Integer Programming Postoptimality Analysis Using Decision Diagrams

John Hooker
Carnegie Mellon University
Thiago Serra
Mitsubishi Electric Research Laboratories

INFORMS 2018

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

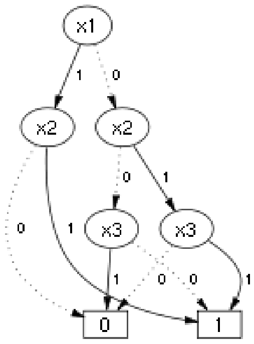
Proposed

Nontransparent data structure

$$Ax \geq b$$

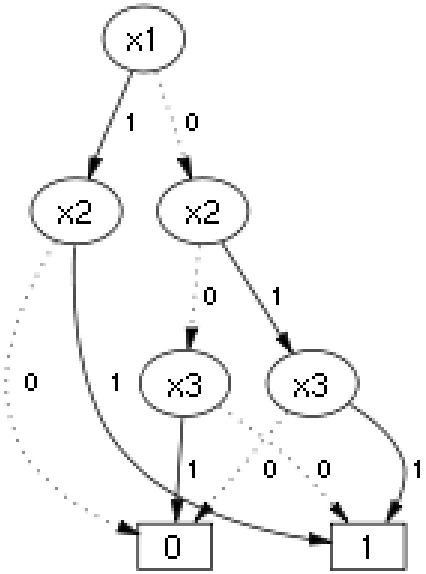
Solver

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.



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 $\delta \leq \Delta$.

Hadžić and JH (2006)

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.

Hadžić and JH (2006)

Sound Decision Diagrams

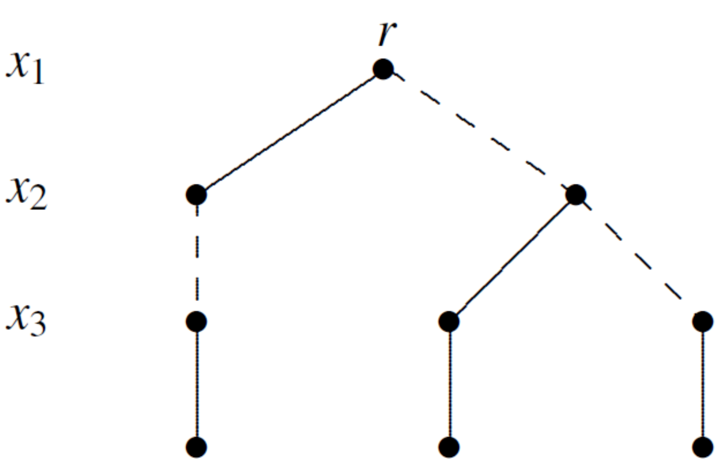
- **Sound** DDs can store near-optimal solutions more compactly.
  - Sound = all Δ-**optimal solutions** are included…
    - …along with some **spurious** solutions (feasible and **infeasible**) that are **worse than Δ-optimal**
      - That is, cost > v^* + Δ.
    - These solutions are easily screened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

Hadžić and JH (2006)

Sound DDs for IP

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

Branching tree

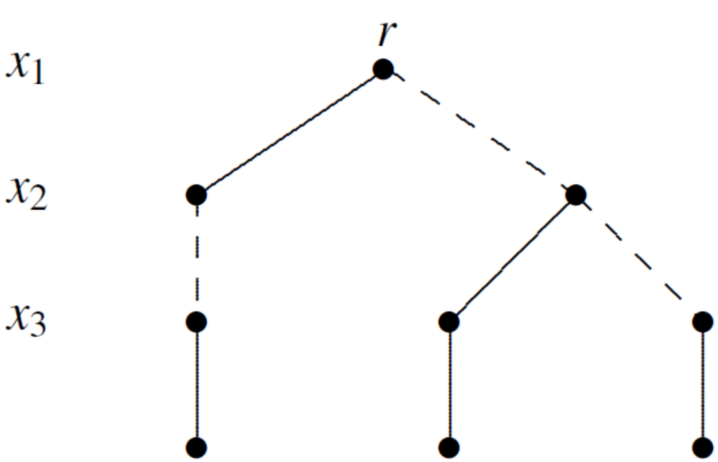


Optimal value = 2

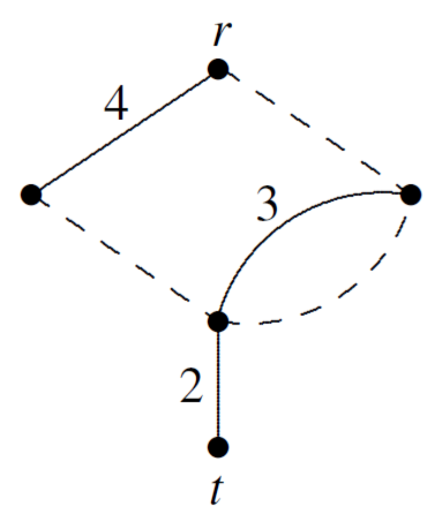
Sound DDs for IP

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

Branching tree



Reduced weighted DD

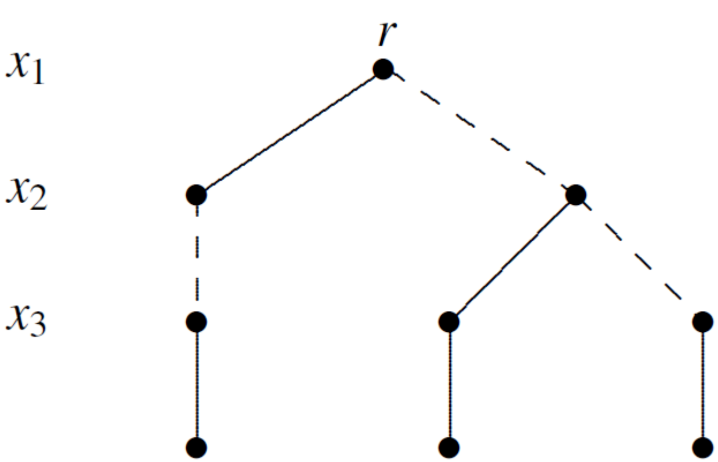


Optimal value = 2

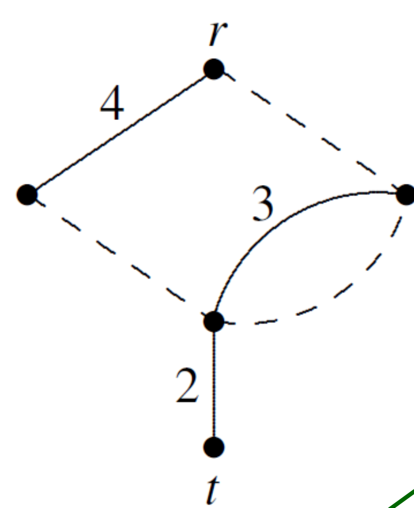
Sound DDs for IP

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

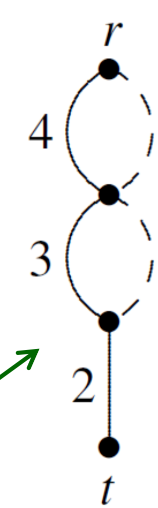
Branching tree



Reduced weighted DD



Sound DD for Δ = 6



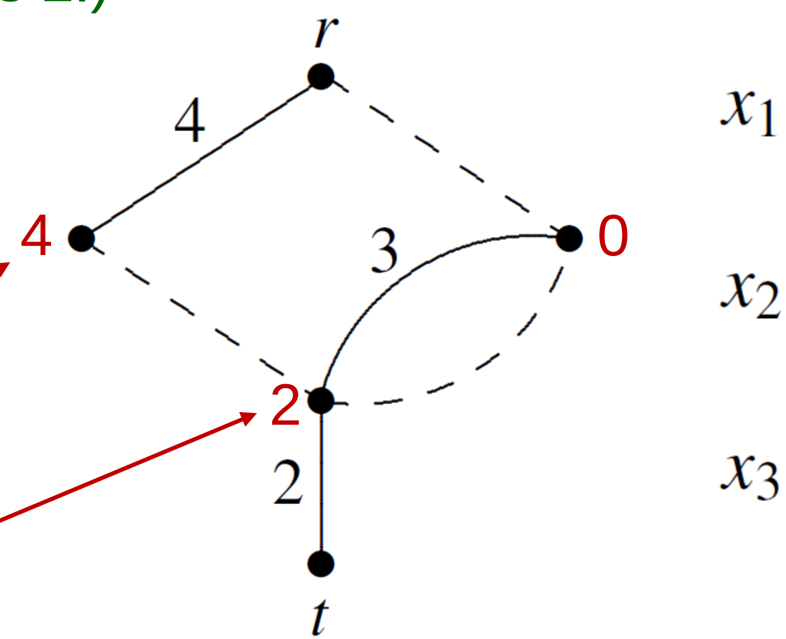
Optimal value = 2

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

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?
(Given that optimal value is 2.)

First use **original** DD
for $\Delta = 6$.



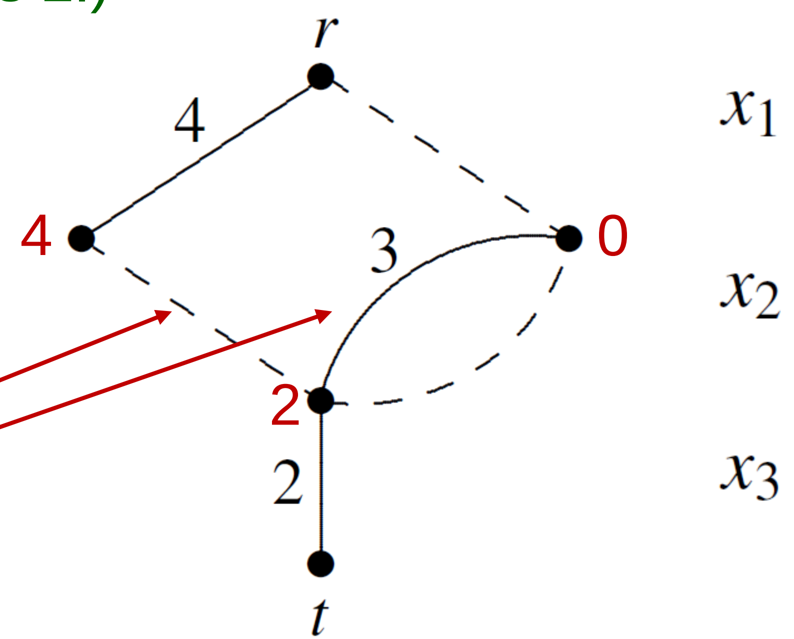
Shortest distance to root

Shortest distance to terminus

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?
(Given that optimal value is 2.)

First use **original** DD
for $\Delta = 6$.

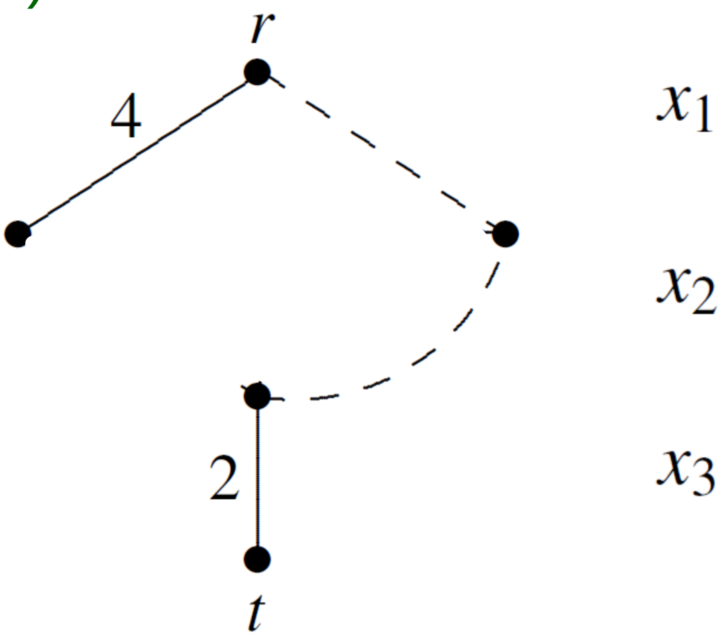


These arcs are removed
because $4 + 0 + 2 > 2 + \delta$
$0 + 3 + 2 > 2 + \delta$

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?
(Given that optimal value is 2.)

First use **original** DD
for $\Delta = 6$.

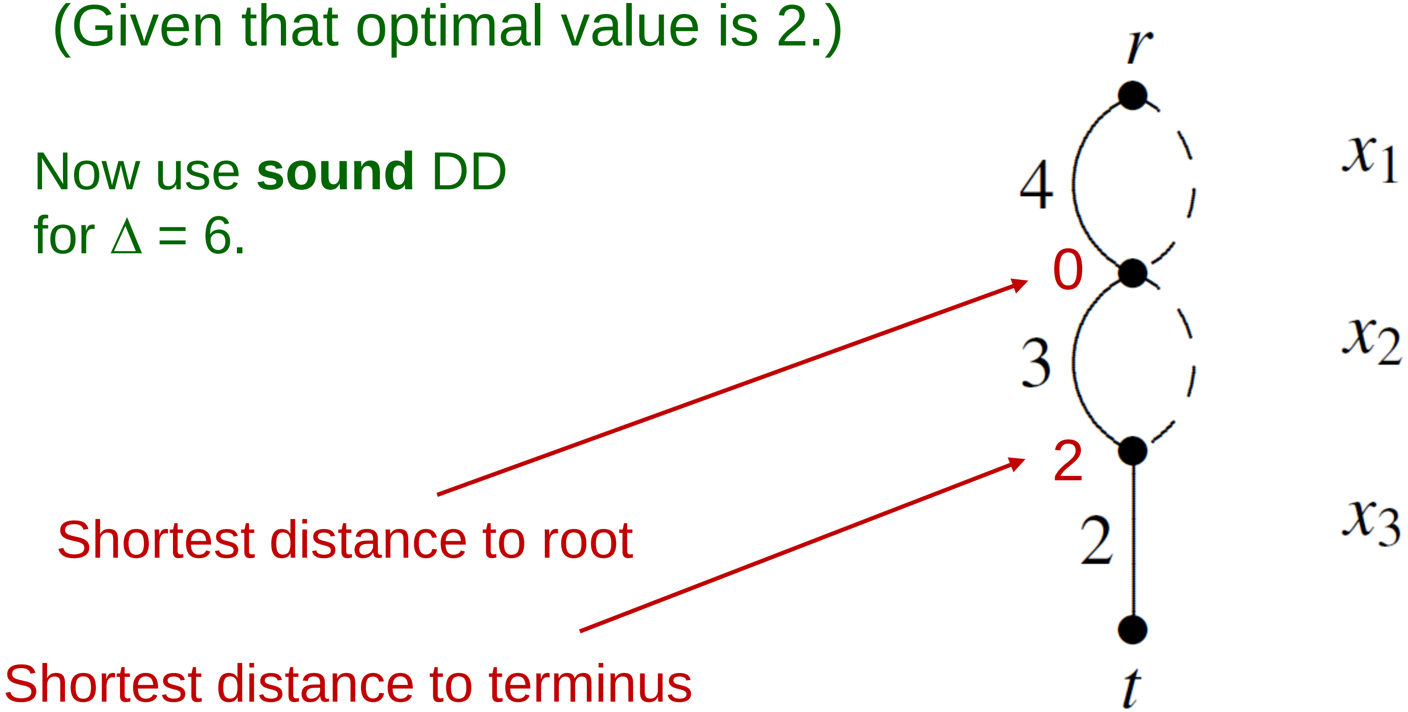


x_2 can take only value 0 when $\delta = 2$.

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?
(Given that optimal value is 2.)

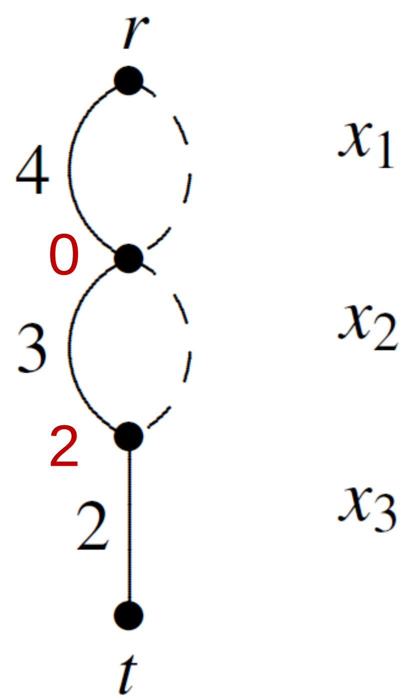
Now use **sound** DD
for $\Delta = 6$.



Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?
(Given that optimal value is 2.)

Now use **sound** DD
for $\Delta = 6$.



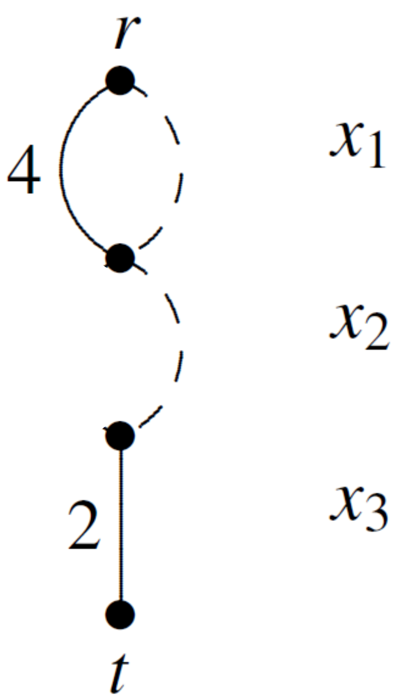
This arc is removed
because $0 + 3 + 2 > 2 + \delta$

Postoptimality Queries

Example: What values can x_2 take when $\delta = 2$?
(Given that optimal value is 2.)

Now use **sound** DD
for $\Delta = 6$.

Spurious solution (1,1,1)
is screened out in the
process.



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

Minimal Sound DDs

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

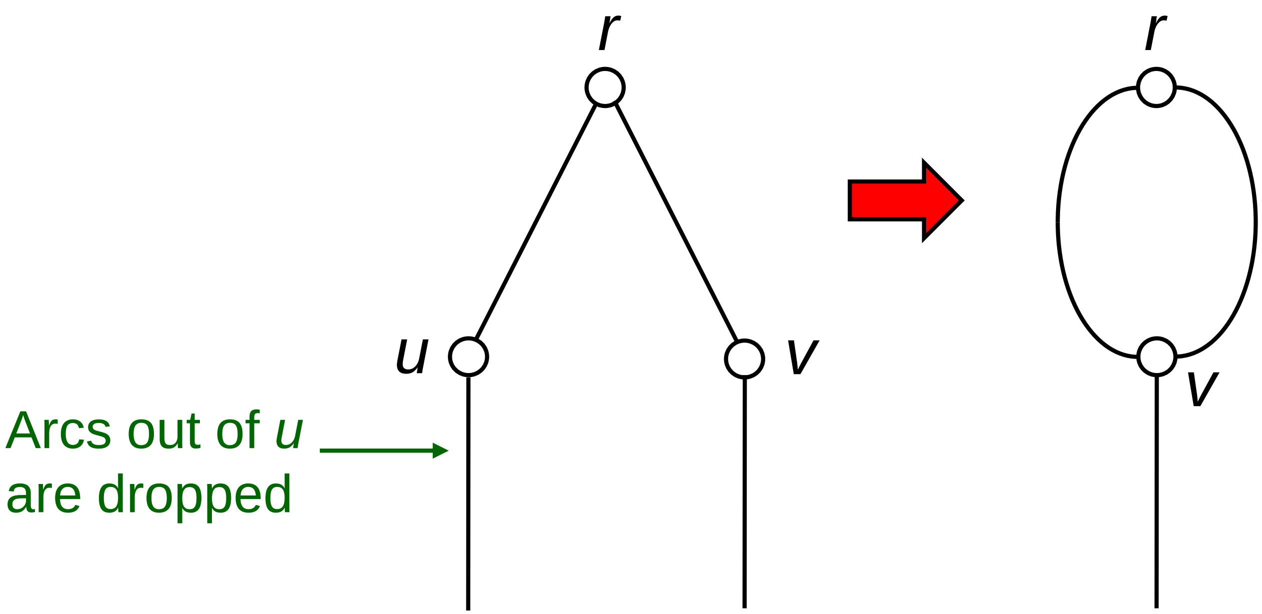
Theorem. A minimal sound DD for $\Delta = 0$ never contains spurious solutions.

So there is no point in using sound DDs for **optimal solutions only**. (Not so for MIP.)

Serra and JH (2018)

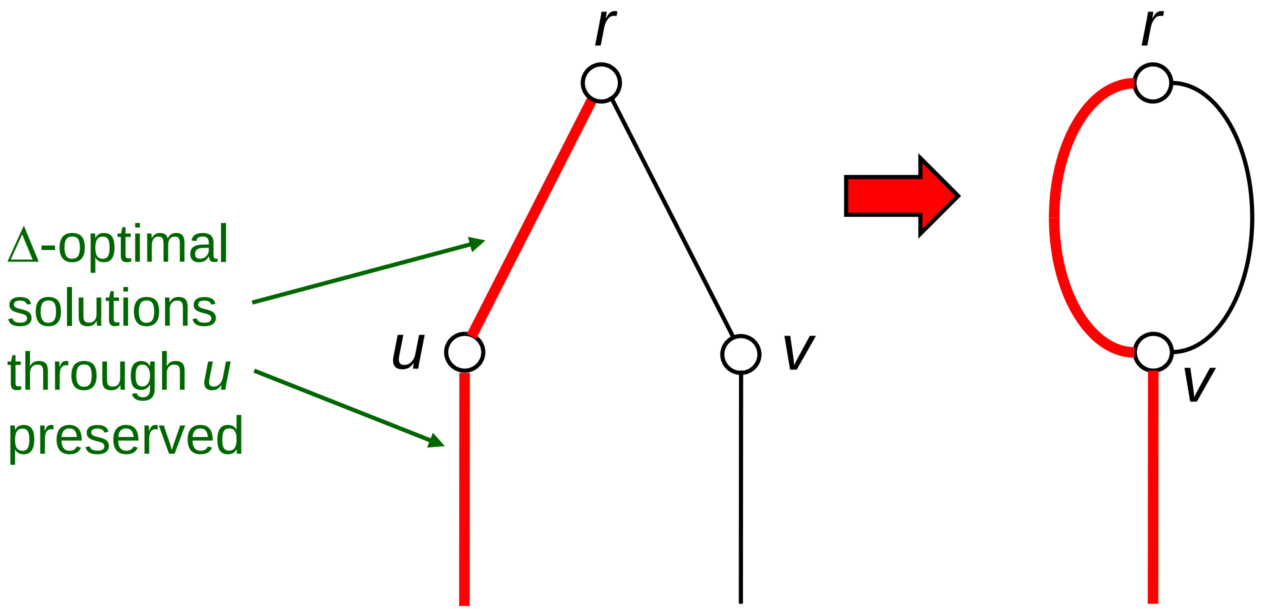
Sound Reduction

- We can **sound reduce** node ***u*** into node ***v*** when this removes no Δ-optimal solutions and introduces only spurious solutions.
  - Can be checked recursively while building diagram.



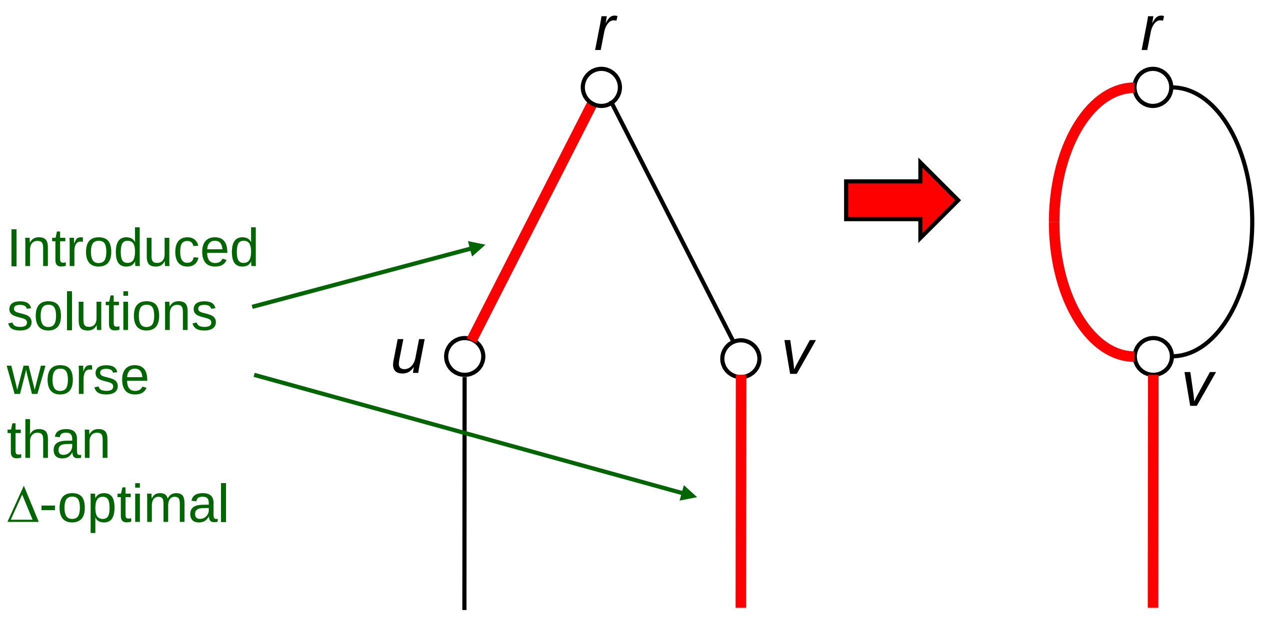
Sound Reduction

- We can **sound reduce** node ***u*** into node ***v*** when this removes no Δ-optimal solutions and introduces only spurious solutions.
  - Can be checked recursively while building diagram.



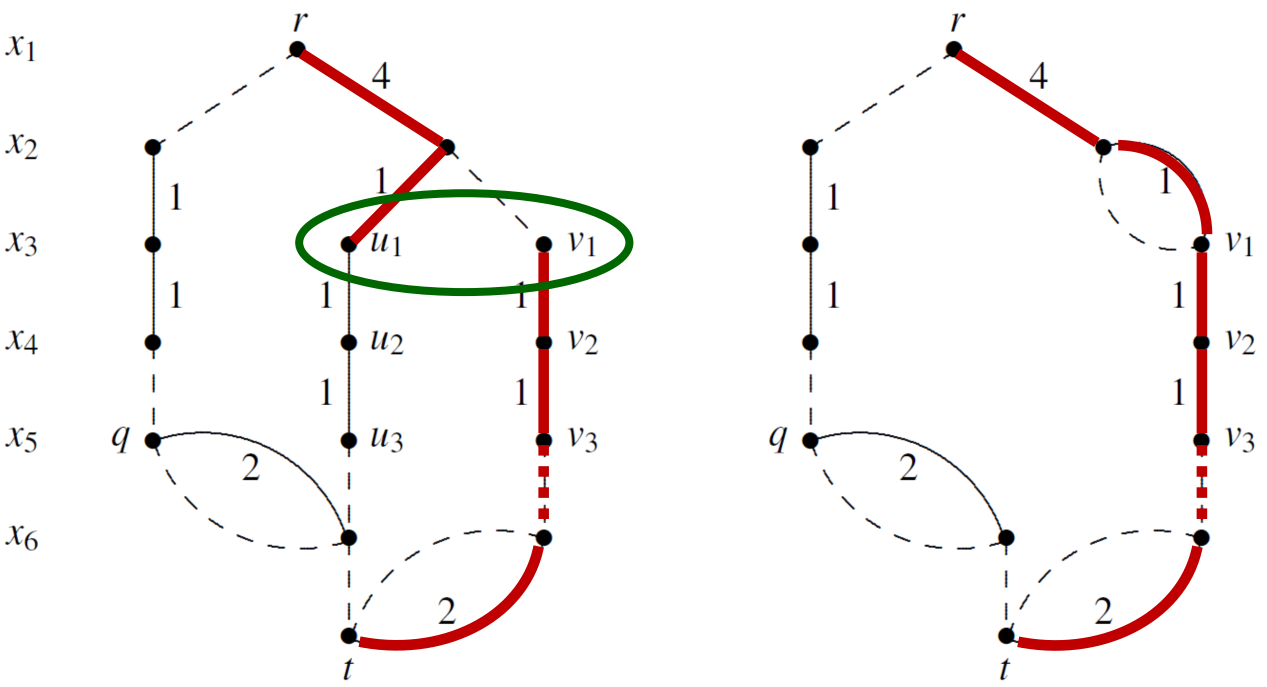
Sound Reduction

- We can **sound reduce** node ***u*** into node ***v*** when this removes no Δ-optimal solutions and introduces only spurious solutions.
  - Can be checked recursively while building diagram.



Sound Reduction

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



Introduced solution is spurious, value = 9

Sound Reduction

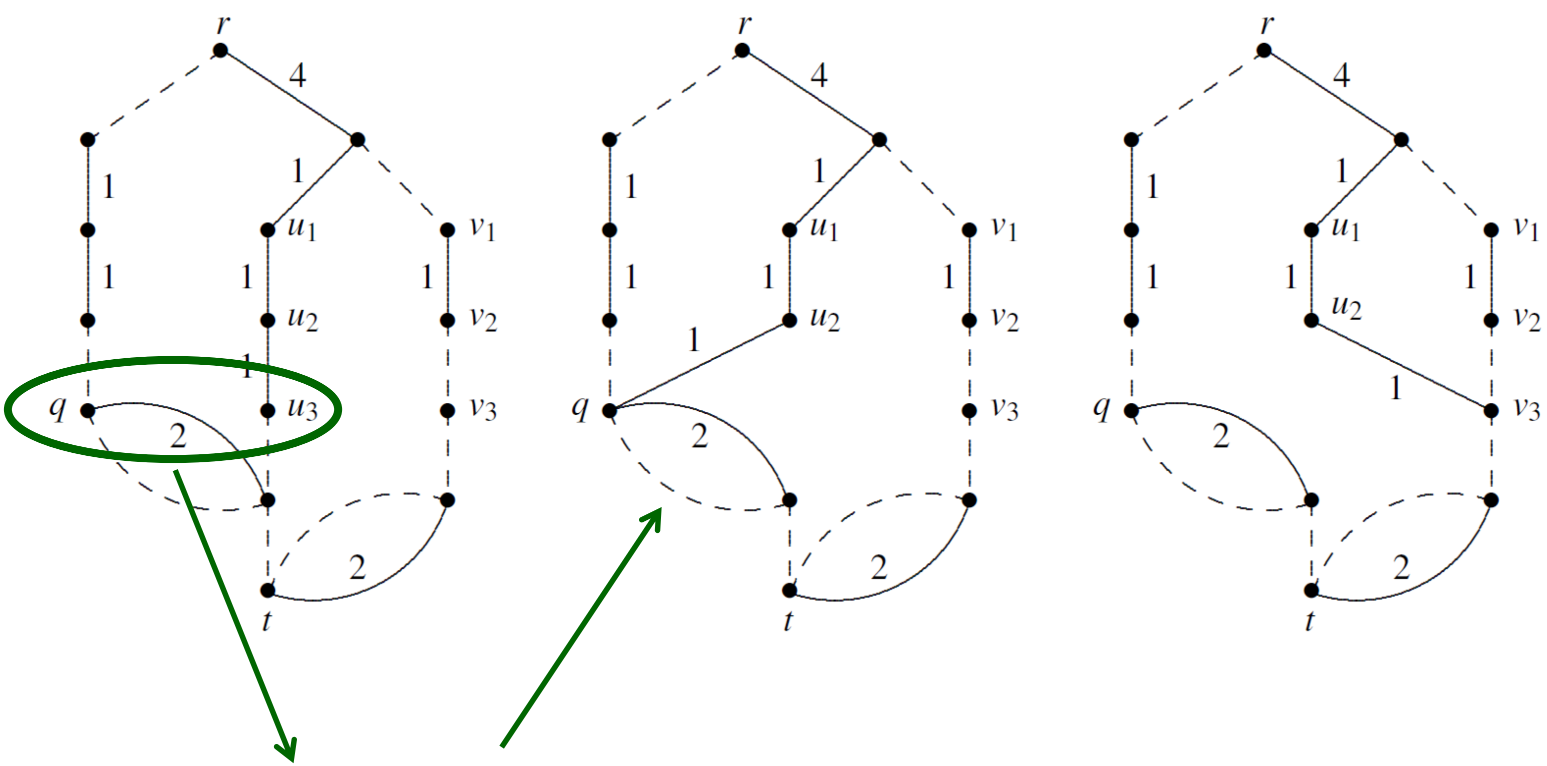
Theorem. Repeated application of the sound reduction operation (in any order) yields a **sound** DD of **minimum size**.

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

Serra and JH (2018)

Sound Reduction

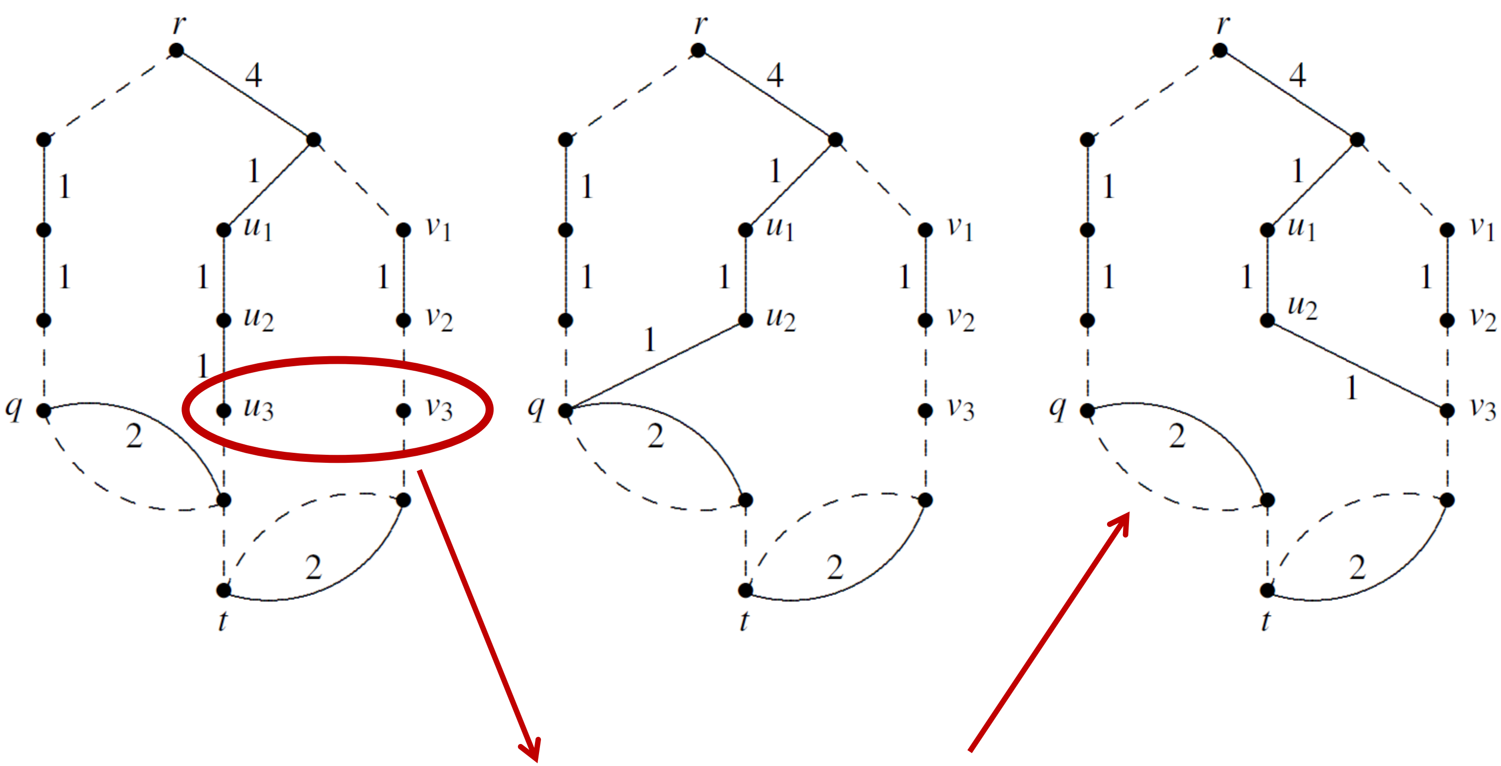
Mergers yield 2 different sound-reduced diagrams,
but of the same size



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

Building a Sound DD

- Two options:
- Stand-alone approach
  - Build the sound DD and identify Δ-optimal solutions simultaneously.
  - Use branching search with backtracking.
    - Use only the optimal value, obtained from a solver.
    - Identify nodes and sound-reduce nodes when possible.
- Solver-assisted approach
  - Obtain Δ-optimal solutions from a solver.
  - Use similar backtracking algorithm, without search for solutions.

Building a Sound DD

Technical conditions for sound-reducing *u* into *v*:

Shortest *r*–*u* distance

$$\boxed{w(r,u)} + \mathrm{LCDS}(u,v) > \boxed{z^*} + \Delta$$

$$w(r,u) + \boxed{\mathrm{LCDS}}(v,u) > z^* + \Delta$$

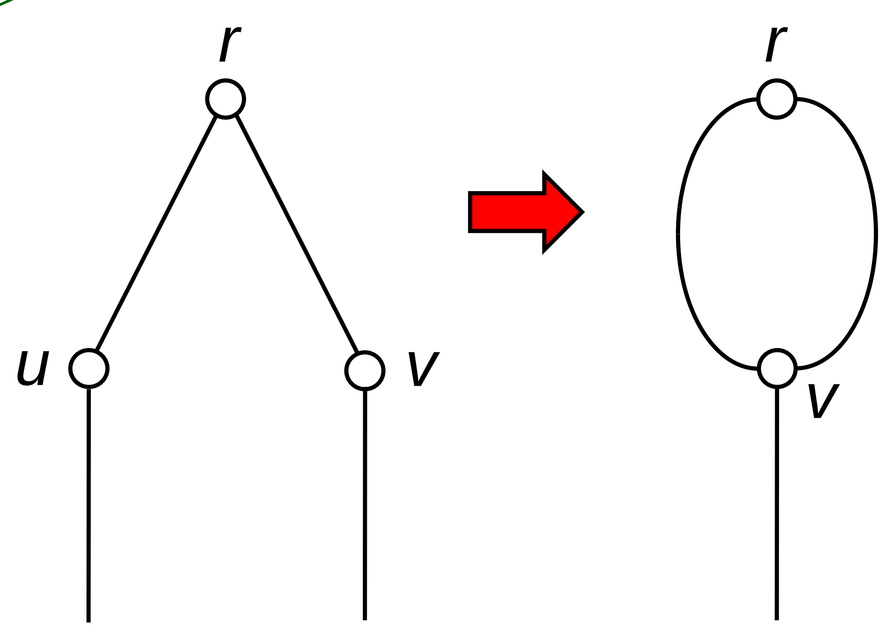
Optimal value

Least-cost differing suffix
= min length suffix of *v*
that is not a suffix of *u*

Suffix = path to terminus

Computed recursively
while backtracking.

LCDS is known when we
have backtracked to both
u and v.



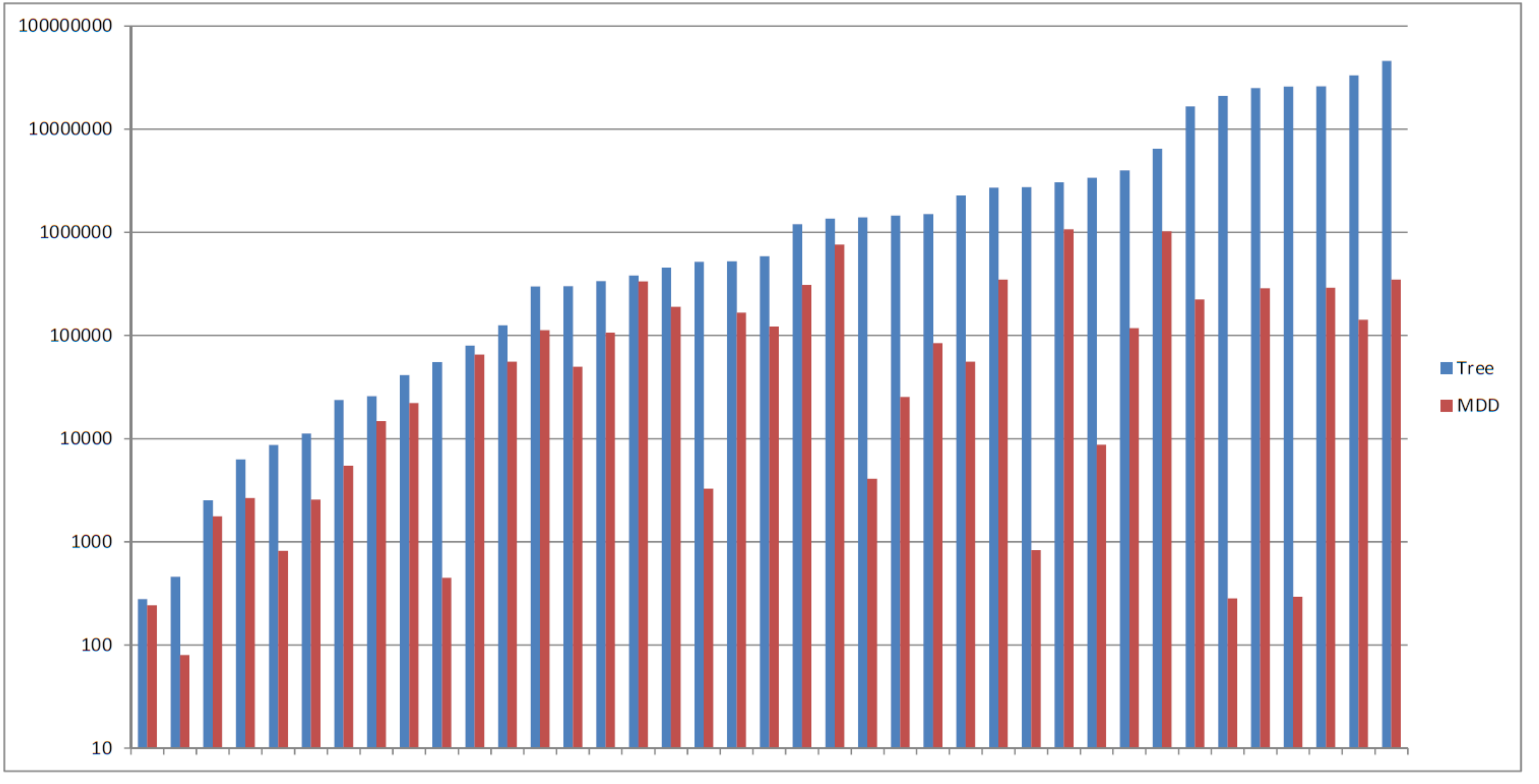
Compression

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

MDD Compression

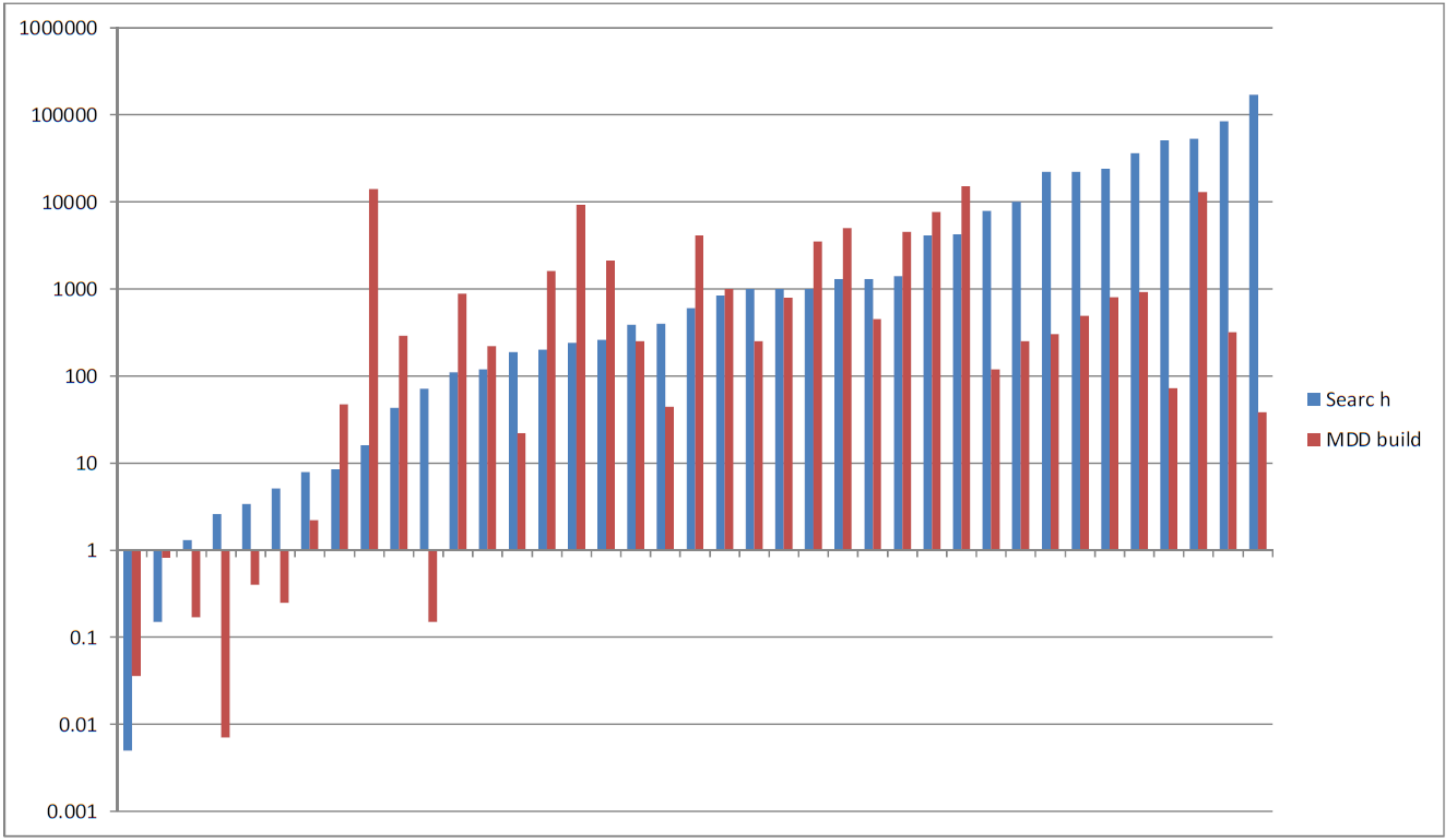
Tree size & sound-reduced DD size for large Δ

39 IP instances from MIPLIB 3.0 and MIPLIB 2010

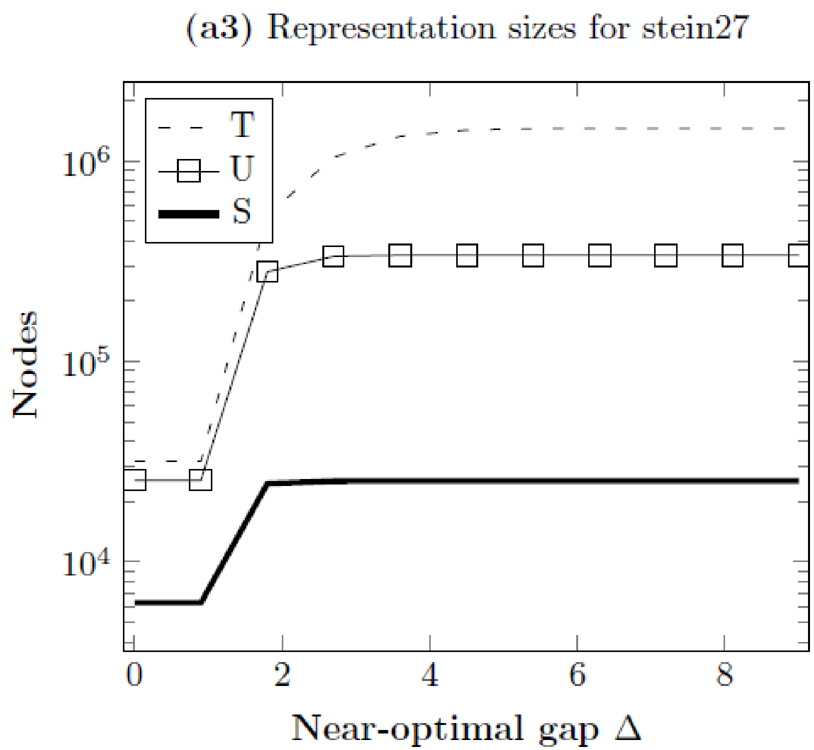


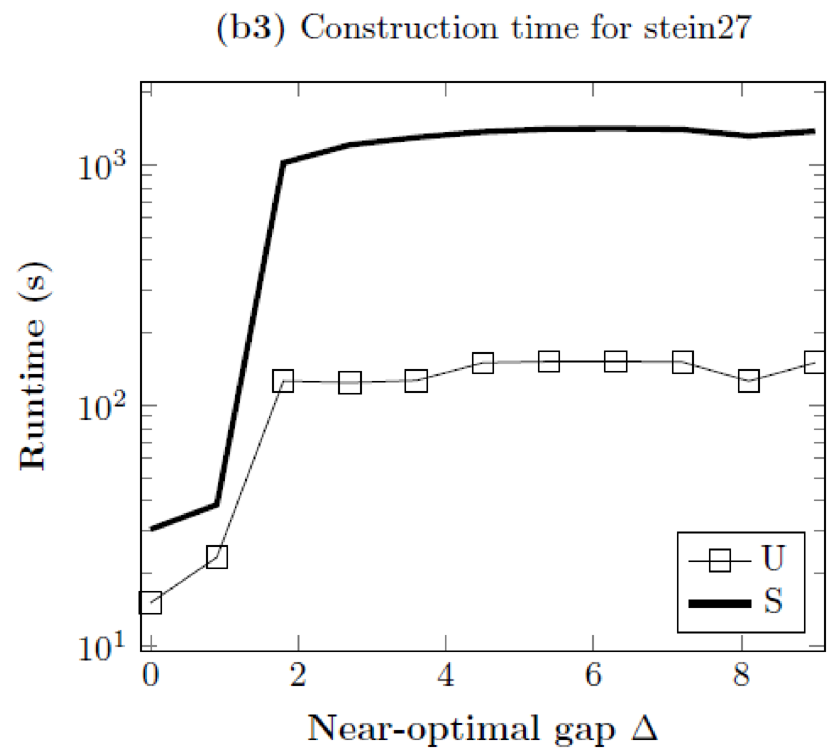
Search & Compression Time

CPLEX search time & DD build time (sec) for large Δ



MDD Compression & Time vs Δ

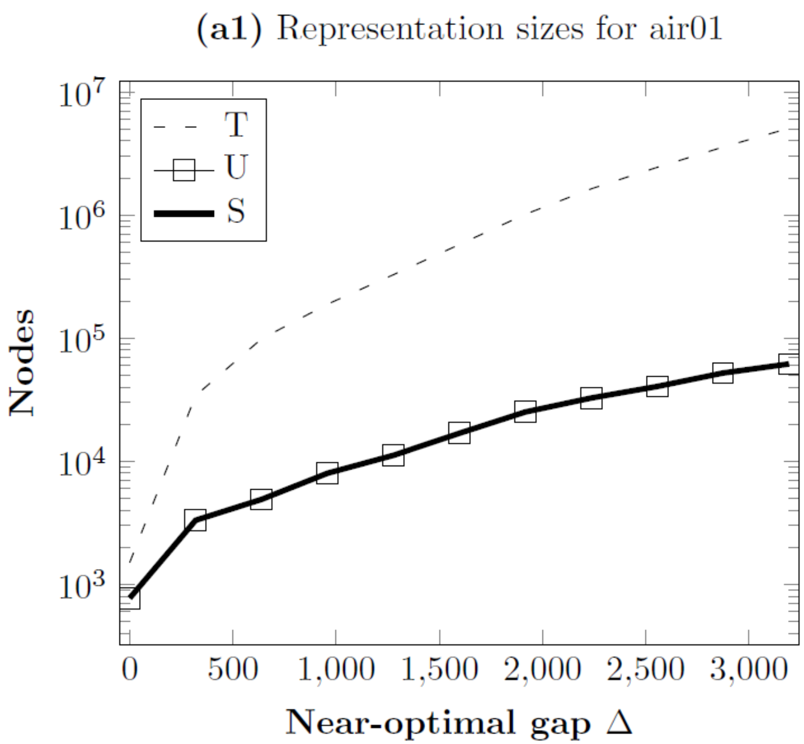


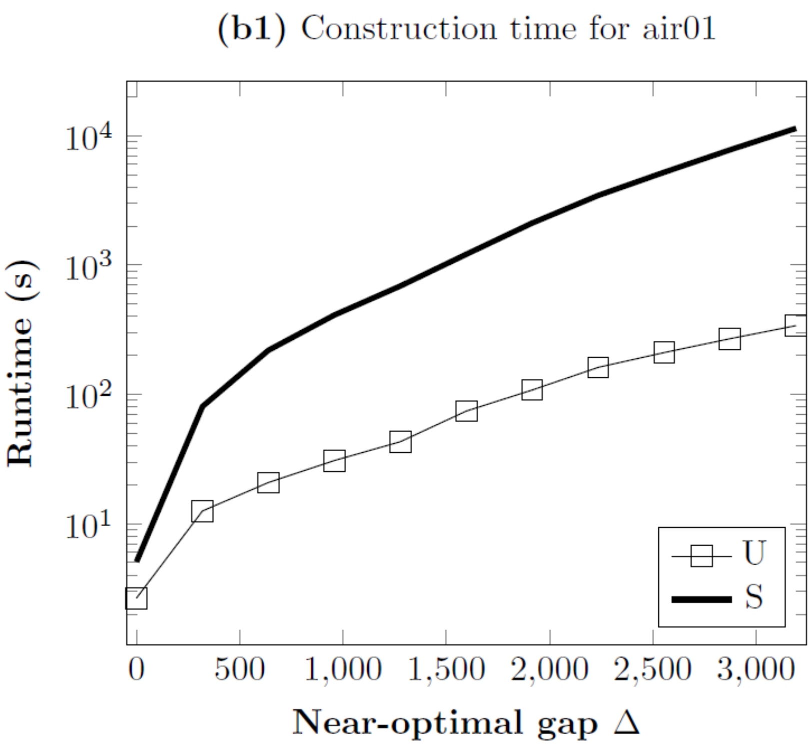


Stand-alone method

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

MDD Compression & Time vs Δ

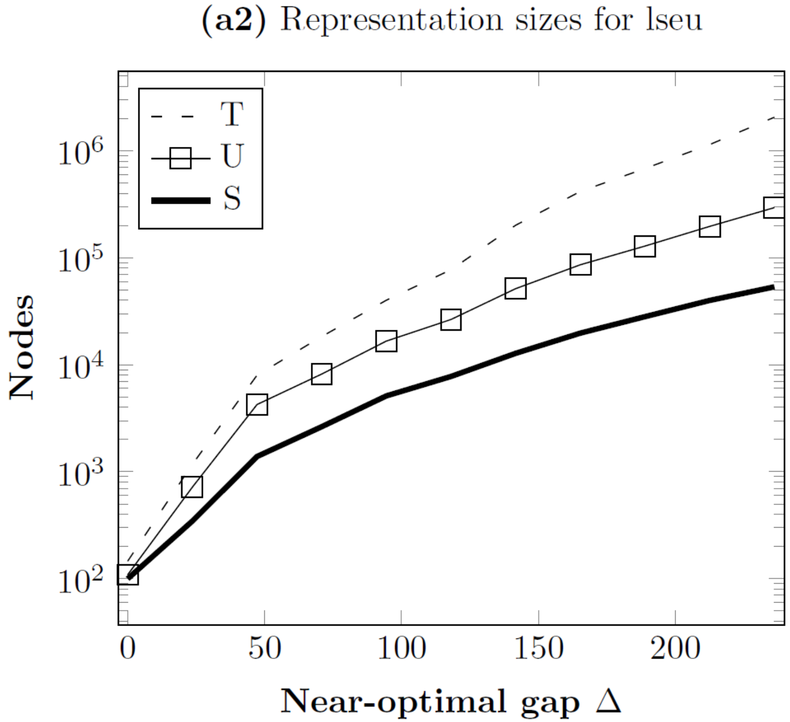


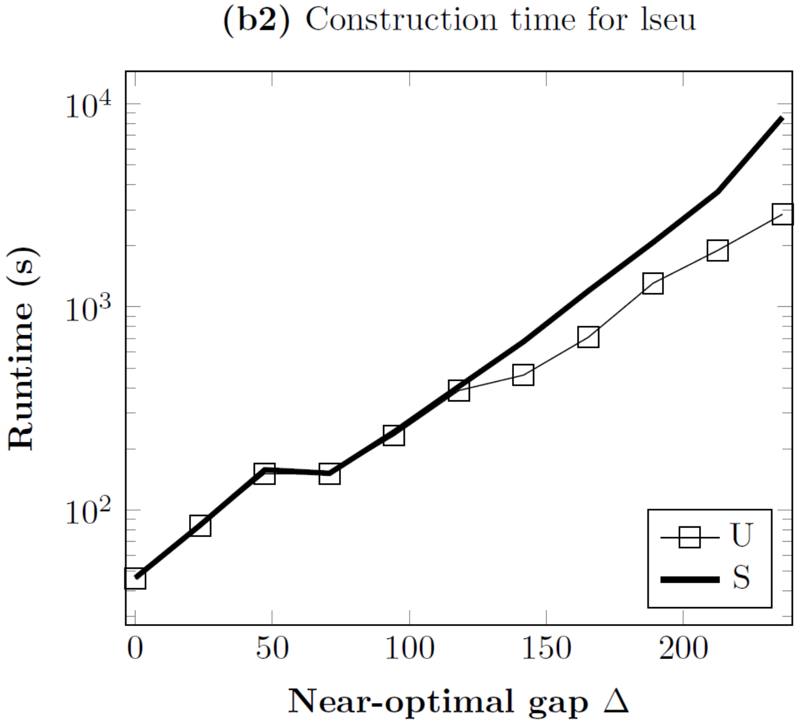


Stand-alone method

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

MDD Compression & Time vs Δ

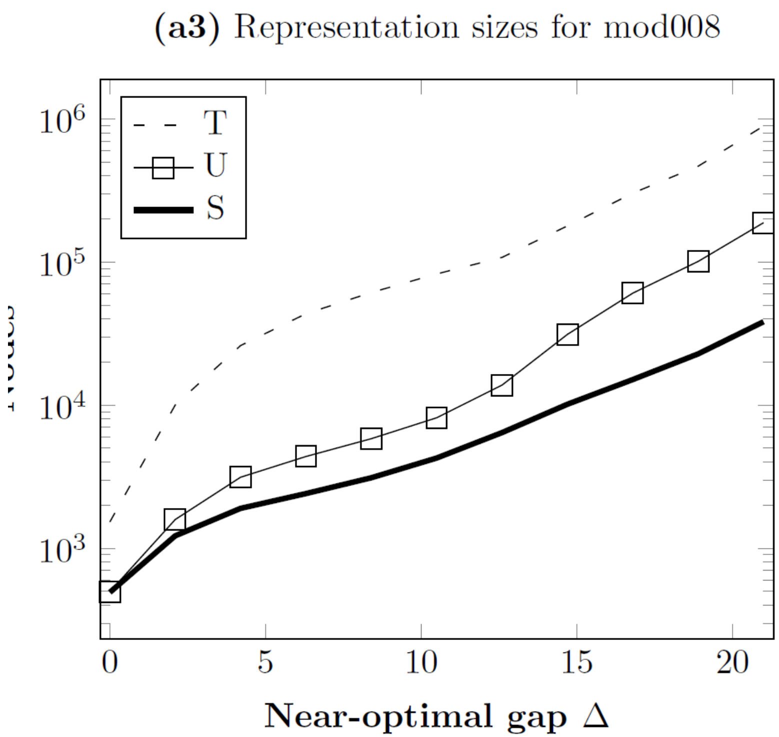


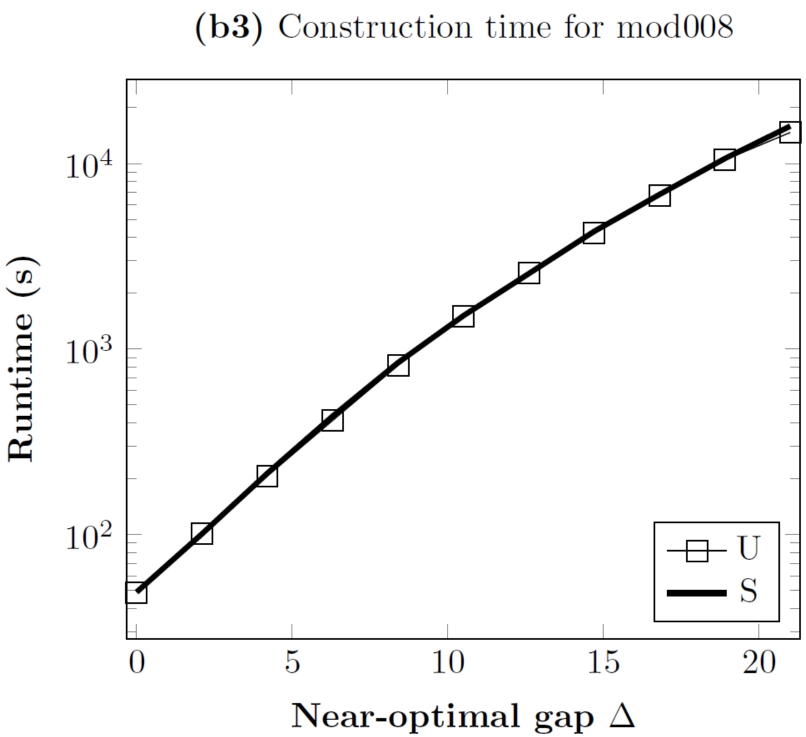


Stand-alone method

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

MDD Compression & Time vs Δ



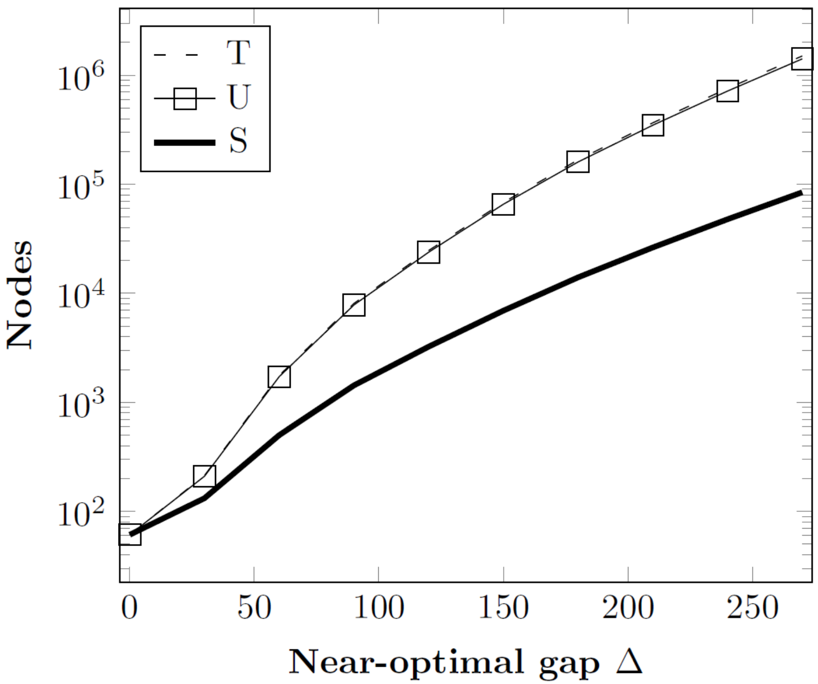


Stand-alone method

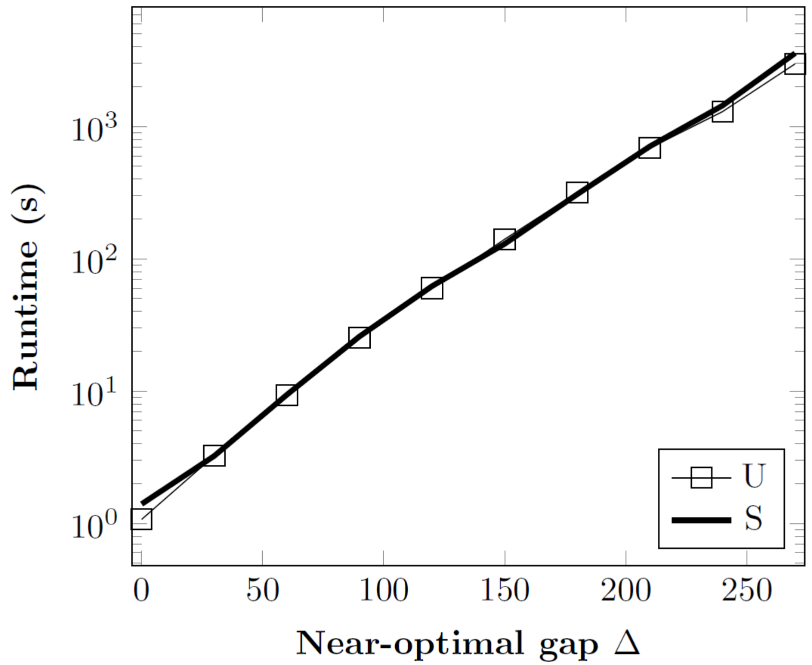
T = tree representation
U = reduced DD
S = sound-reduced DD

MDD Compression & Time vs Δ

(a) Representation sizes for sentoy



(c) Construction time for sentoy



Stand-alone method

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

Research Issues

- Extension to MILP
  - Under development
- Applications to:
  - Multiobjective optimization
    - Original application!
  - General mixed discrete/continuous programming
    - Not just MILP
- How to combine DD-based solution with DD-based postoptimality?
  - As in "**1-tree**" method for MILP