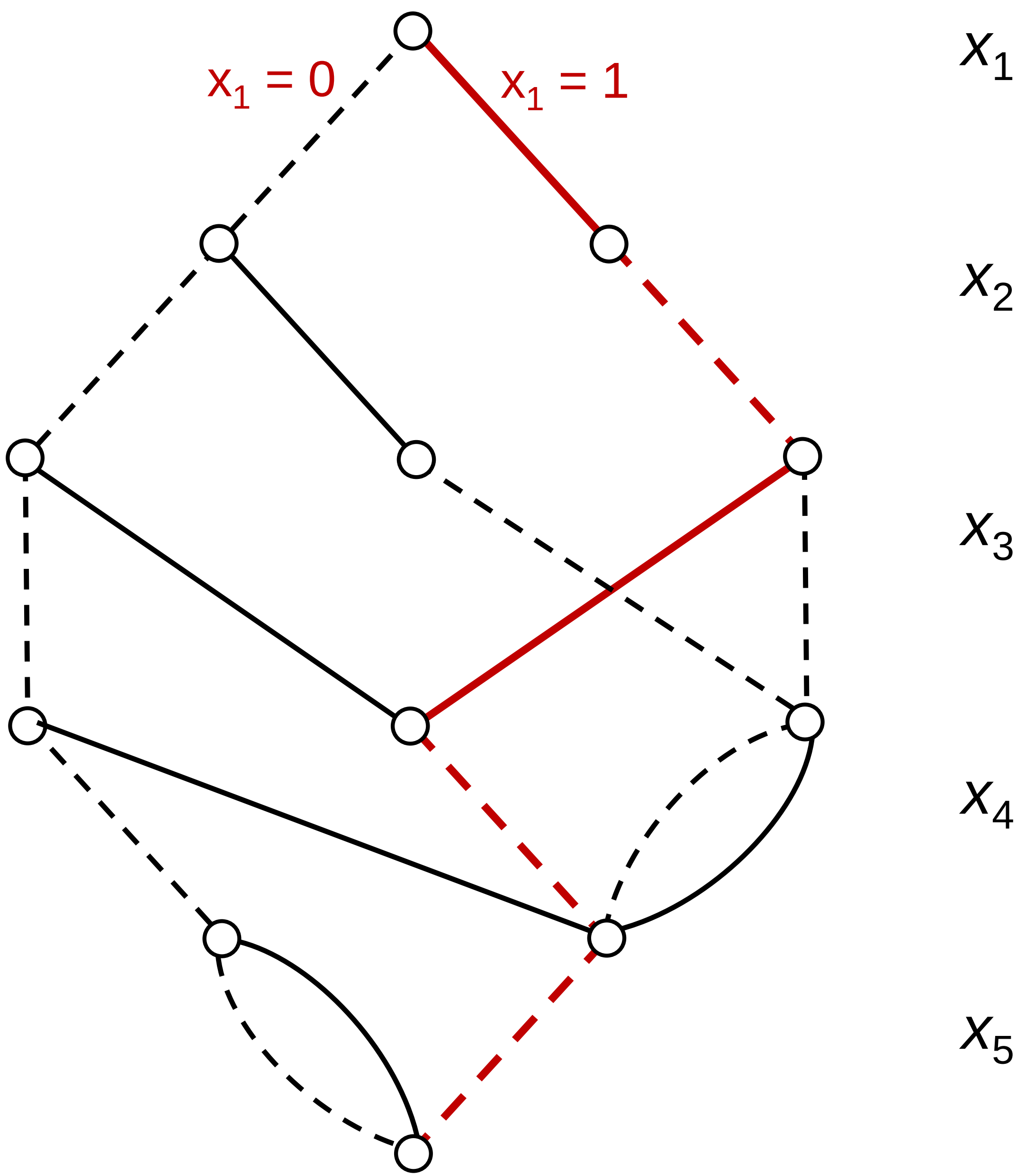


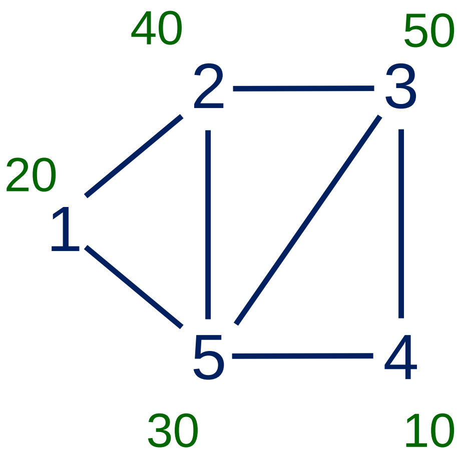


Exact DD for stable set problem

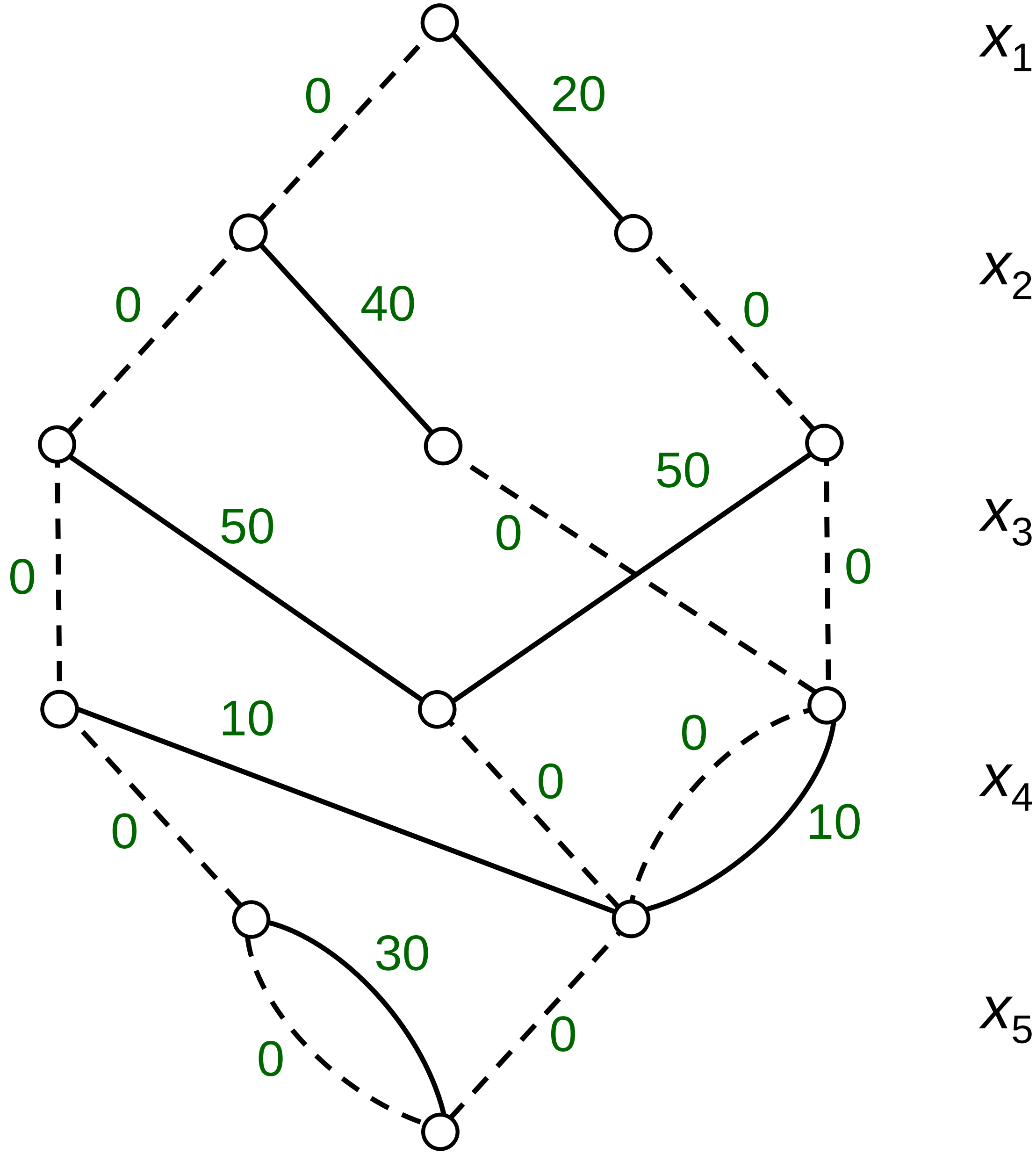


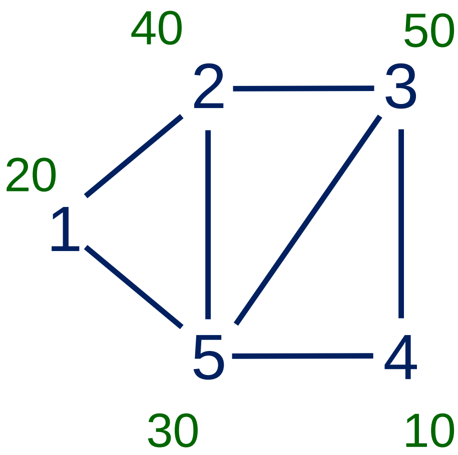
Paths from top to bottom correspond to the 9 feasible solutions





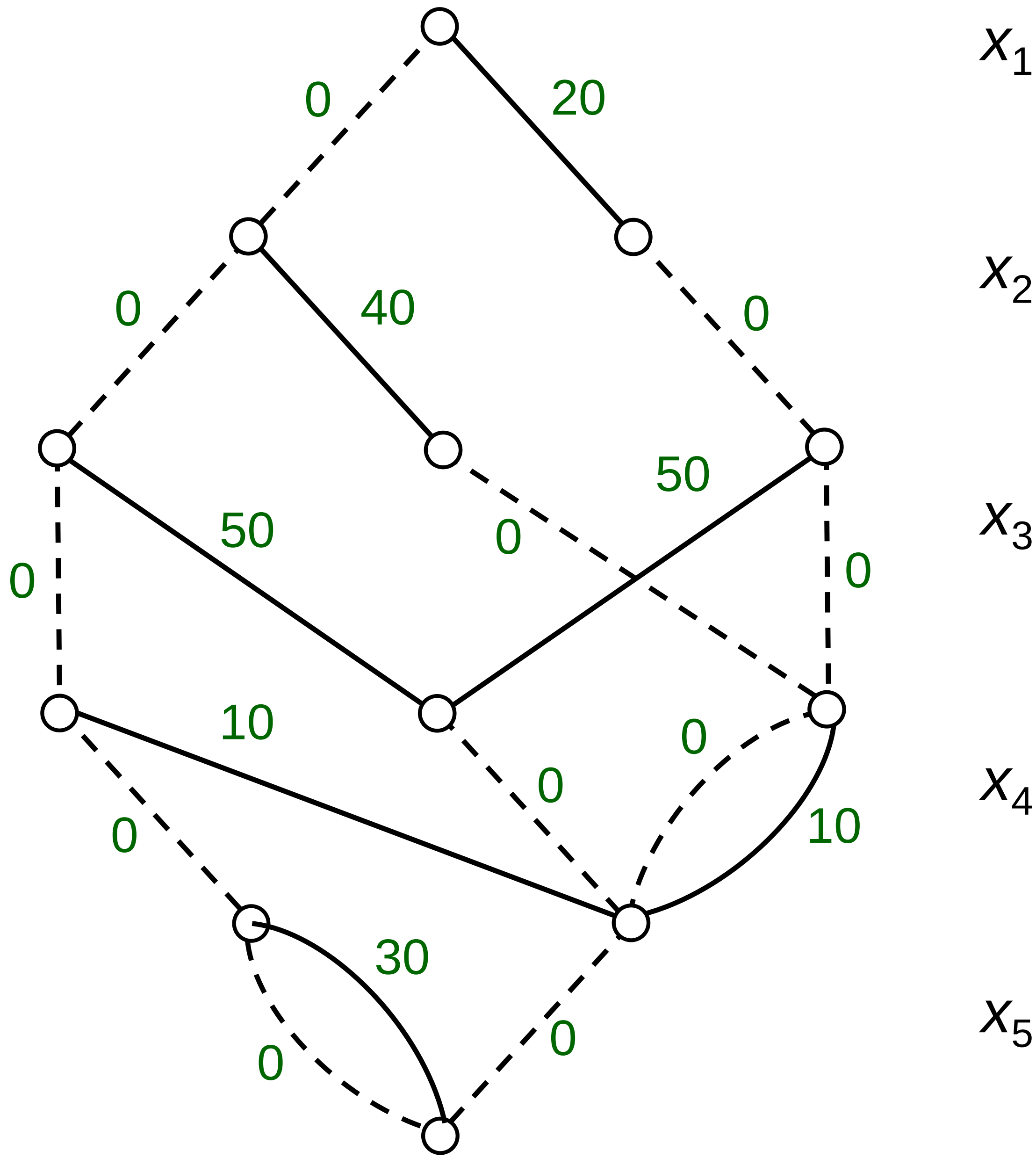
For objective function, associate weights with arcs

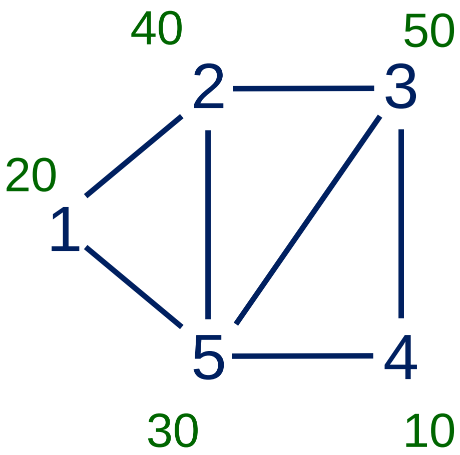




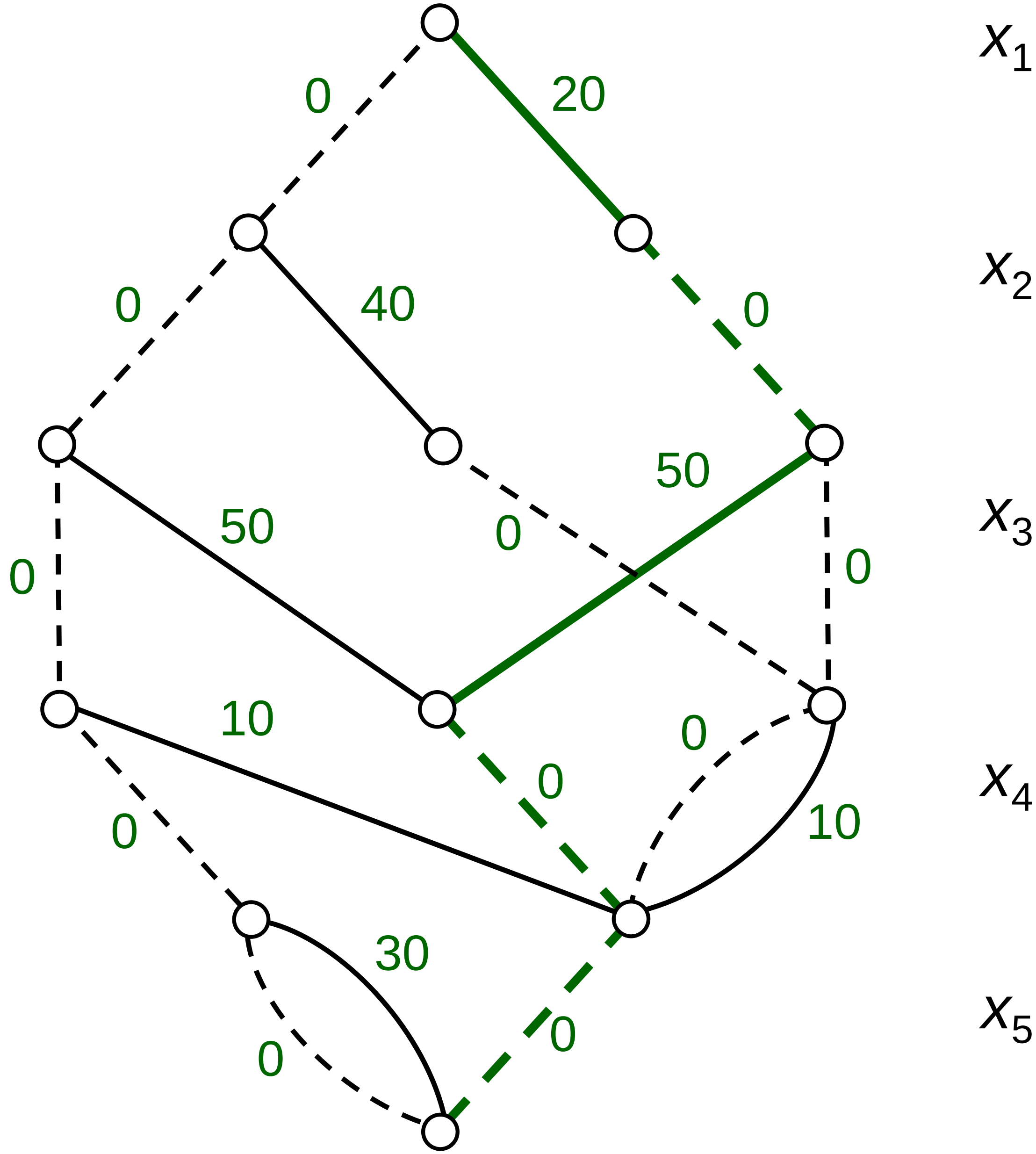
For objective function, associate weights with arcs

Optimal solution is **longest path**





For objective function, associate weights with arcs



Optimal solution is **longest path**

Near-optimal Solutions

- Let v^* = optimal cost.
- A solution is **Δ-optimal** if it is feasible and its cost is $\leq v^* + \Delta$
  - We wish to represent all Δ-optimal solutions in a decision diagram.
  - The diagram is generated once for multiple queries.
    - In general, the user will be interested in δ-optimal solutions for $\delta < \Delta$.

Hadžić and JH (2007)

Sound Decision Diagrams

- **Sound** DDs can store near-optimal solutions more compactly.
  - Sound = all Δ-**optimal solutions** are included…
    - …along with some **spurious** solutions (feasible and infeasible) that are **worse than Δ-optimal**
      - That is, cost > $v^* + \Delta$.

Hadžić and JH (2007)

Sound Decision Diagrams

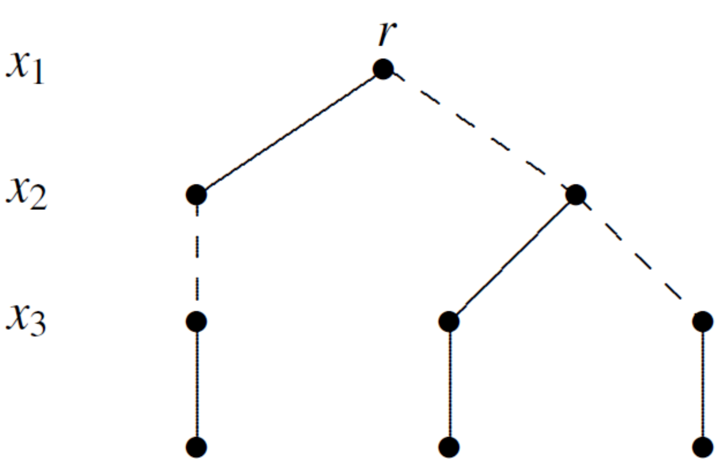
- **Sound** DDs can store near-optimal solutions more compactly.
  - Sound = all Δ-**optimal solutions** are included…
    - …along with some **spurious** solutions (feasible and infeasible) that are **worse than Δ-optimal**
      - That is, cost > v^* + Δ.
    - These solutions are easily screened out.
    - No effect whatever on most queries.
  - Paradoxically, this can result in a **smaller** DD.

Hadžić and JH (2007)

Sound Decision Diagrams

$$\begin{array}{ll} \text{minimize} & 4x_1 + 3x_2 + 2x_3 \\ \text{subject to} & x_1 + x_3 \geq 1, \quad x_2 + x_3 \geq 1, \quad x_1 + x_2 + x_3 \leq 2 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{array}$$

Branching tree



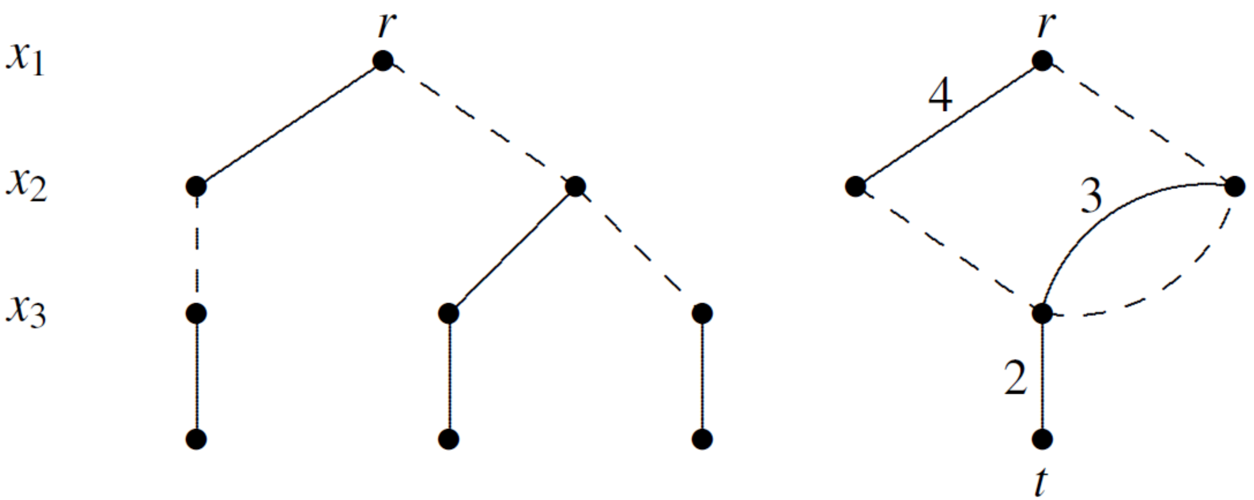
Optimal value = 2

Sound Decision Diagrams

$$\begin{array}{ll} \text{minimize} & 4x_1 + 3x_2 + 2x_3 \\ \text{subject to} & x_1 + x_3 \geq 1, \;\; x_2 + x_3 \geq 1, \;\; x_1 + x_2 + x_3 \leq 2 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{array}$$

Branching tree

Reduced weighted DD



Optimal value = 2

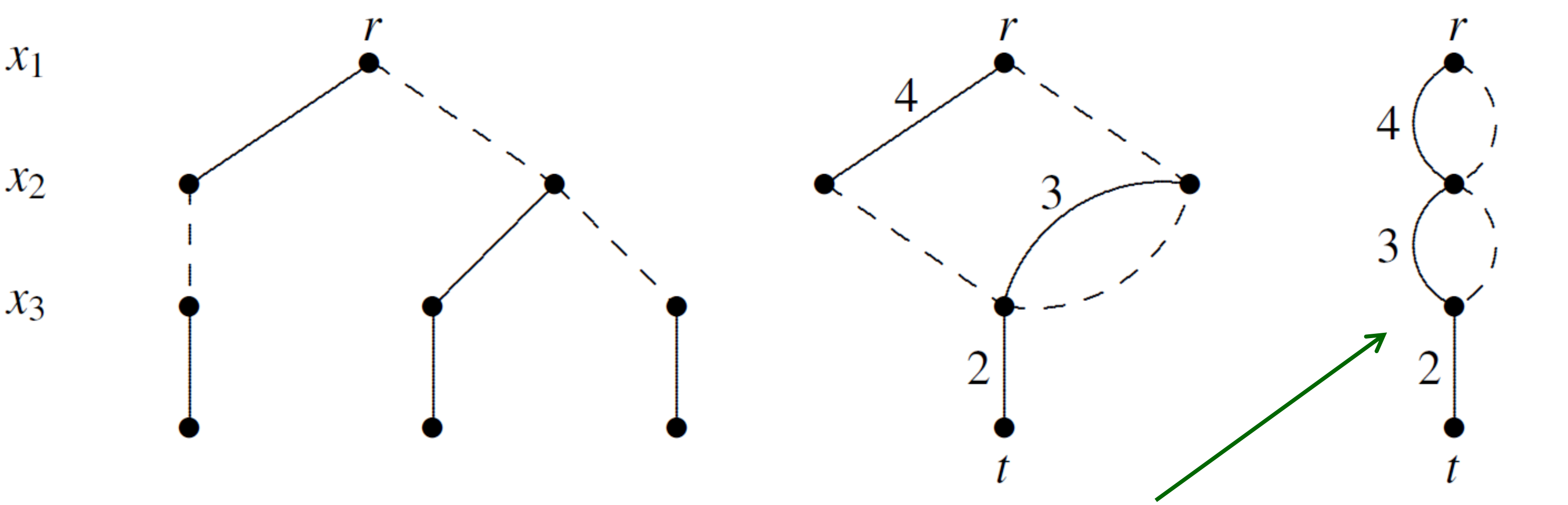
Sound Decision Diagrams

$$
\begin{array}{ll}
\text{minimize} & 4x_1 + 3x_2 + 2x_3 \\
\text{subject to} & x_1 + x_3 \geq 1, \;\; x_2 + x_3 \geq 1, \;\; x_1 + x_2 + x_3 \leq 2 \\
 & x_1, x_2, x_3 \in \{0, 1\}
\end{array}
$$

Branching tree

Reduced weighted DD

Sound DD for Δ = 6



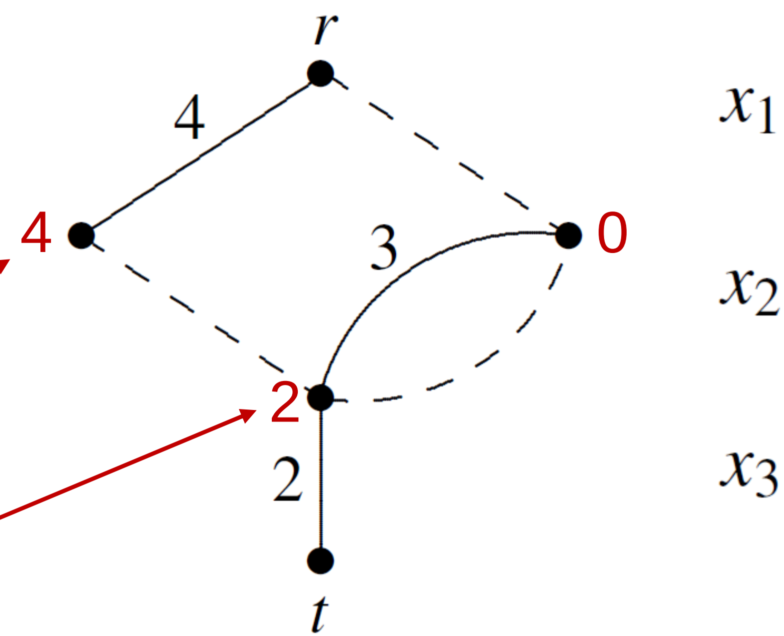
Optimal value = 2

Contains spurious solution x = (1,1,1)
Its value = 9 > 2 + Δ

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?

First use **original** DD.
Set $\Delta = 6$, so that all feasible solutions are Δ-optimal and included in the diagram



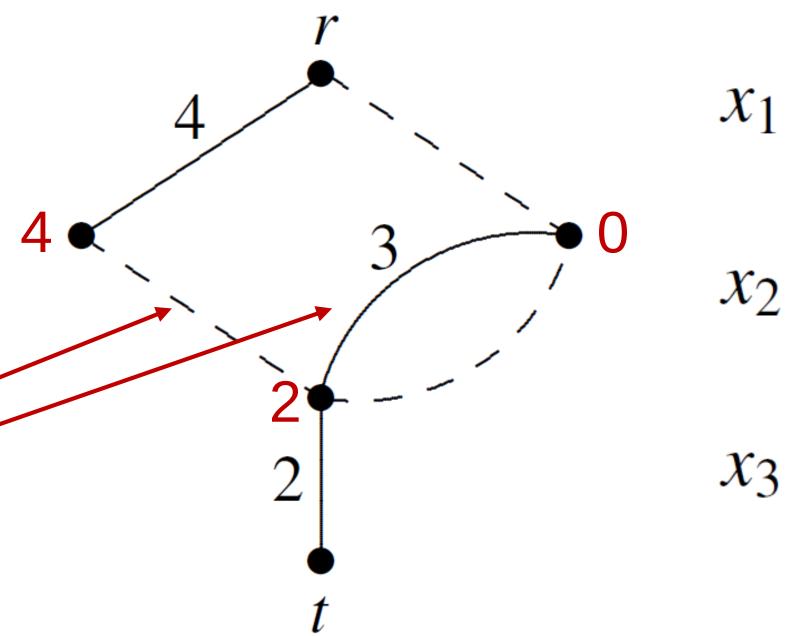
Shortest distance to root

Shortest distance to terminus

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?

First use **original** DD. Set $\Delta = 6$, so that all feasible solutions are Δ-optimal and included in the diagram

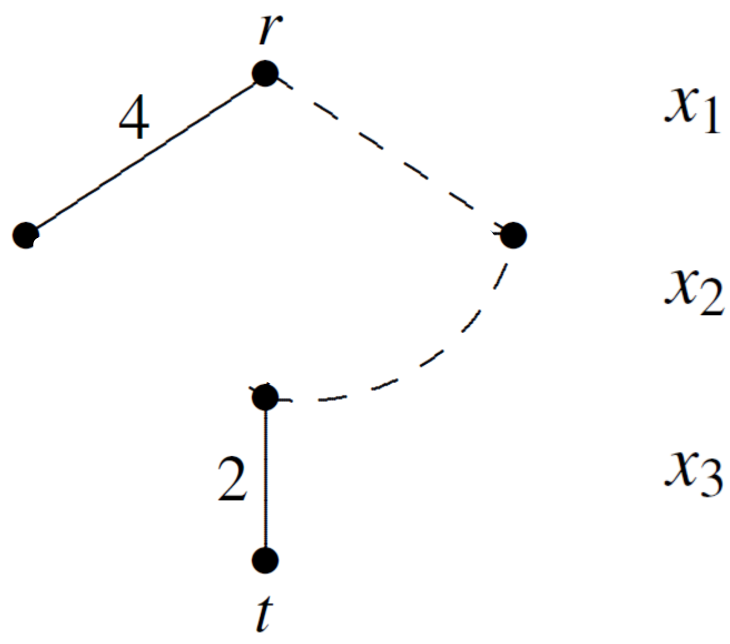


These arcs are removed because $4 + 0 + 2 > 2 + \delta$
$0 + 3 + 2 > 2 + \delta$

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?

First use **original** DD.
Set $\Delta = 6$, so that all feasible solutions are Δ-optimal and included in the diagram

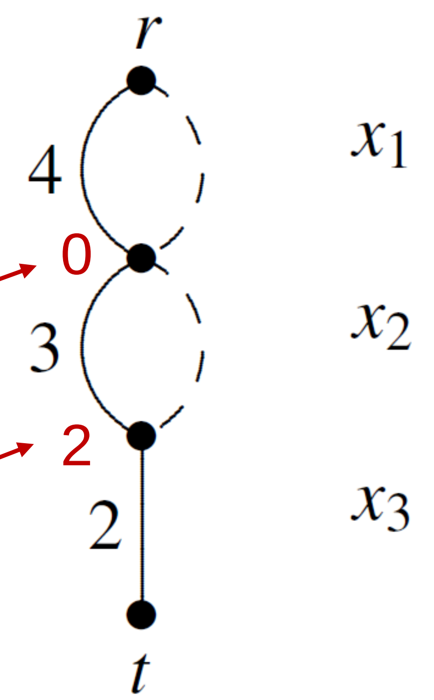


x_2 can take only value 0 when $\delta = 2$.

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?

Now use **sound** DD.
Again set $\Delta = 6$, so that all feasible solutions are Δ-optimal and included in the diagram



Shortest distance to root

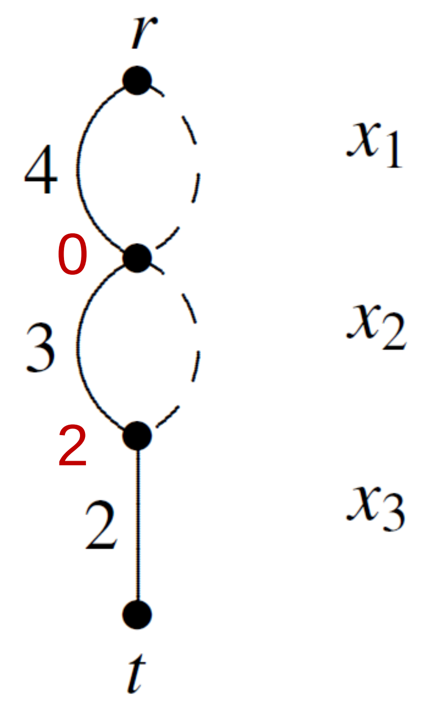
Shortest distance to terminus

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?

Now use **sound** DD.
Again set $\Delta = 6$, so that all feasible solutions are Δ-optimal and included in the diagram

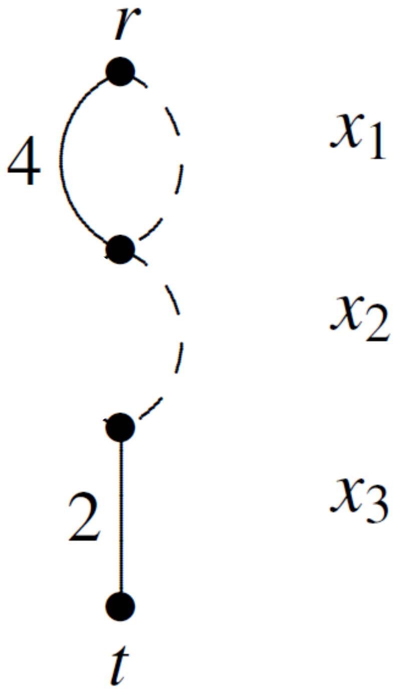
This arc is removed because $0 + 3 + 2 > 2 + \delta$



Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?

Now use **sound** DD.
Again set $\Delta = 6$, so that all feasible solutions are Δ-optimal and included in the diagram



x_2 can take only value 0 when $\delta = 2$.
Spurious solution has no effect on the analysis.

Sound DDs

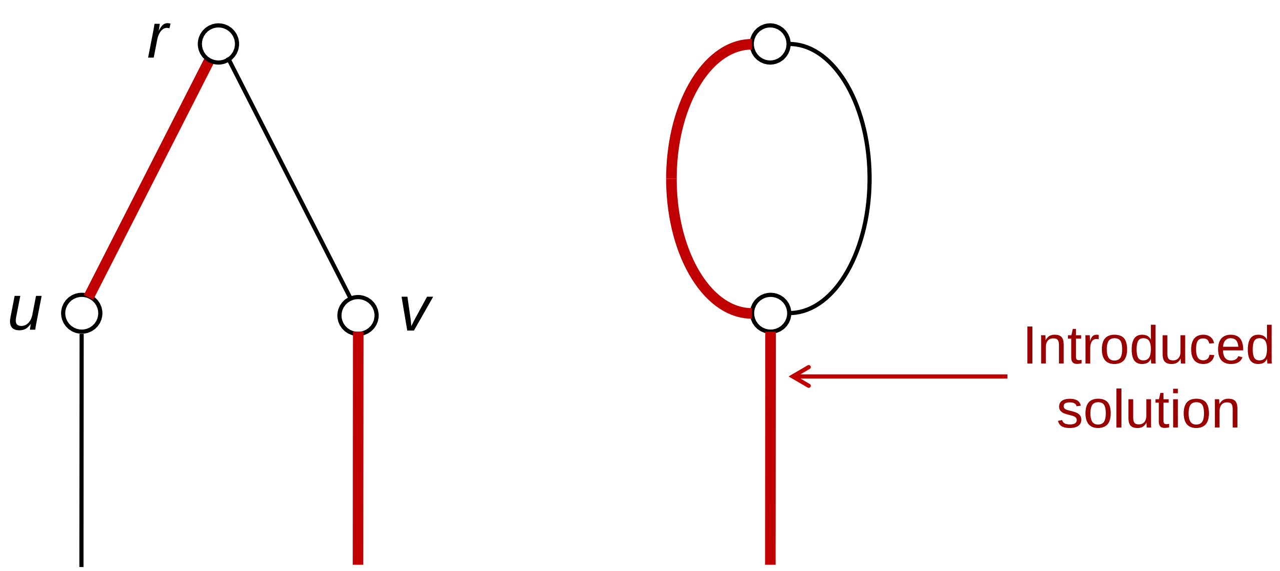
- A sound diagram is **minimal** if no arcs/nodes can be removed without destroying soundness.
  - Easy to check whether an arc/node can be removed.

Theorem. A minimal sound diagram for $\Delta = 0$ (optimal solutions only) never contains spurious solutions.

So there is no point in using sound diagrams for optimal solutions only.

Sound Reduction

- We can **sound reduce** node ***u*** into node ***v*** when this introduces only spurious solutions
  - Easy to check while building diagram. Then merge ***u*** and ***v***.

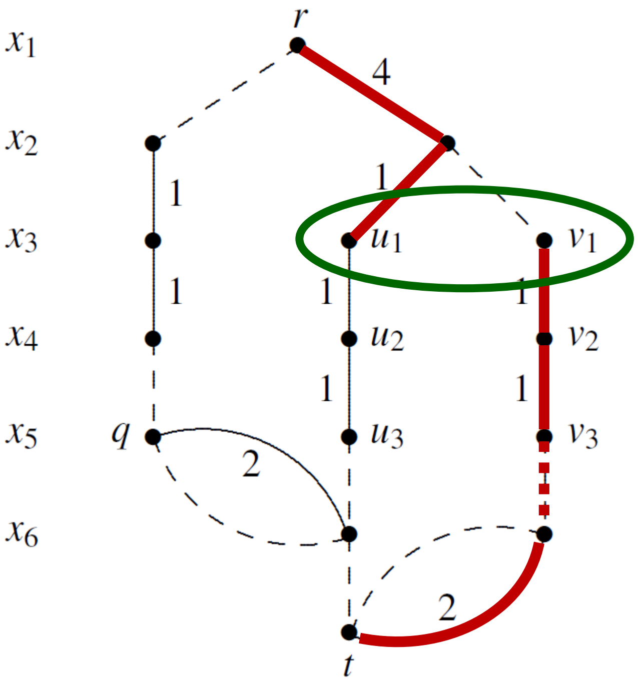


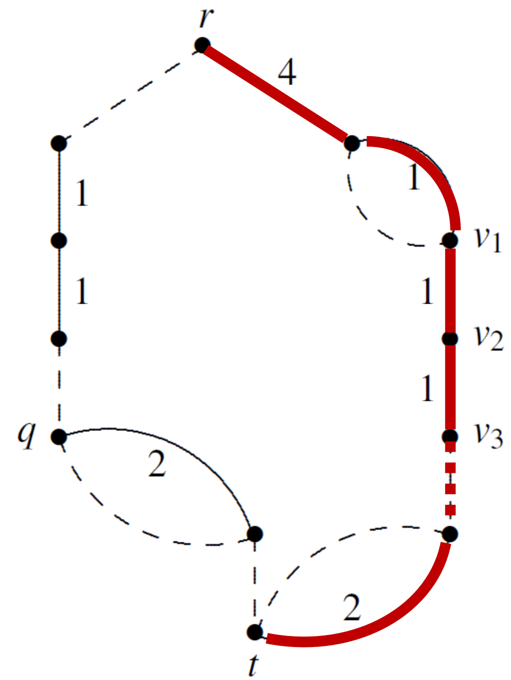
$$\mathrm{Suf}_\Delta(u) \subseteq \mathrm{Suf}(v)$$

$$w(\pi) + w(\sigma) > z^* + \Delta \;\text{ when } x(\pi) \in \mathrm{Pre}(u) \text{ and } x(\sigma) \in \mathrm{Suf}(v) \setminus \mathrm{Suf}(u)$$

Sound Reduction

Optimal value = 2, Δ = 6.
Sound-reduce u_1 into v_1





Introduced solution is spurious, value = 9

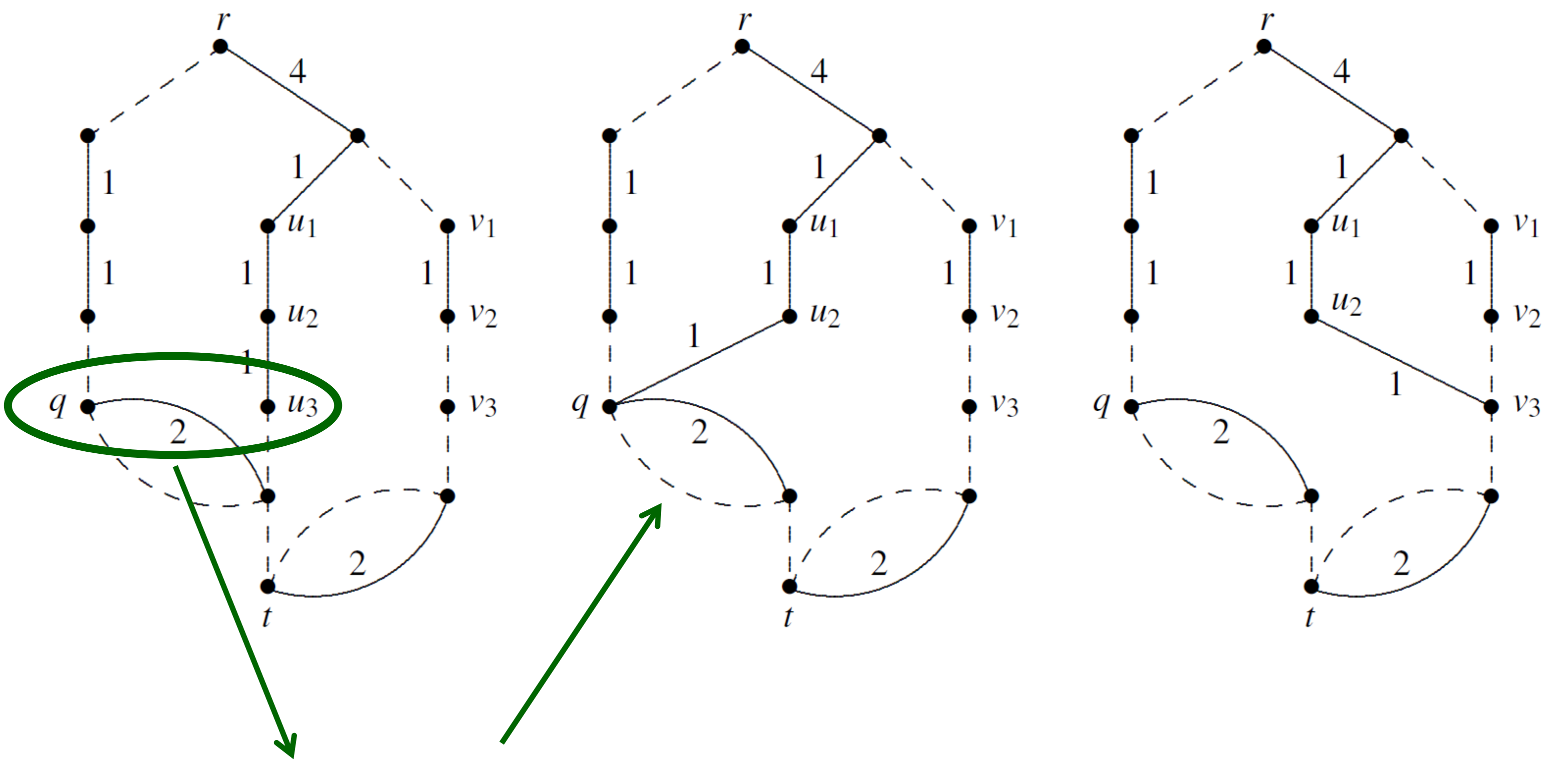
Sound

Theorem. Repeated application of the node merger operation (in any order) yields a **sound reduced** DD – i.e., a sound DD of **minimum size**.

Different reduction orders can yield different diagrams, but **they all have the same size!**

Sound Reduction

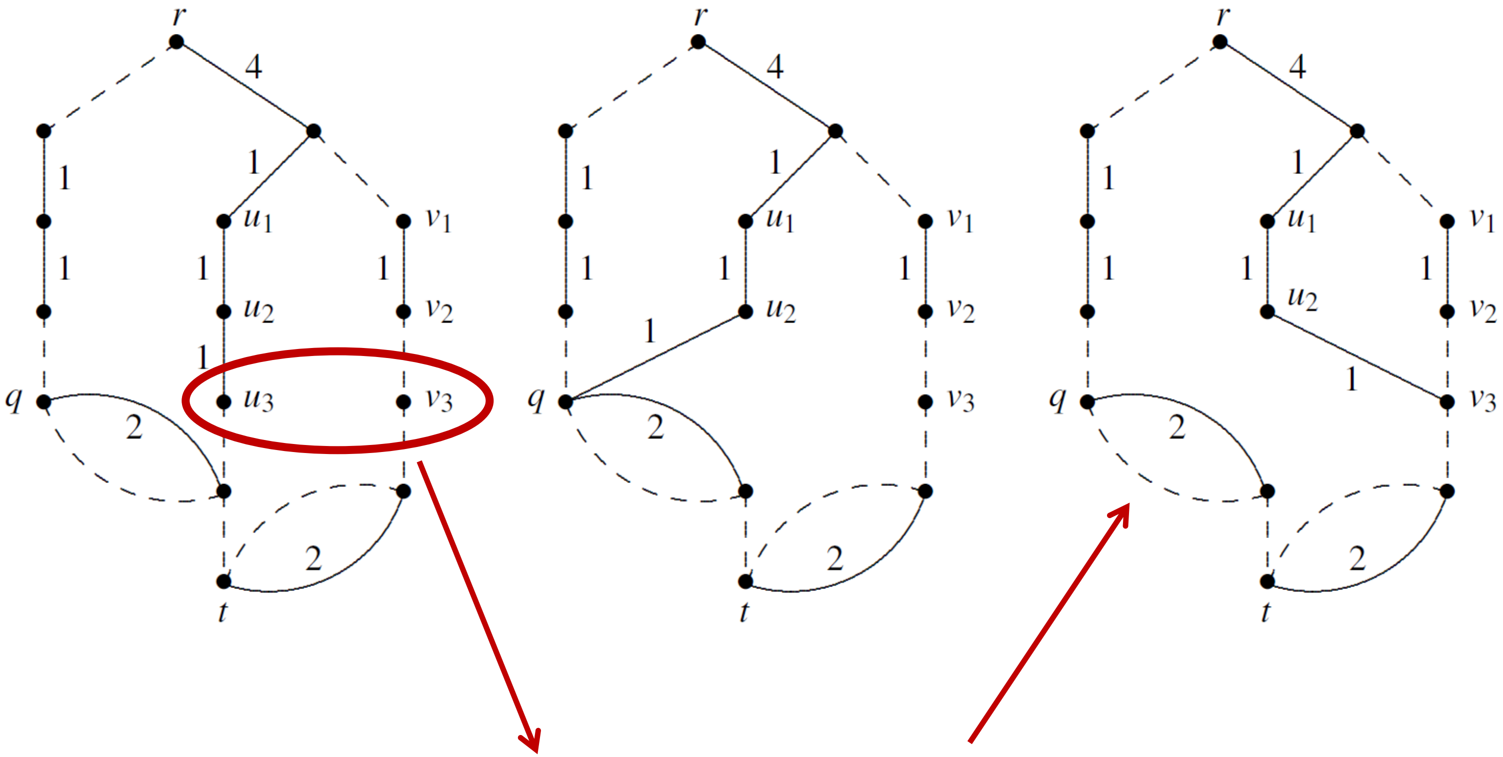
Mergers yield 2 different sound-reduced diagrams,
but of the same size



This merger yields one sound-reduced diagram

Sound Reduction

Mergers yield 2 different sound-reduced diagrams,
but of the same size



This merger yields another, of the same minimum size

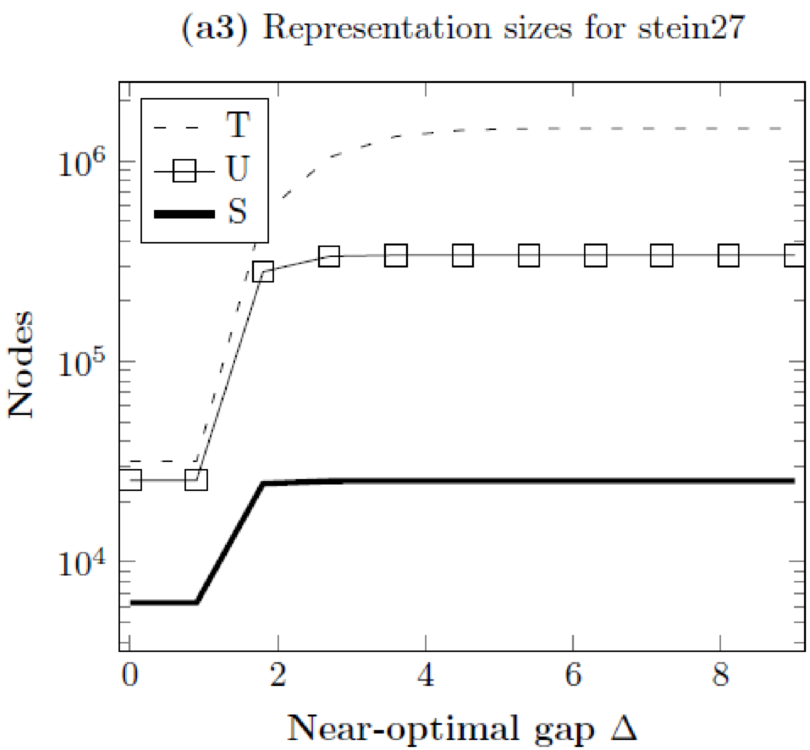
Compression

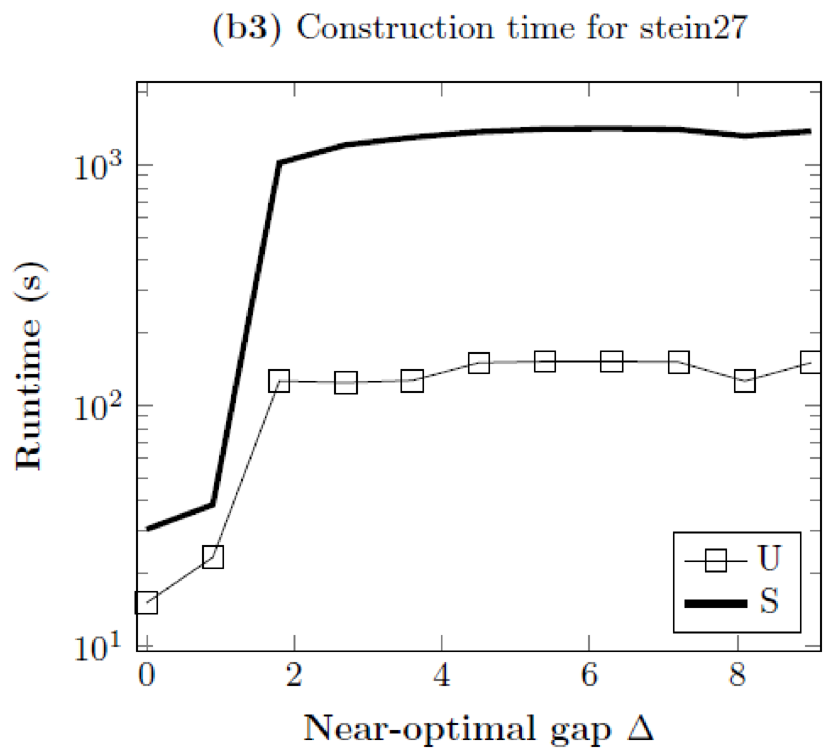
- Sound reduction can significantly compress a diagram that represents near-optimal solutions.
  - We investigate compression **over a range of Δ's**, most larger than needed in practice.
  - Diagram is **generated for one tolerance Δ**.
  - Same diagram used for **multiple queries**, using different tolerances $\delta < \Delta$.
    - For some instances, maximum Δ is large enough to include all feasible solutions.

Compression

- We measure:
  - Size of **tree** representation of Δ-optimal solutions.
  - Size of **reduced DD** representation.
  - Size of **sound-reduced DD** representation.
  - Computation times
    - Including time needed to find Δ-optimal solutions, giuven optimal value from solver.
    - The solver may be able to find the Δ-optimal solutions (e.g., CPLEX).
      - Then only DD-construction is needed, which is very fast.

DD Compression

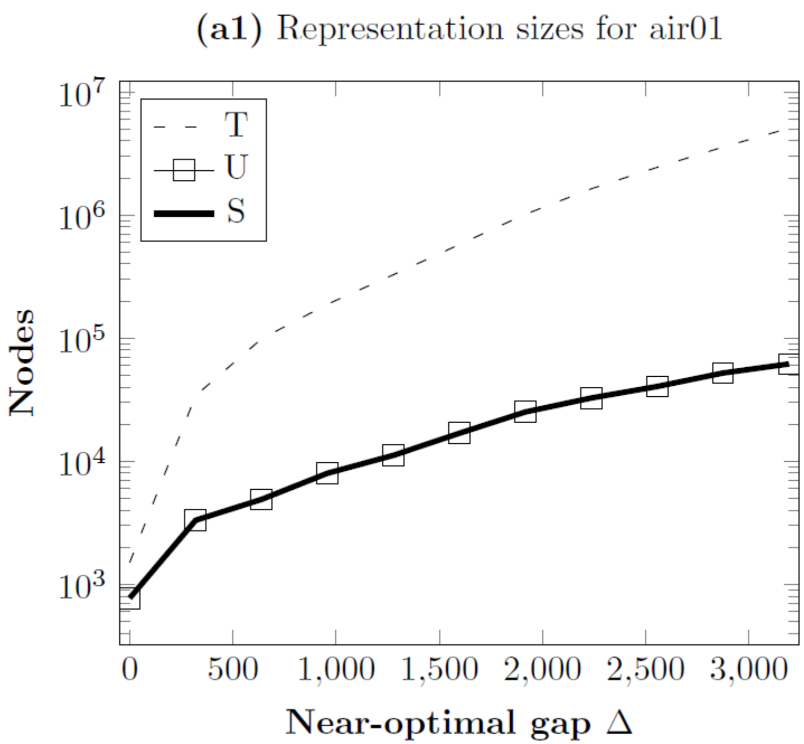


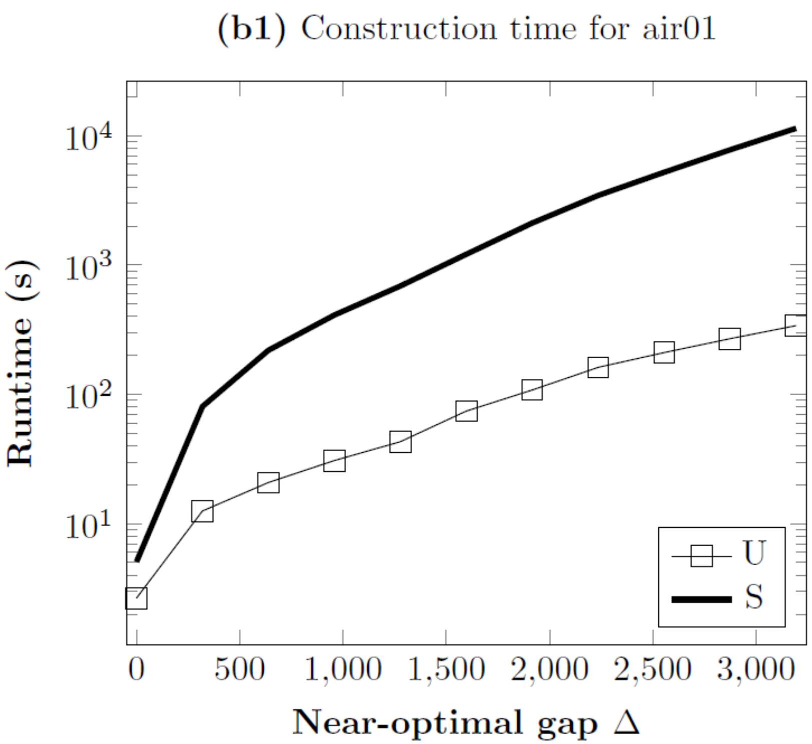


Includes time to find alternate optimal solutions, given optimal value from IP solver

T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression

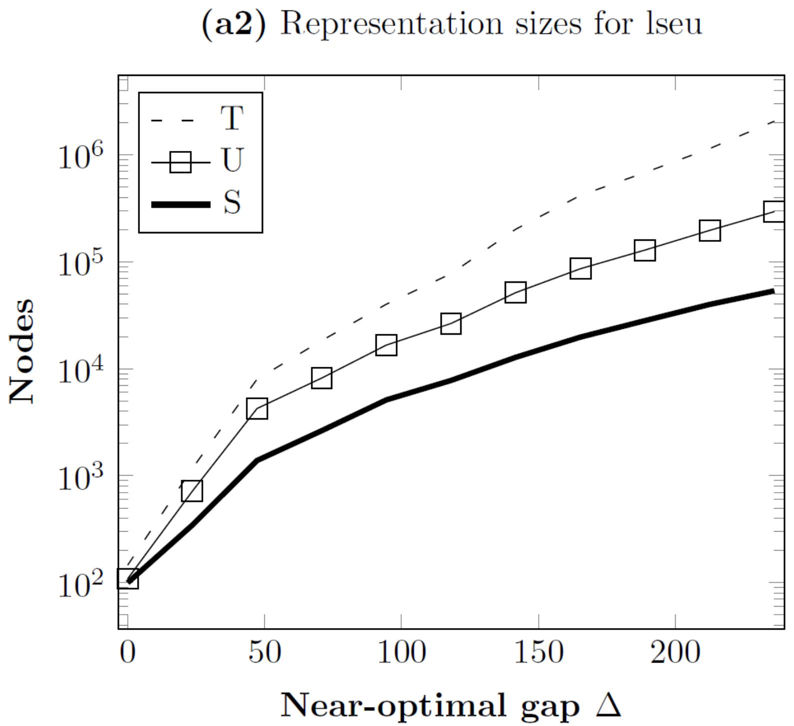


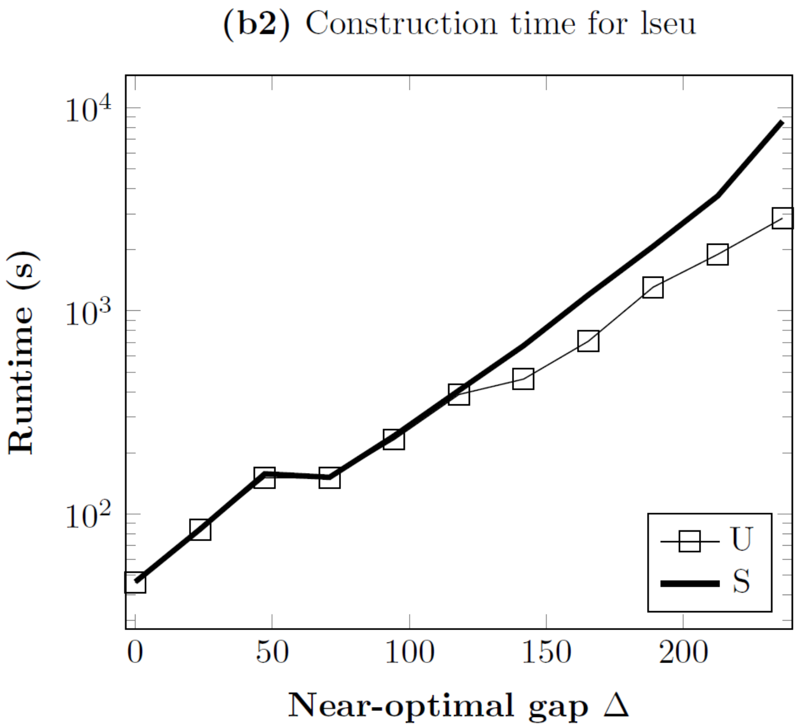


Includes time to find alternate optimal solutions, given optimal value from IP solver

T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression



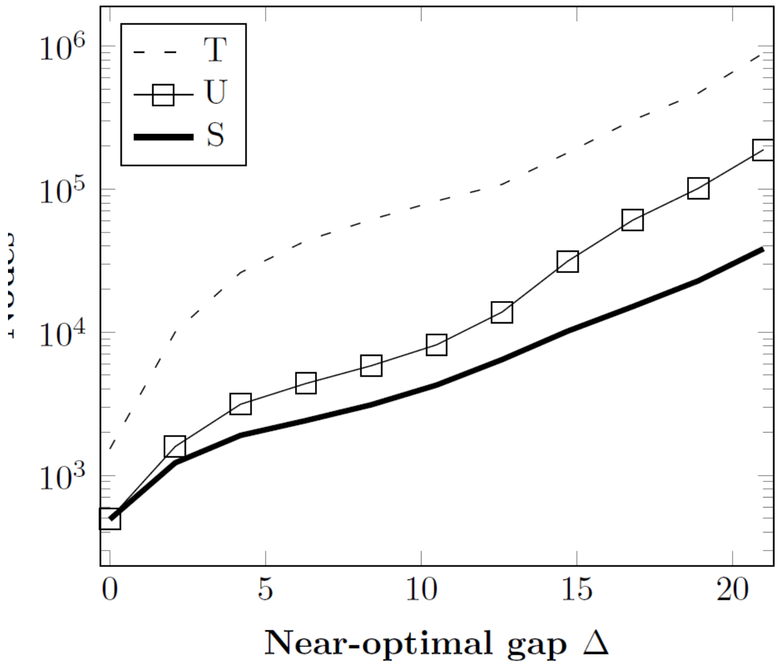


Includes time to find alternate optimal solutions, given optimal value from IP solver

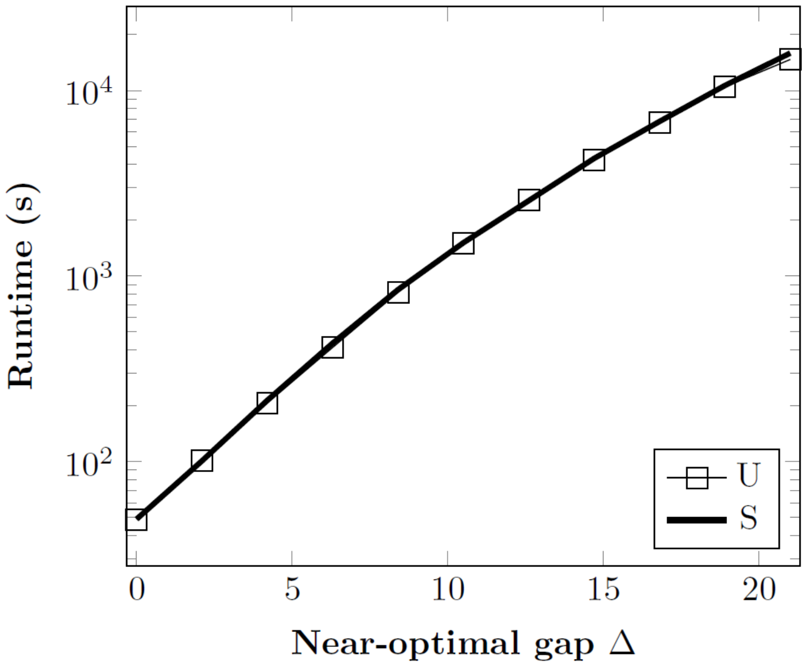
T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression

(a3) Representation sizes for mod008



(b3) Construction time for mod008

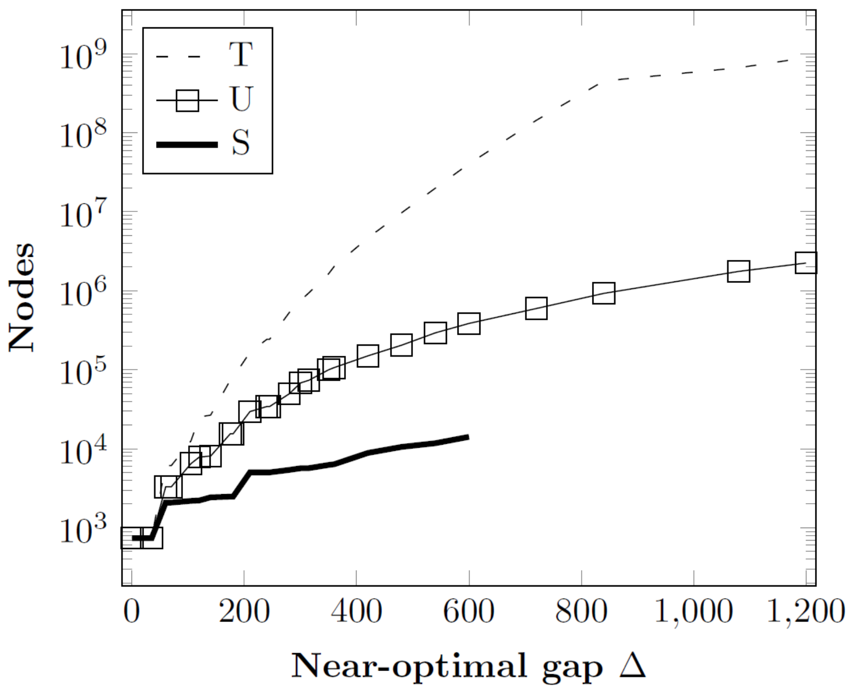


Includes time to find alternate optimal solutions, given optimal value from IP solver

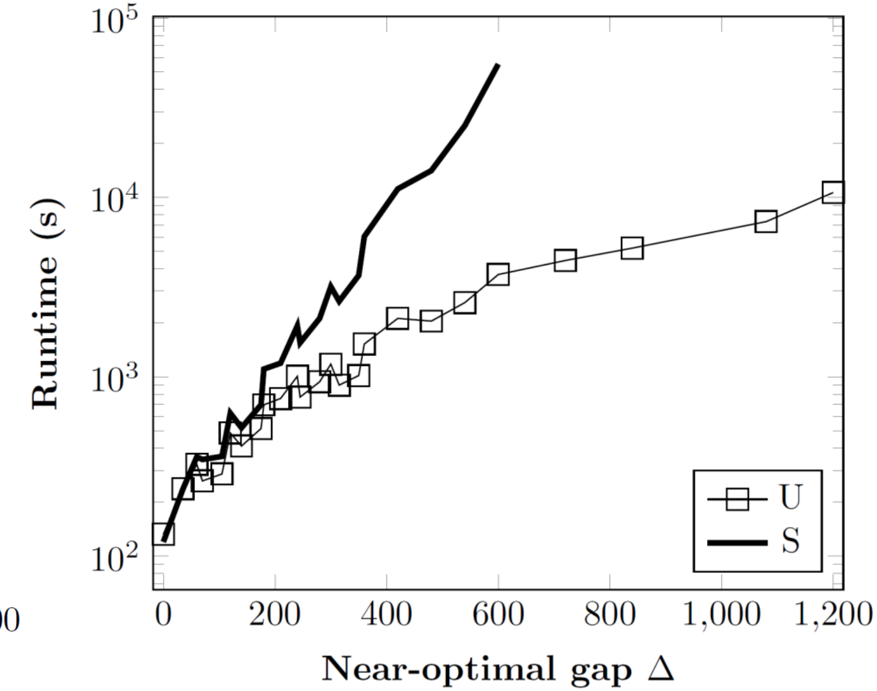
T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression

(a) Representation sizes for p0201



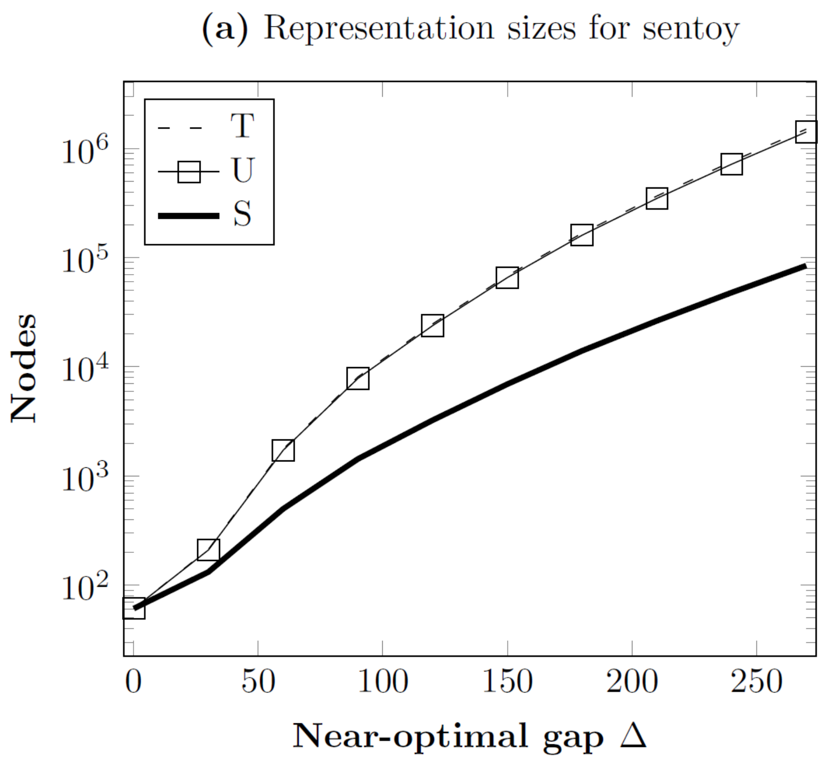
(c) Construction time for p0201

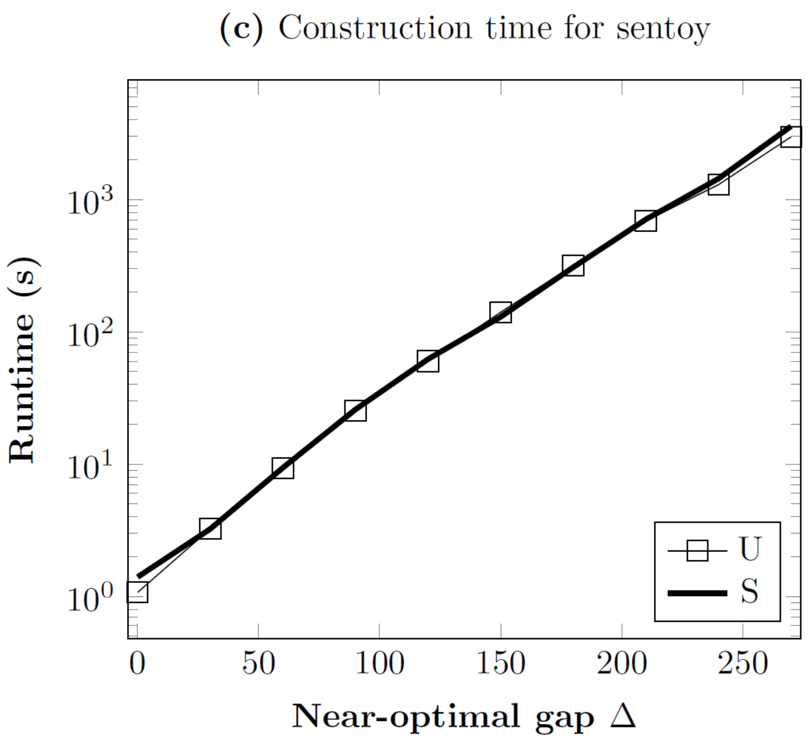


Includes time to find alternate optimal solutions, given optimal value from IP solver

T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression





Includes time to find alternate optimal solutions, given optimal value from IP solver

T = tree representation
U = reduced DD
S = sound-reduced DD

Ongoing Research

- Extend to MILP
  - Represent **integer solutions only** in DD.
  - Path length computed by **solving LP** after fixing integer variables.
  - Scale up by “dualizing” troublesome constraints.
    - This introduces spurious solutions that are easily ignored.

Ongoing Research

- Extend to MILP
  - Represent **integer solutions only** in DD.
  - Path length computed by **solving LP** after fixing integer variables.
  - Scale up by “dualizing” troublesome constraints.
    - This introduces spurious solutions that are easily ignored.
- Next: open-source software
  - Independent of solver.
  - Needs only optimal value and problem formulation.