MDD-based Postoptimality Analysis for Mixed-integer Programs

John Hooker, Ryo Kimura
Carnegie Mellon University
Thiago Serra
Mitsubishi Electric Research Laboratories

Symposium on Decision Diagrams for Optimization (DDOPT)
Carnegie Mellon University, October 2018

Two Perspectives on Optimization

Traditional

Nontransparent data structure

$$Ax \geq b$$

Solver $\Rightarrow$

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

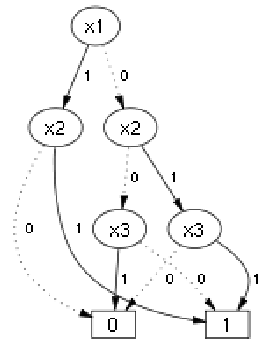
Proposed

Nontransparent data structure

$$Ax \geq b$$

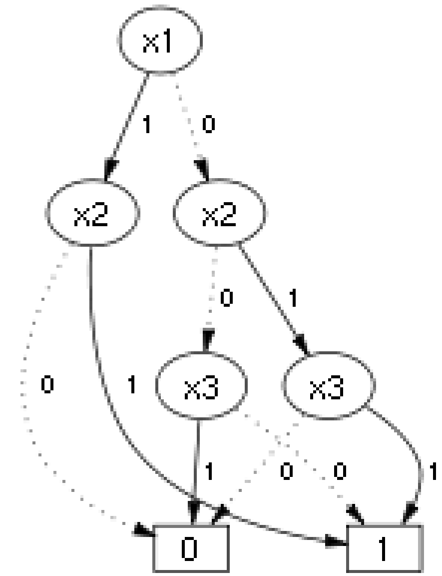
Solver

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.



Outline

- **Basic concepts**
- **Pure integer programming**
  - Sound diagrams for IP
  - Postoptimality analysis using sound DDs
  - Sound reduction
  - Computational results
- **Mixed-integer programming**
  - Sound diagrams for MILP
  - Identifying equivalent states
    - By computation of equivalency ranges
    - Arc deletion and contraction
    - Separable constraints
  - Introducing spurious solutions
    - By constraint dualization
    - By sound reduction

Near-optimal Solutions

- Let z^* = optimal cost.
- A solution is **Δ-optimal** if it is feasible and its cost is $\leq z^* + \Delta$
  - 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 > z^* + Δ.

Hadžić and JH (2006)

Sound Decision Diagrams

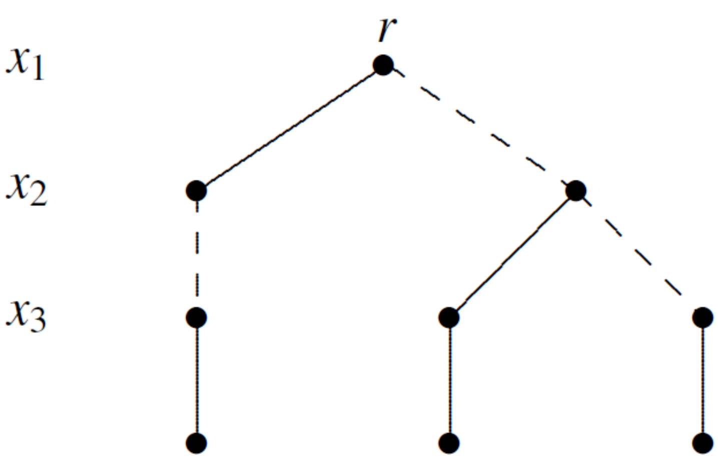
- **Sound** DDs can store near-optimal solutions more compactly.
  - Sound = all Δ-**optimal solutions** are included…
    - …along with some **spurious** solutions (feasible and **infeasible**) that are **worse than Δ-optimal**
      - That is, cost > z^* + Δ.
    - These solutions are easily screened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

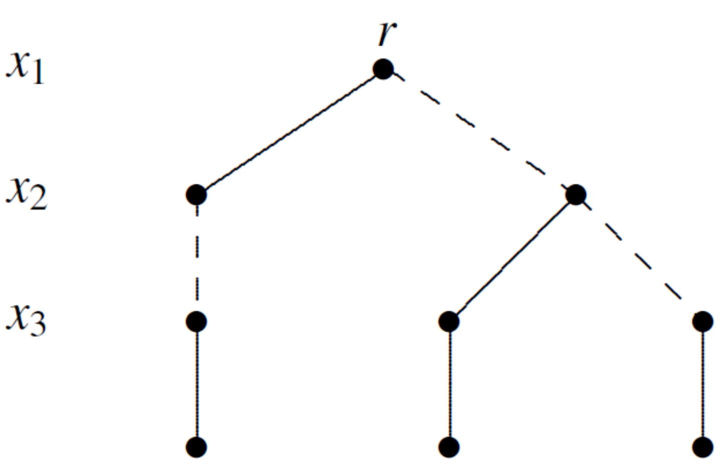


Optimal value = 2

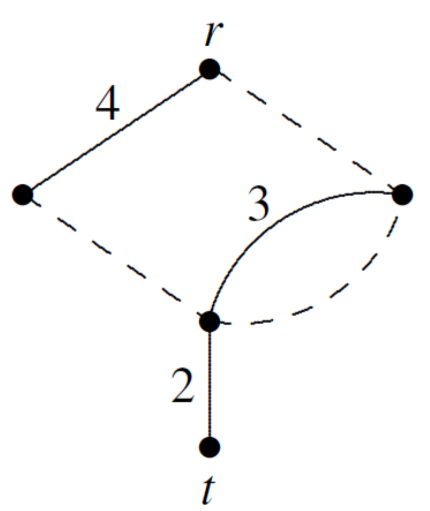
Sound DDs for IP

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

Branching tree



Reduced weighted DD

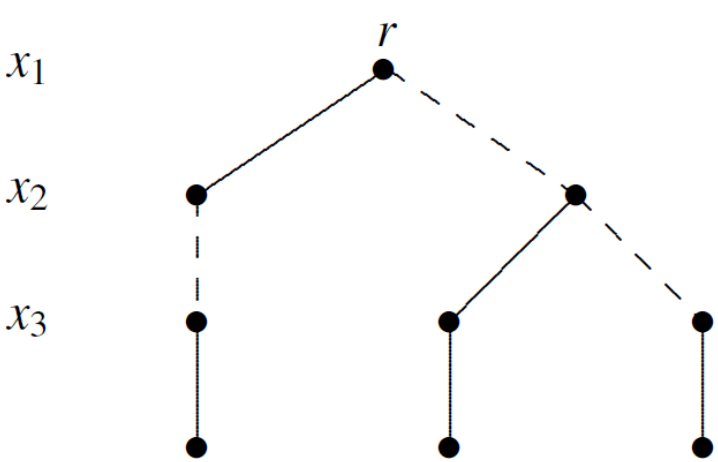


Optimal value = 2

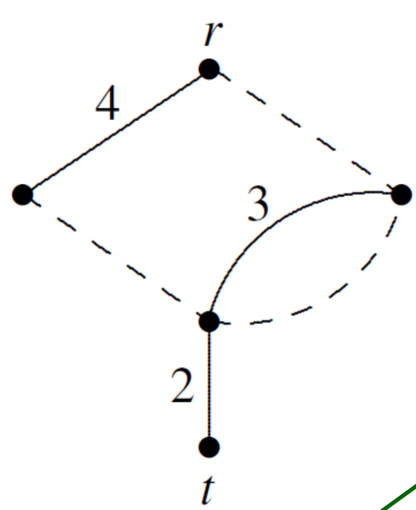
Sound DDs for IP

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

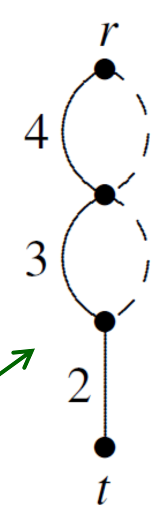
Branching tree



Reduced weighted DD



Sound DD for Δ = 6



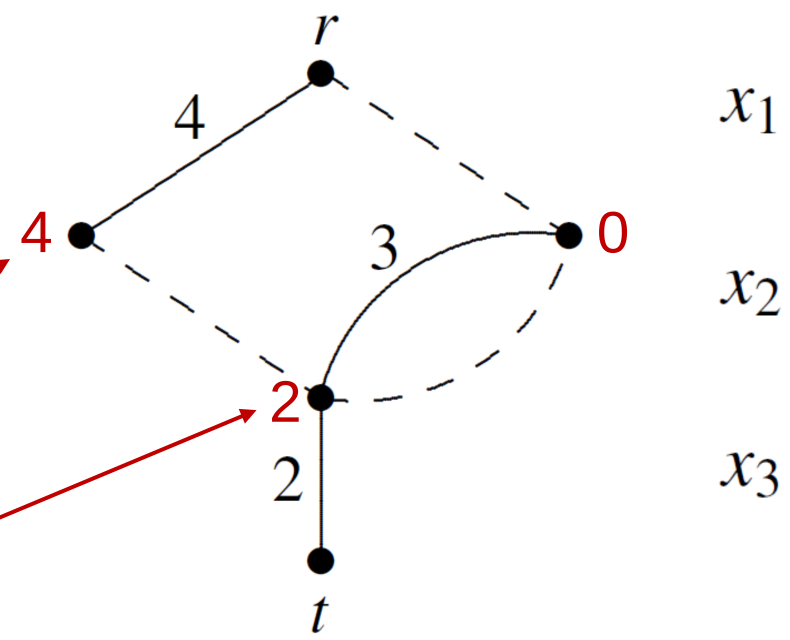
Optimal value = 2

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

Postoptimality Queries

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

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



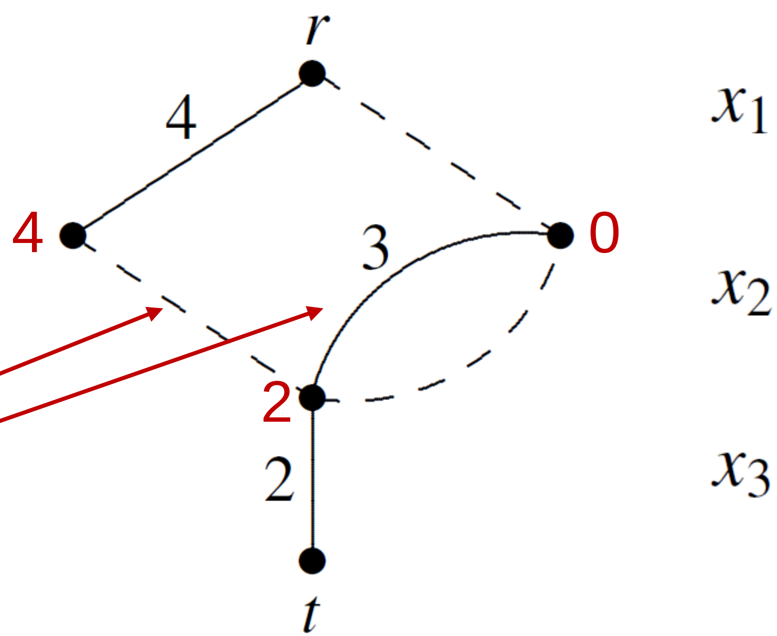
Shortest distance to root

Shortest distance to terminus

Postoptimality Queries

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

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



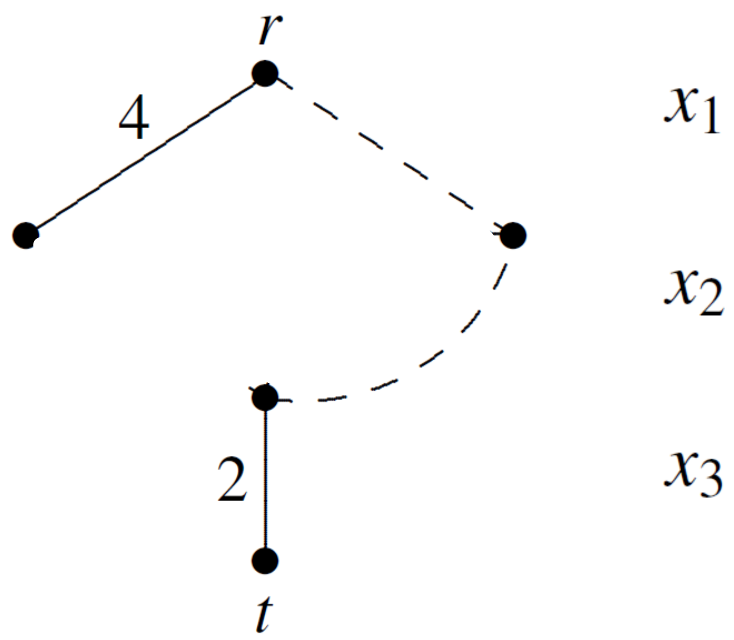
These arcs are removed because $4 + 0 + 2 > 2 + \delta$

$0 + 3 + 2 > 2 + \delta$

Postoptimality Queries

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

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

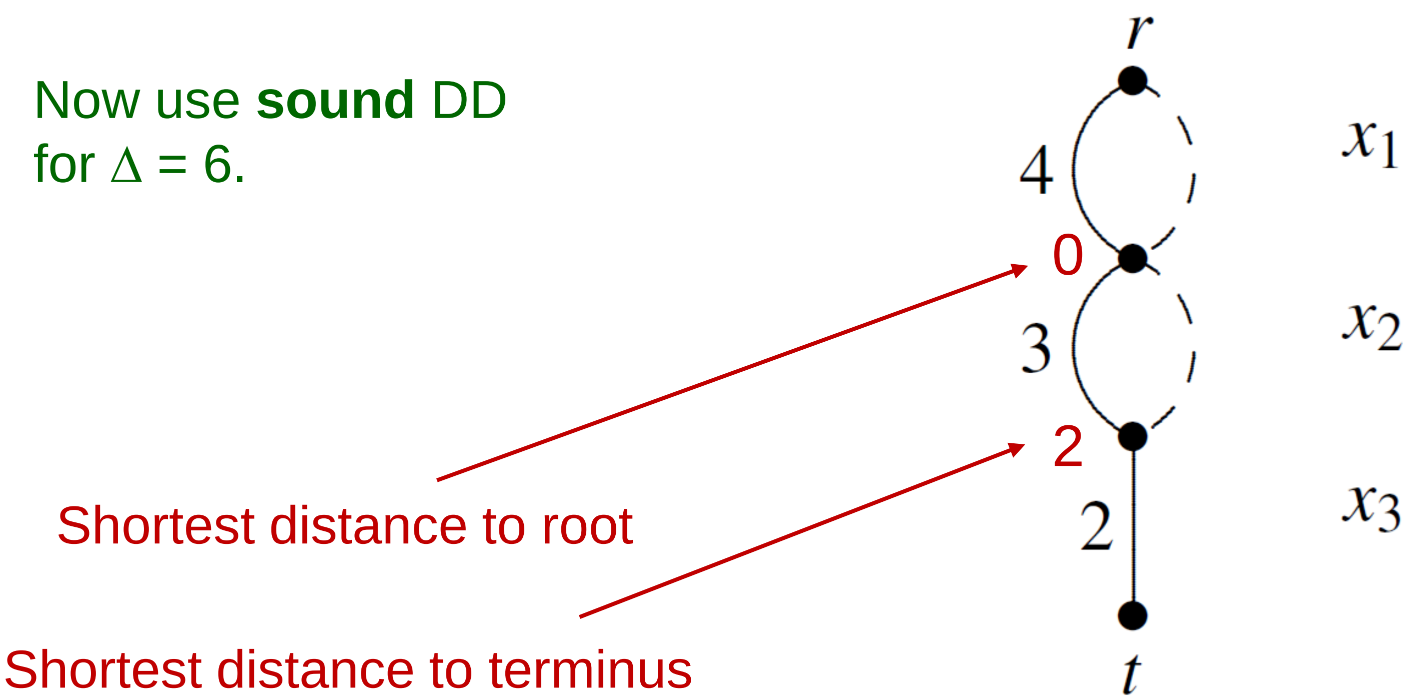


x_2 can take only value 0 when $\delta = 2$.

Postoptimality Queries

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

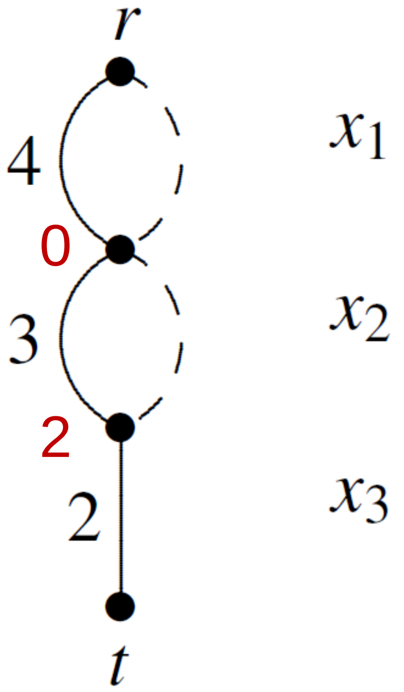
Now use **sound** DD
for $\Delta = 6$.



Postoptimality Queries

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

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



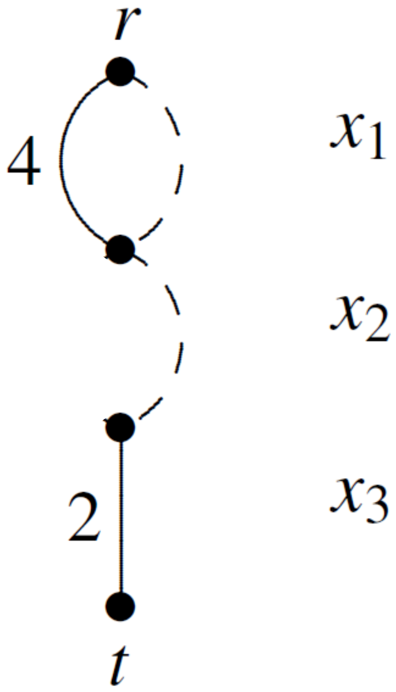
This arc is removed
because $0 + 3 + 2 > 2 + \delta$

Postoptimality Queries

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

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.

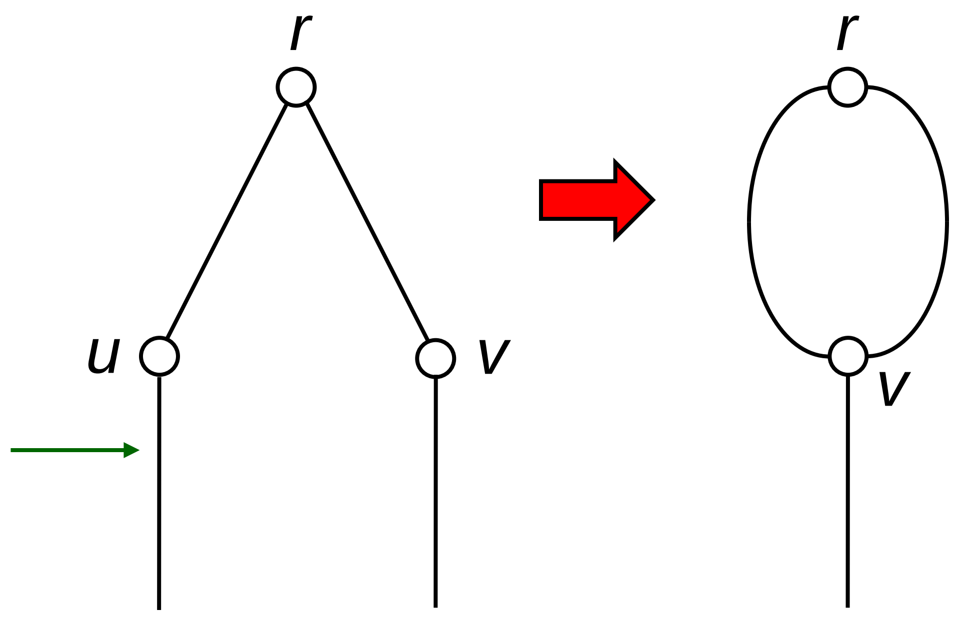
So there is no point in using sound DDs for **optimal solutions only**. (Not so for MIP.)

Serra and JH (2018)

Sound Reduction

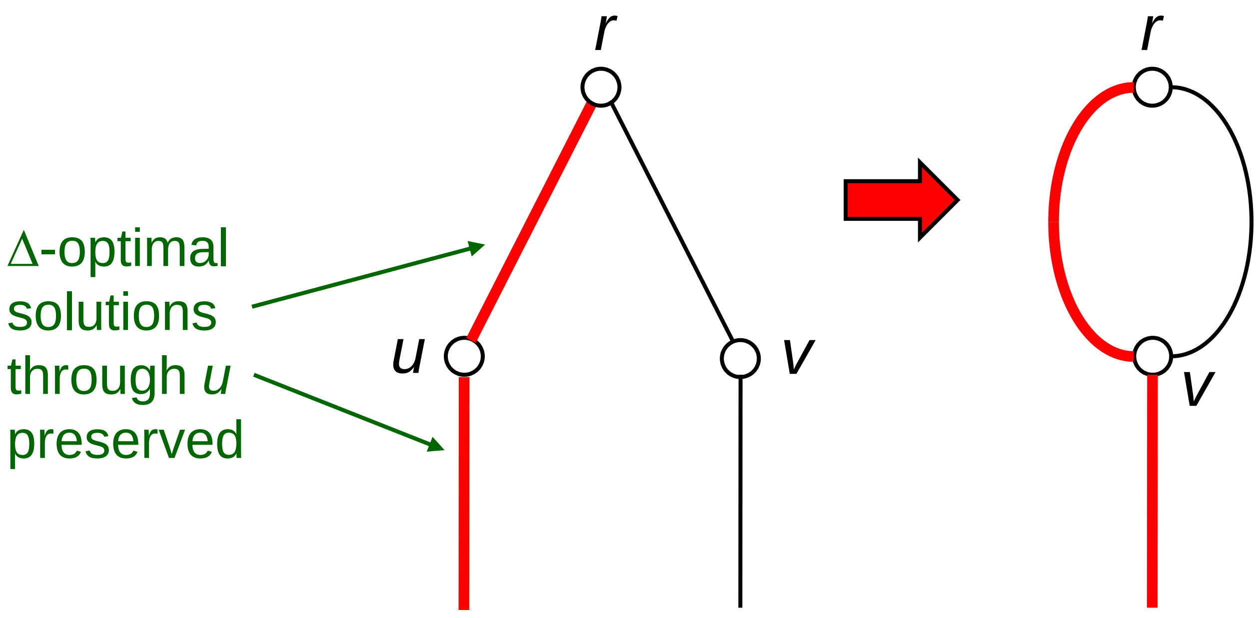
- We can **sound reduce** node ***u*** into node ***v*** when this removes no Δ-optimal solutions and introduces only spurious solutions.
  - Can be checked recursively while building diagram.

Arcs out of *u* are dropped



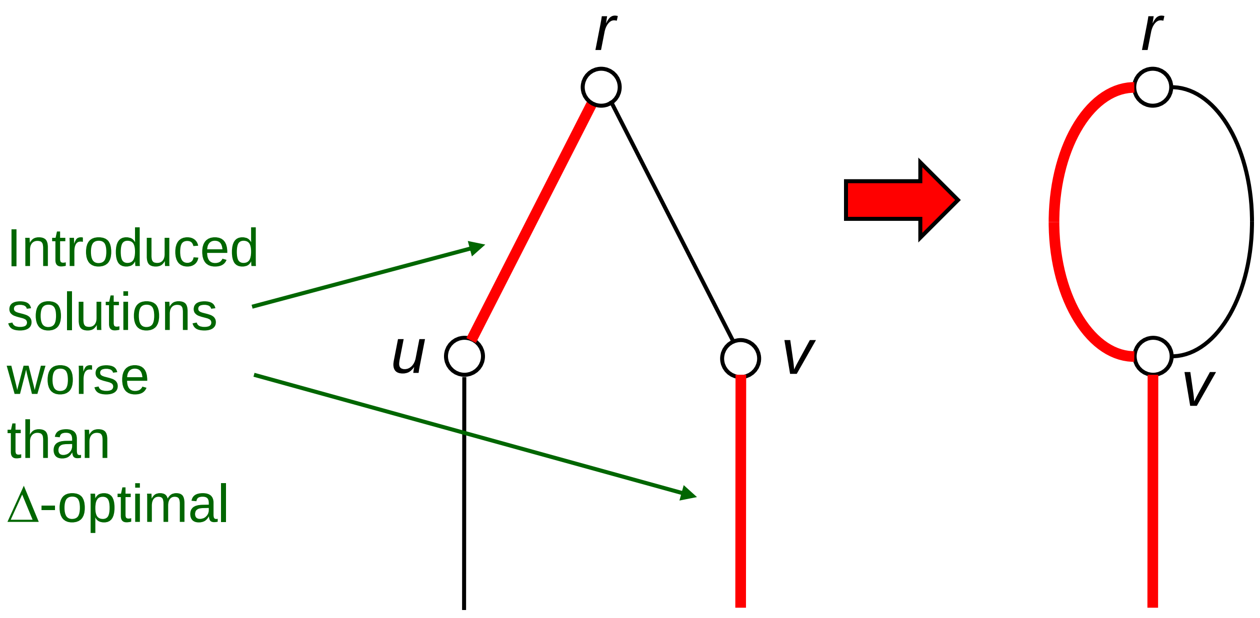
Sound Reduction

- 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.



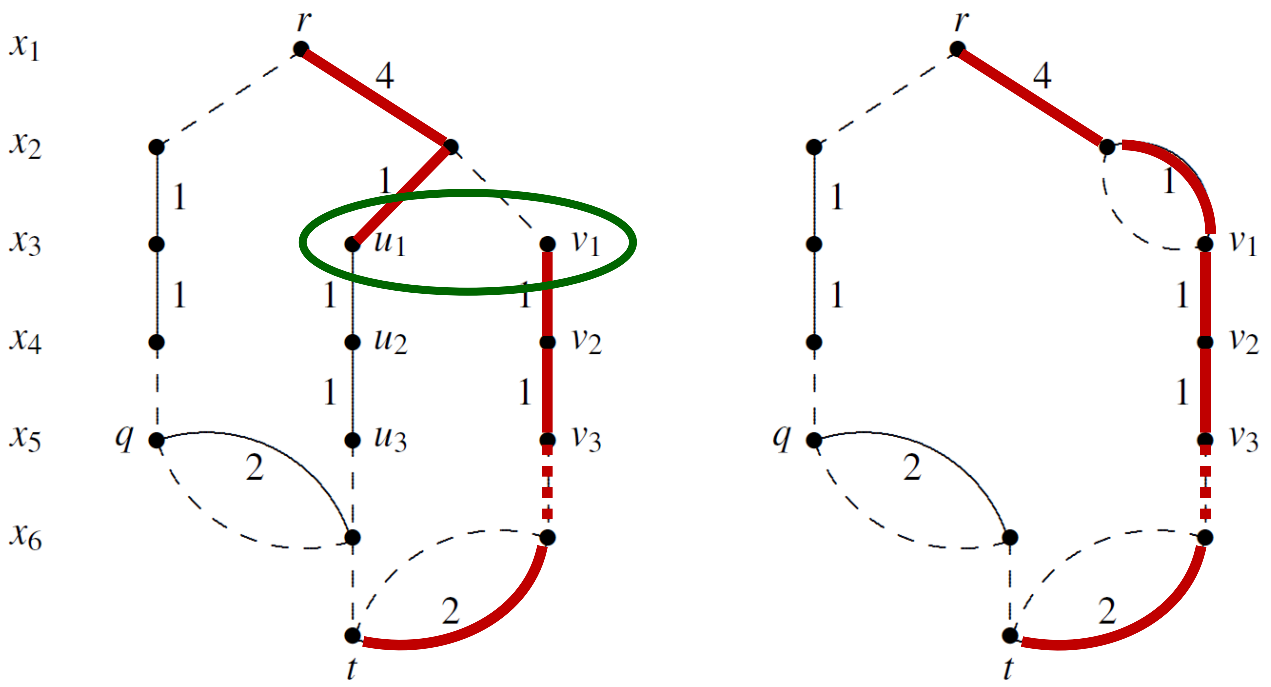
Sound Reduction

- 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**.

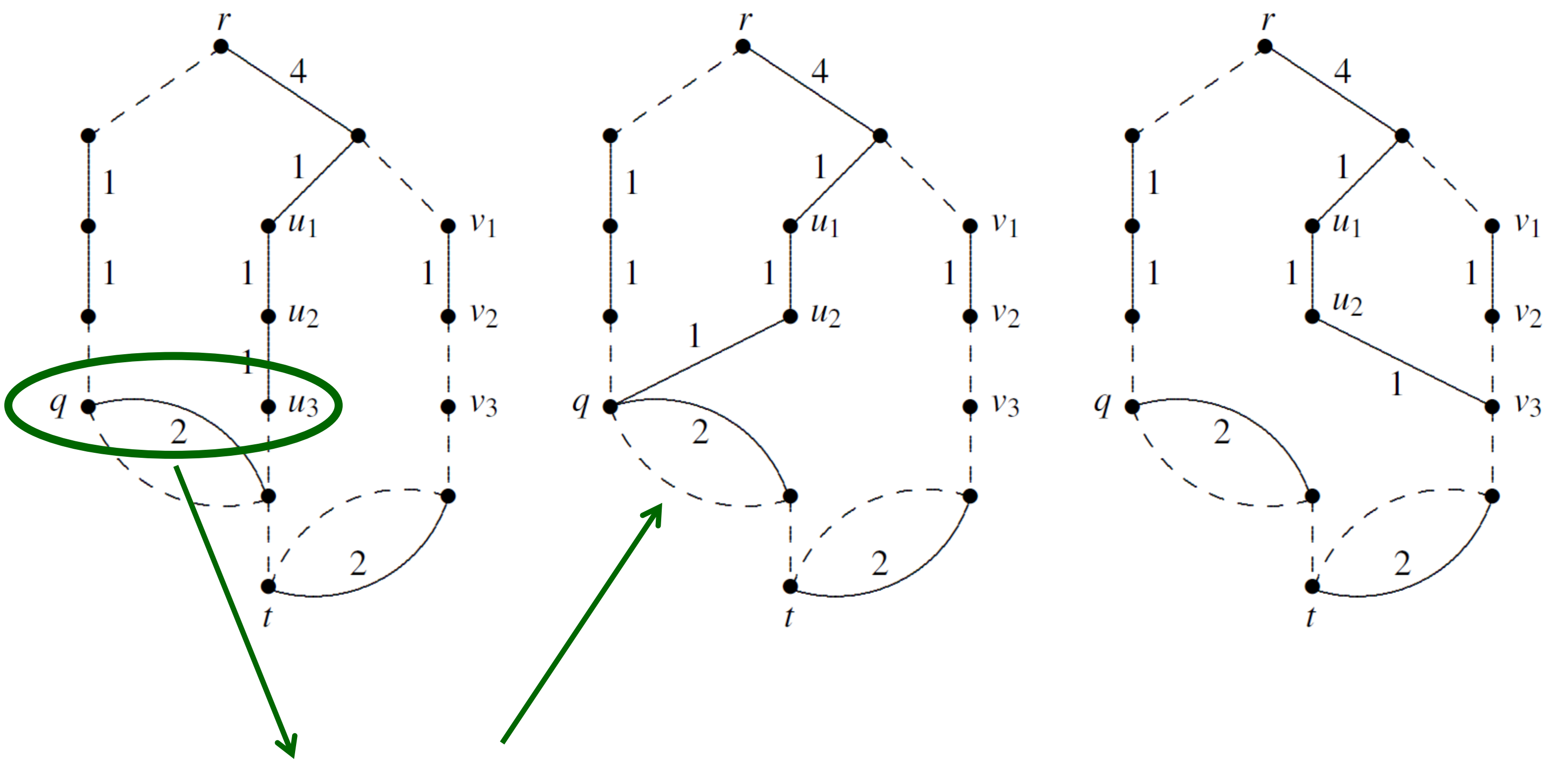
Different reduction orders can yield different diagrams, but **they all have the same size!**

(Does not hold for MILP.)

Serra and JH (2018)

Sound Reduction

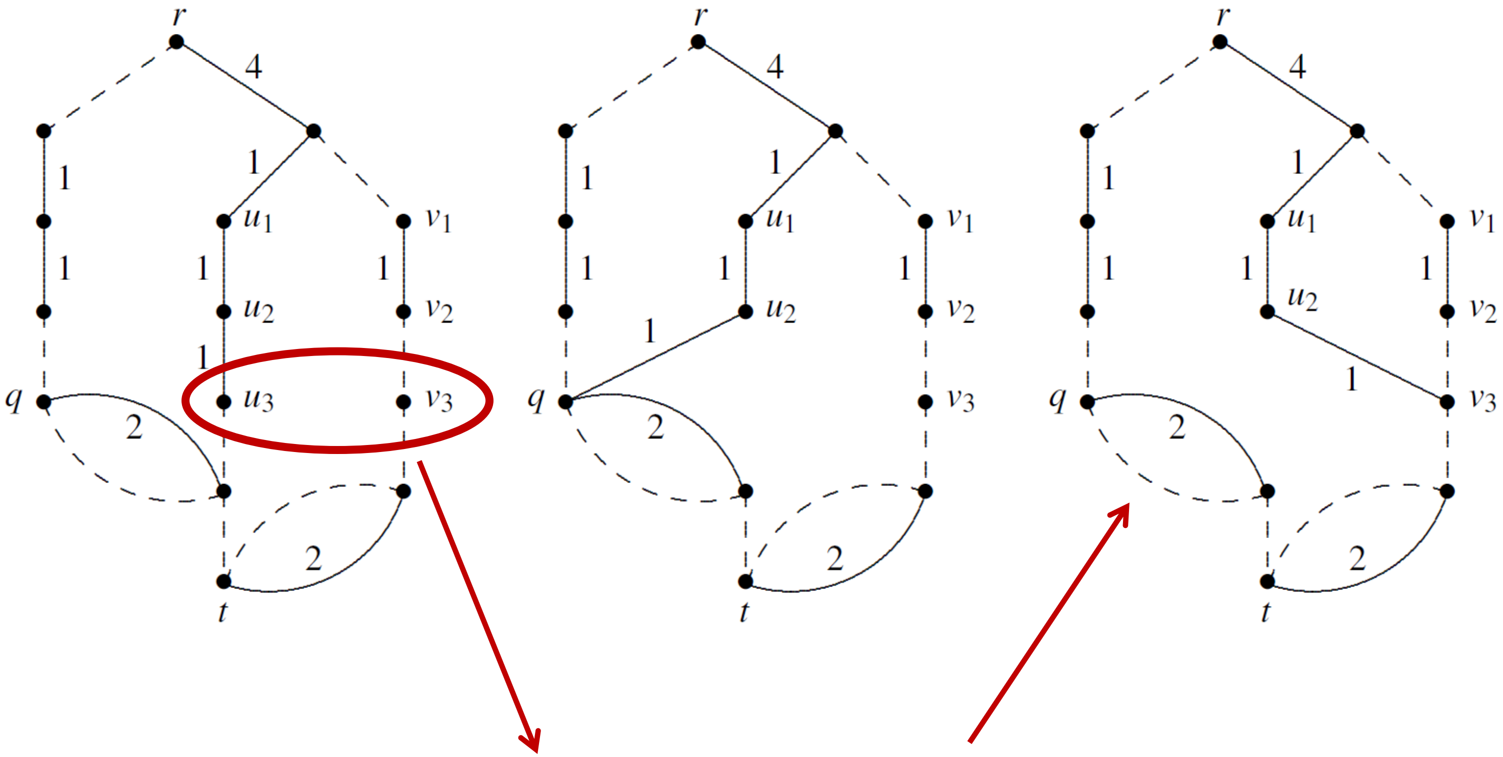
Mergers yield 2 different sound-reduced diagrams,
but of the same size



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

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

Optimal value

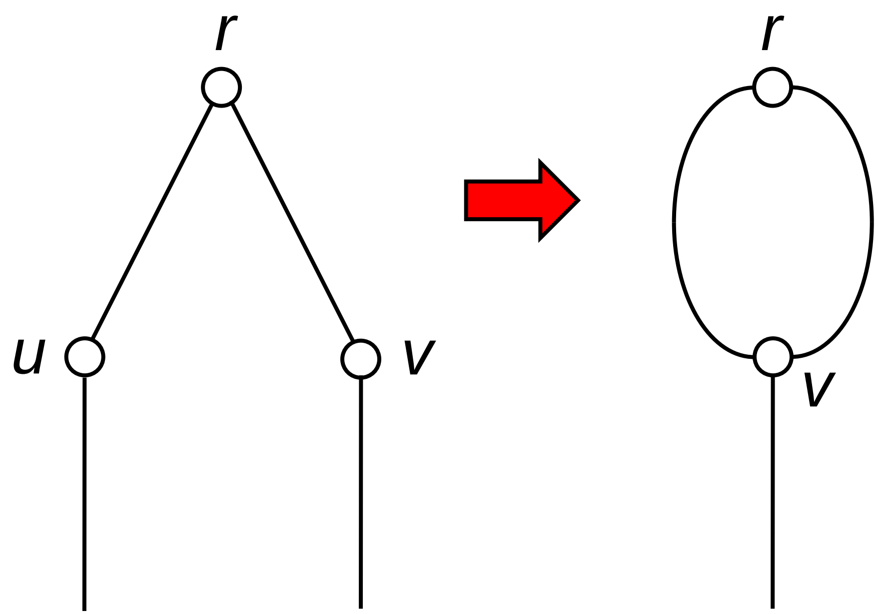
$$w(r,u) + \boxed{\mathrm{LCDS}}(v,u) > z^* + \Delta$$

Least-cost differing suffix
= min cost of 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 $\delta < \Delta$.

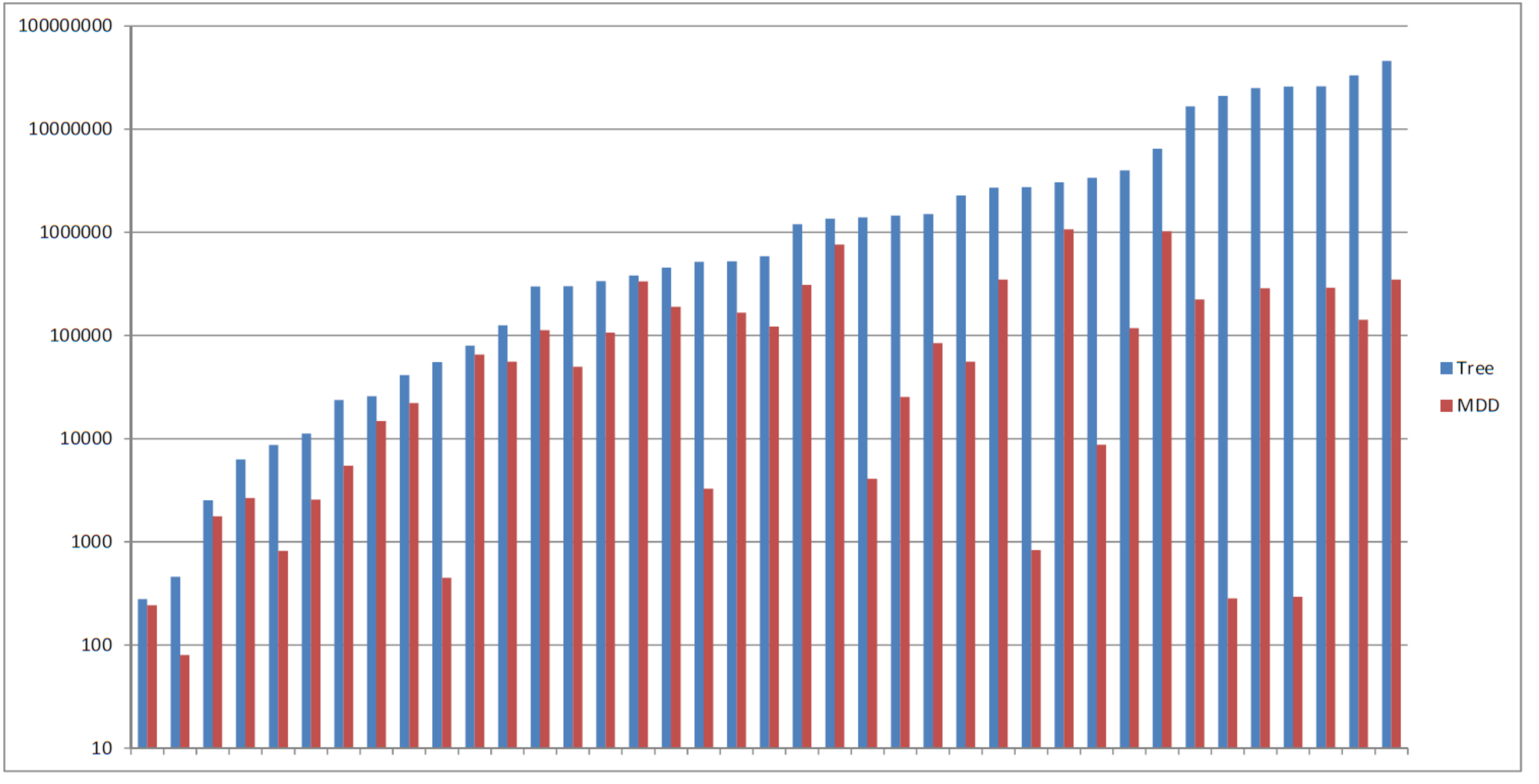
Compression

- We measure:
  - Size of **tree** representation of Δ-optimal solutions.
    - Smaller than a list.
  - Size of **reduced DD** and **sound-reduced DD**.
  - **Computation times**, including search time for Δ-optimal solutions.
    - Stand-alone method
    - CPLEX-assisted method

DD Compression

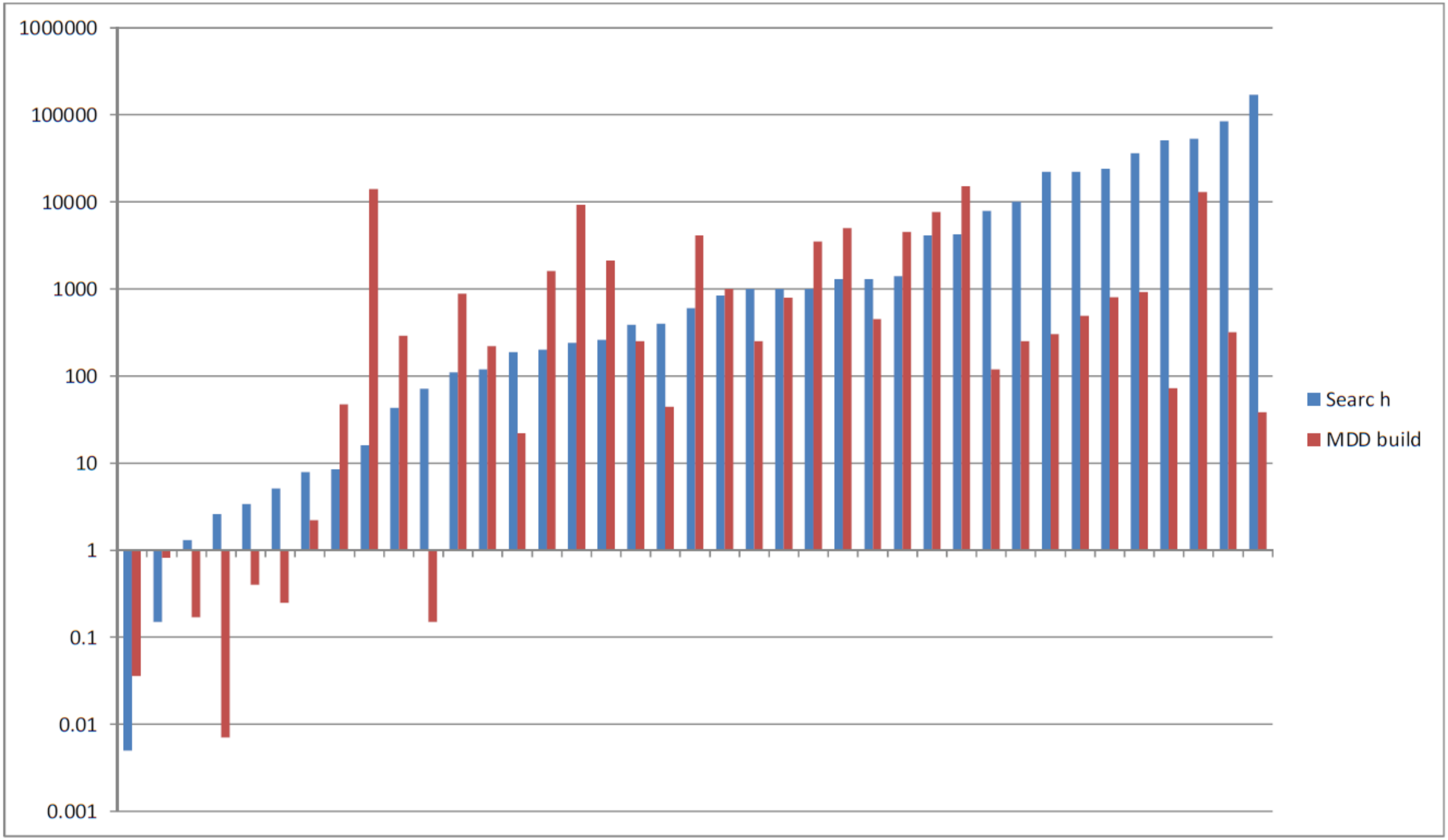
Tree size & sound-reduced DD size for large Δ

39 IP instances from MIPLIB 3.0 and MIPLIB 2010

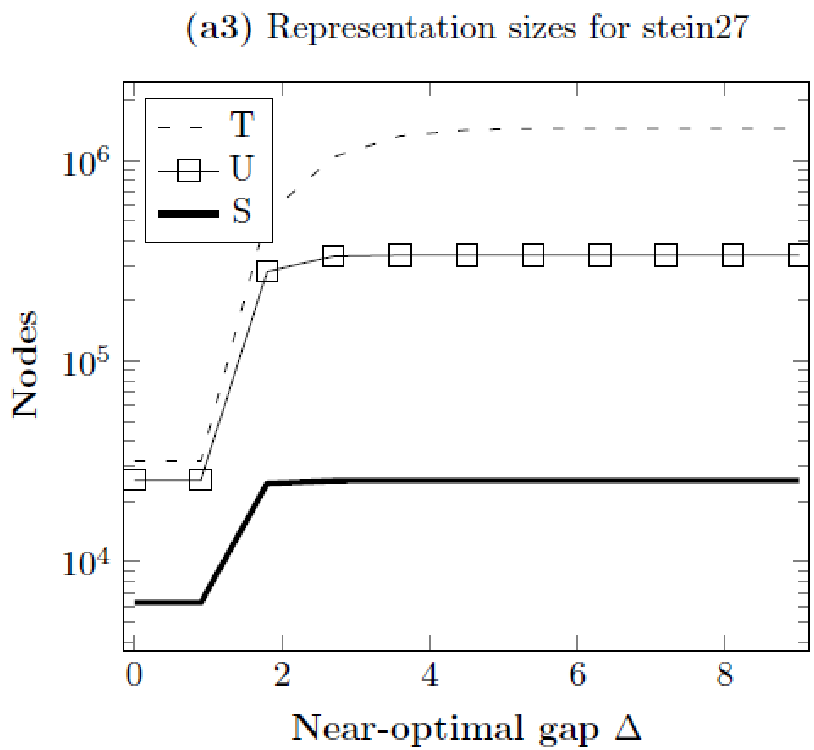


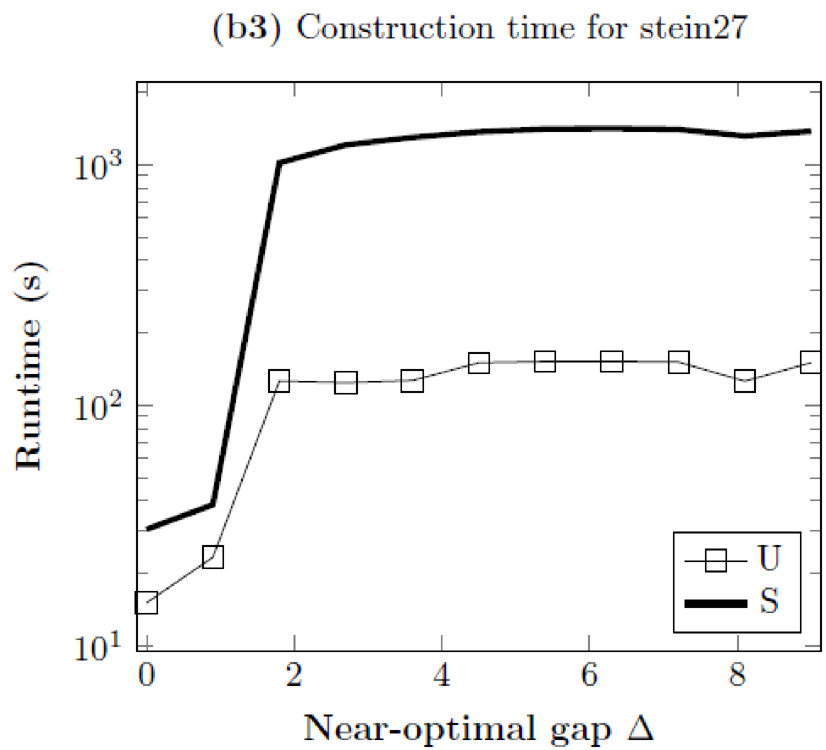
Search & Compression Time

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



DD Compression & Time vs Δ

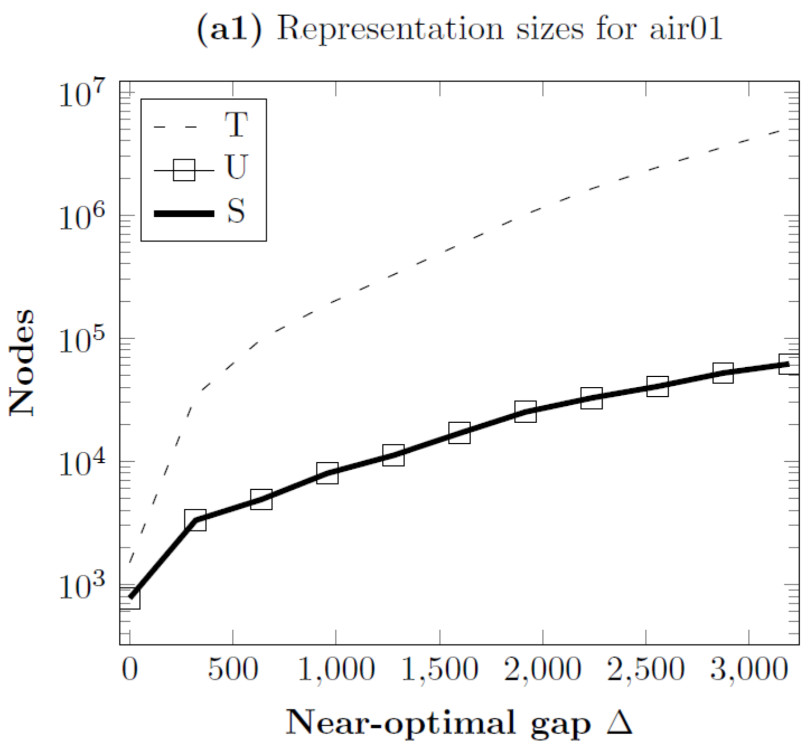


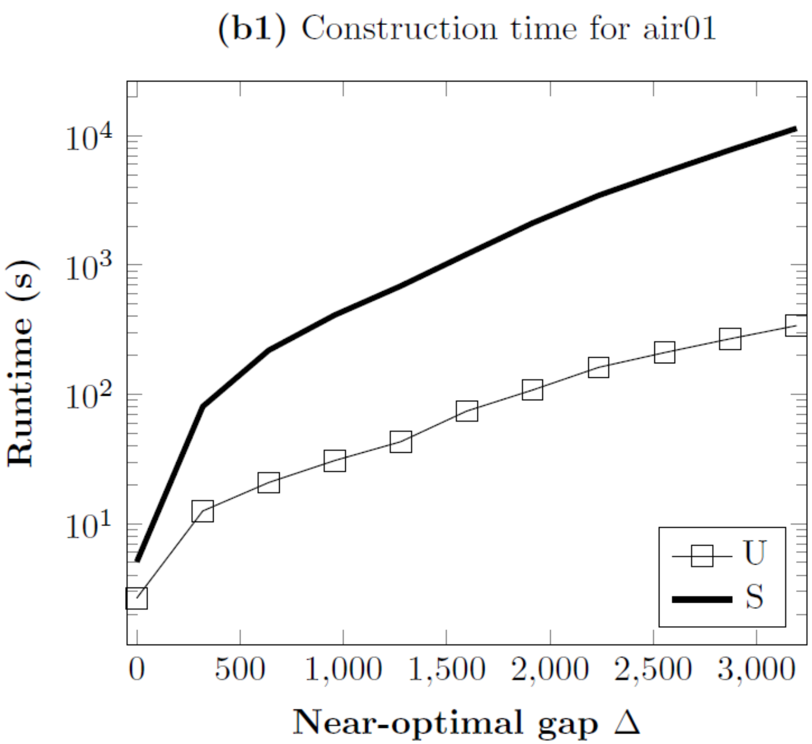


Stand-alone method

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

DD Compression & Time vs Δ



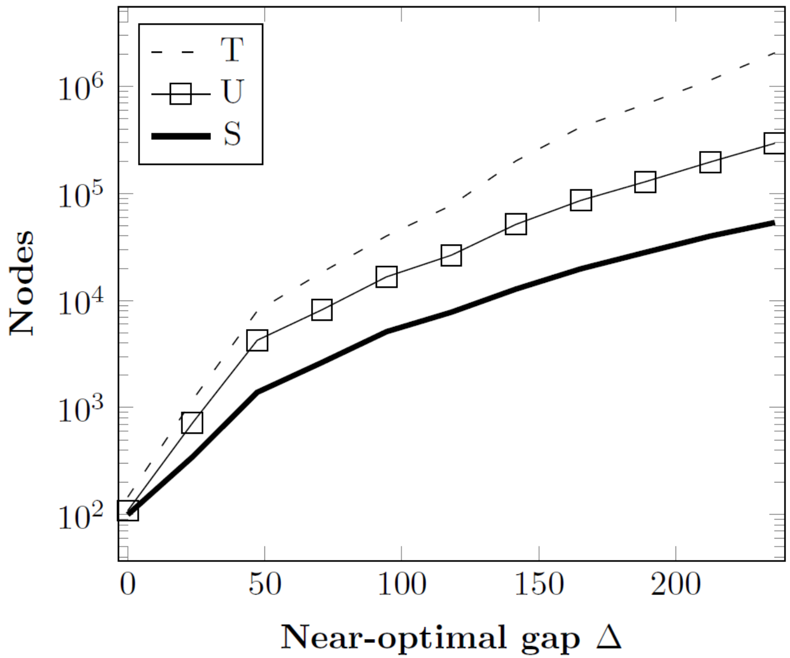


Stand-alone method

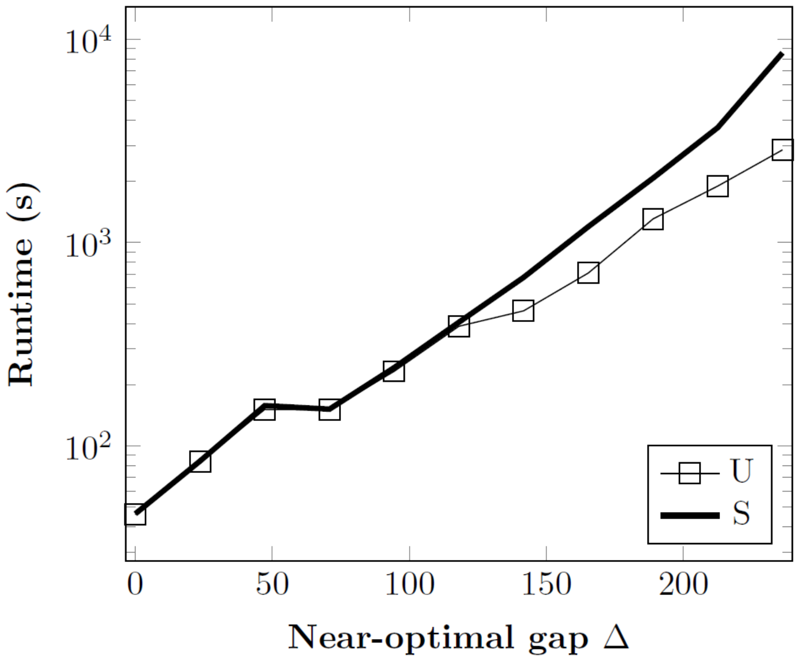
T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression & Time vs Δ

(a2) Representation sizes for lseu



(b2) Construction time for lseu

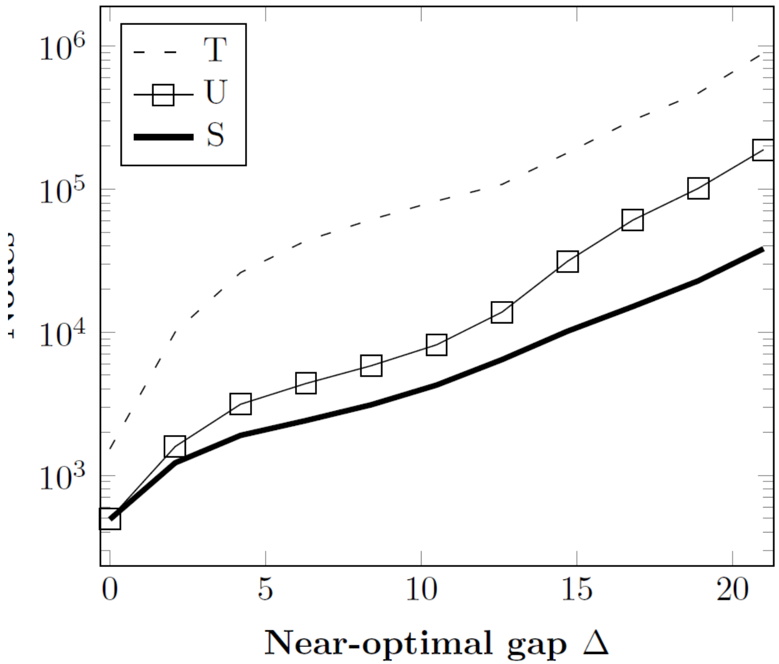


Stand-alone method

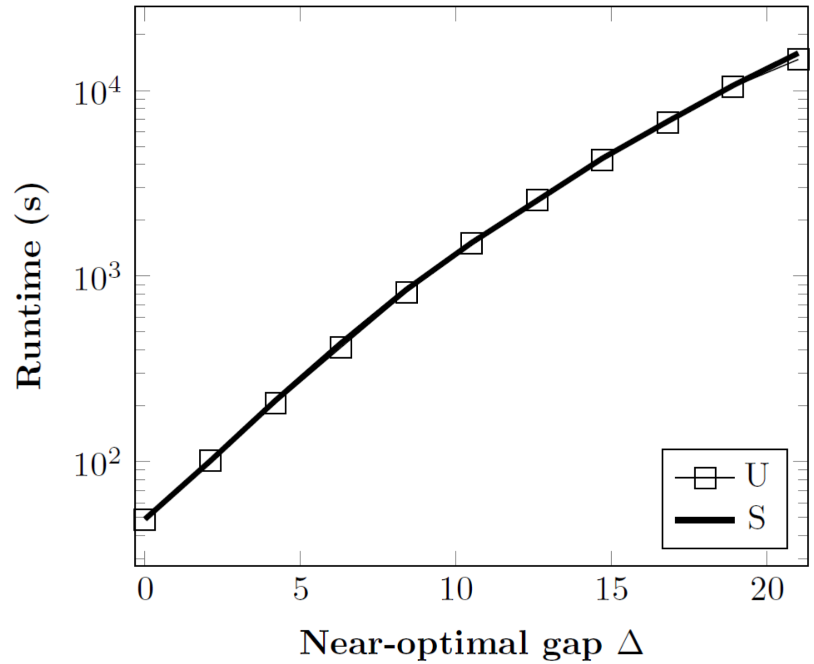
T = tree representation
U = reduced DD
S = sound-reduced DD

DD Compression & Time vs Δ

(a3) Representation sizes for mod008



(b3) Construction time for mod008

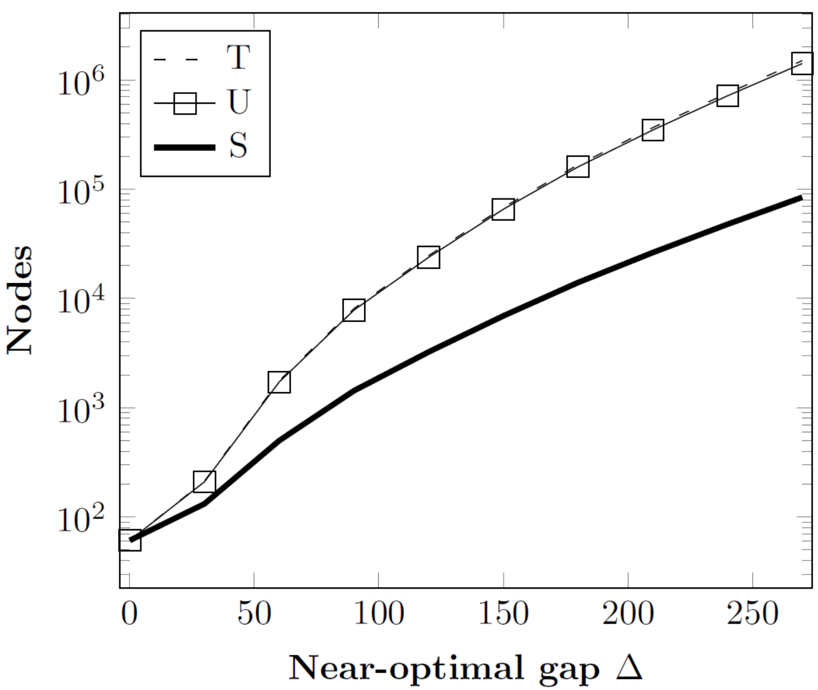


Stand-alone method

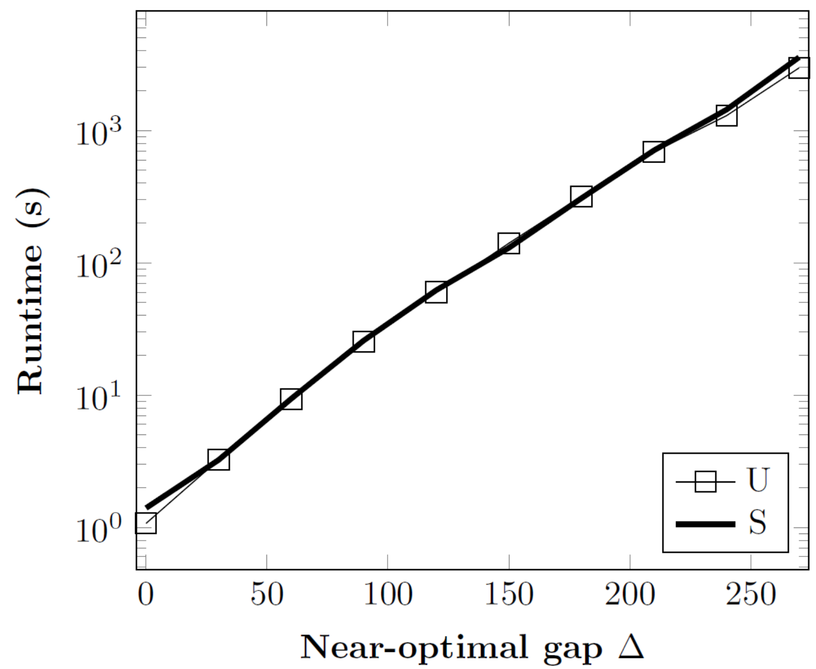
T = tree representation
U = reduced DD
S = sound-reduced DD

DD 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

Extension to MILP

- DD representation of MILP
  - DD represents **only integer solutions**.
  - Distinguish:
    - path **length** = cost of integer variables on path
    - path **cost** = value of **LP relaxation** after fixing integer variables on path

Extension to MILP

- DD representation of MILP
  - DD represents **only integer solutions.**
  - Distinguish:
    - path **length** = cost of integer variables on path
    - path **cost** = value of **LP relaxation** after fixing integer variables on path
- Two basic strategies
  - Merge nodes with **equivalent** states
    - More effective for MILP than IP
  - Shrink DD by introducing **spurious solutions.**
    - By **dualizing** constraints to obtain node equivalence.
    - By **sound reduction**, as in IP but modified.

Soundness

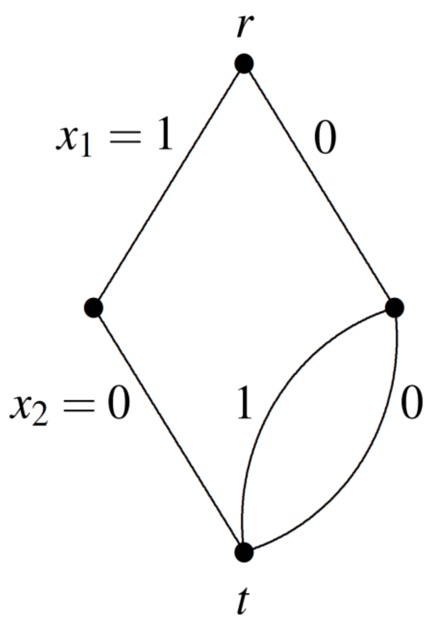
- Soundness defined as before
  - Admit spurious solutions with cost greater than $z^* + \Delta$
  - Spurious solutions can be feasible or infeasible.

Soundness

- Soundness defined as before
  - Admit spurious solutions with cost greater than $z^* + \Delta$
  - Spurious solutions can be feasible or infeasible.
- Sound reduction now useful for **optimal** as well as near-optimal solutions.
  - A minimal sound DD can contain spurious solutions even when $\Delta = 0$.

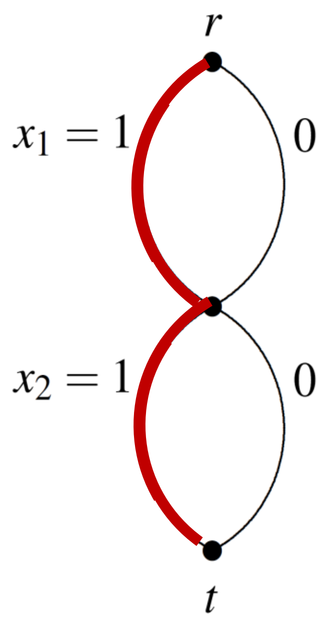
$$\min\left\{z = x_1 + x_2 + y_1 \;\middle|\; y_1 \geq 1 - x_1 - x_2,\ x_1, x_2 \in \{0,1\},\ y_1 \geq 0\right\}$$

Exact DD
for $\Delta = 0$



3 optimal solutions
with $z^* = 1$

Minimal sound DD
for $\Delta = 0$



Minimal because
every arc is part of a
Δ-optimal solution

Spurious solution
is suboptimal.

DD is smaller.

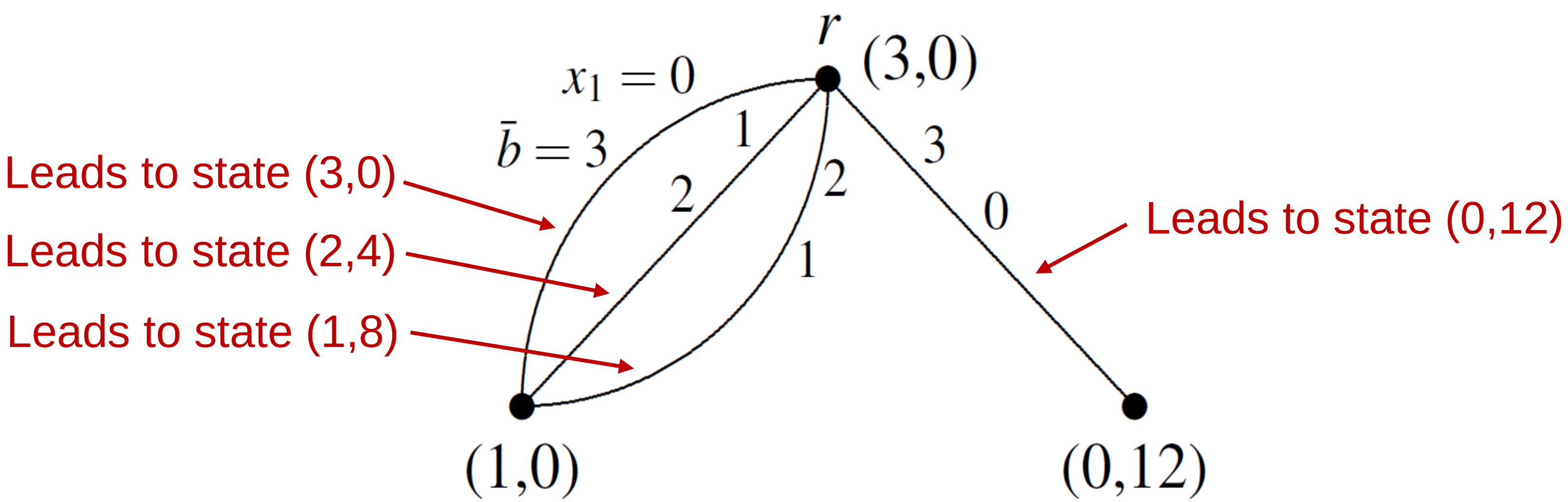
Equivalent States

- Nodes with **equivalent RHS** states $\bar{b}$ can be identified.
  - Node state is $(\bar{b}, v)$

Modified RHS after variables along path down to current node are fixed (in general, a tuple of RHSs)

Length (not cost) of shortest path from root to current node

$$\min\left\{ z = 4x_1 + 5x_2 - y_1 \;\middle|\; x_1 + 3x_2 - y_1 \geq 3,\ x_1, x_2 \in \{0,1,2,3\},\ y_1 \geq 0 \right\}$$



All RHS states $\bar{b} \in [\varepsilon, 3]$ are equivalent because they allow the same values of x_1, x_2. So arcs leading to (3,0), (2,4), (1,8) can lead to the same node with state (min{3,2,1}, min{0,4,8}) = (1,0).

We say $[\varepsilon, 3]$ is an **equivalency range for** $\bar{b}$.

Equivalent States

- MILP states are more often equivalent than IP states.
  - Presence of continuous variables often leads to equivalency range $[-\infty, \infty]$.
  - So many constraints have same equivalency range.

$$ax + d_1 y_1 - d_2 y_2 \geq \beta, \quad d_1, d_2 > 0$$

$$a'x - d_1' y_3 + d_2' y_4 \geq \beta', \quad d_1', d_2' > 0$$

Both constraints have equivalency range $[-\infty, \infty]$

Deleting Arcs

- An arc can be deleted when it cannot be part of a Δ-optimal solution.
  - Based on LP bound $L_j(\bar{b})$ of cost between node at the end of the arc and the terminus.

If the MILP is

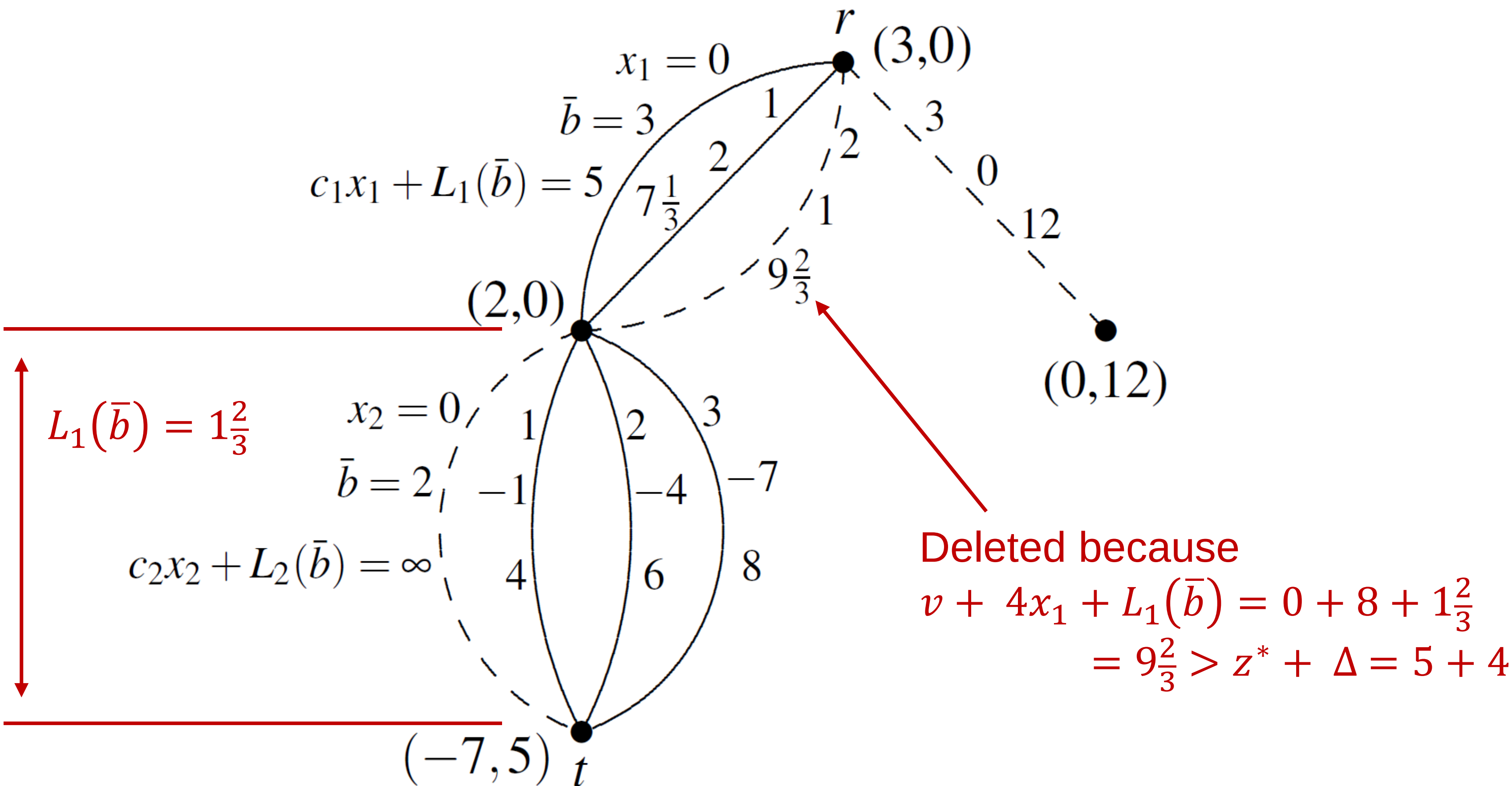
$$\min\left\{cx + dy \;\middle|\; Ax + By \geq b,\ x_j \in \{L_j, \ldots, U_j\},\ \text{all } j\right\}$$

Then

$$L_j(\bar{b}) = \min\left\{ \sum_{i=j+1}^{n} c_i x_i + dy \;\middle|\; \sum_{i=j+1}^{n} A_i x_i + By \geq \bar{b},\right.$$
$$\left. L_i \leq x_i \leq U_j,\ i = j+1, \ldots, n \right\}$$

$$\min\left\{z = 4x_1 + 5x_2 - y_1 \;\middle|\; x_1 + 3x_2 - y_1 \geq 3,\ x_1, x_2 \in \{0,1,2,3\},\ y_1 \geq 0\right\}$$

Dashed arcs can be deleted when Δ = 4.



Top-Down Compilation

Build a DD by top-down branching, identifying equivalent states, and deleting arcs when possible.

Theorem. This procedure results in a sound DD for a given Δ.

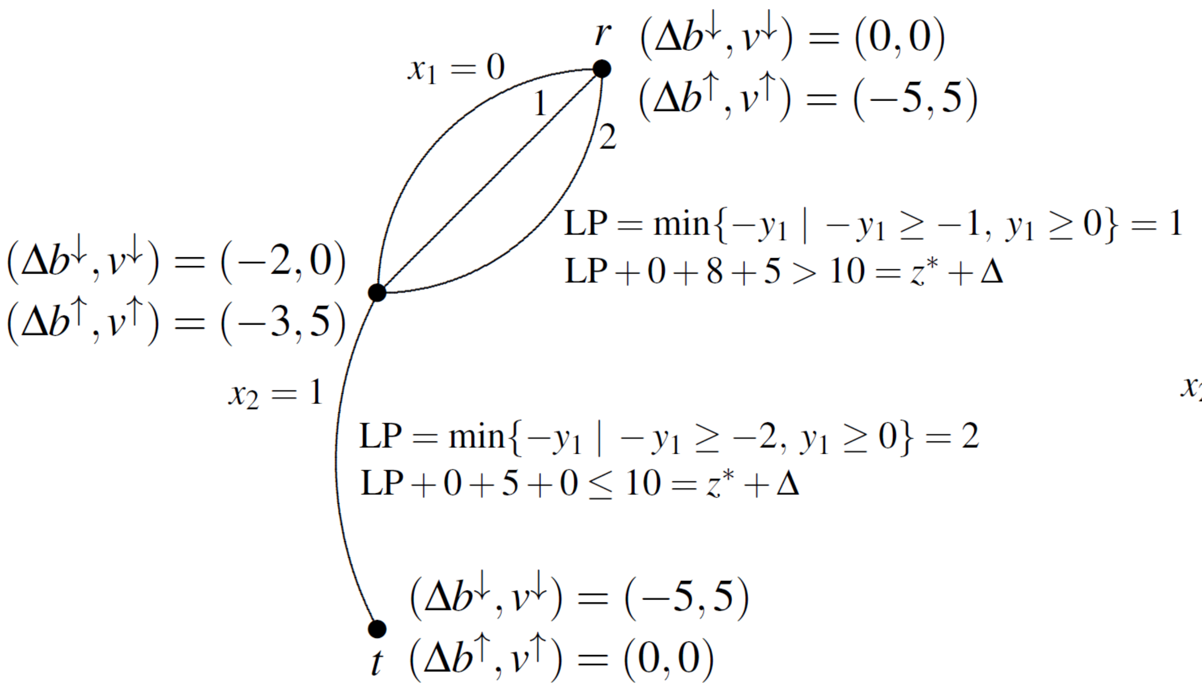
When identifying nodes, use the state with the smallest LP bound on cost, rather than taking mins.

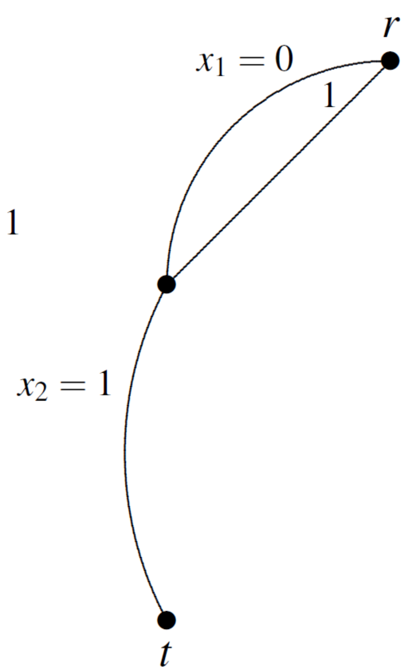
Theorem. This can result in a smaller sound DD.

Theorem. A bottom-up pass can yield a still smaller DD.

$$\min\left\{z = 4x_1 + 5x_2 - y_1 \;\middle|\; x_1 + 3x_2 - y_1 \geq 3,\ x_1, x_2 \in \{0,1,2,3\},\ y_1 \geq 0\right\}$$

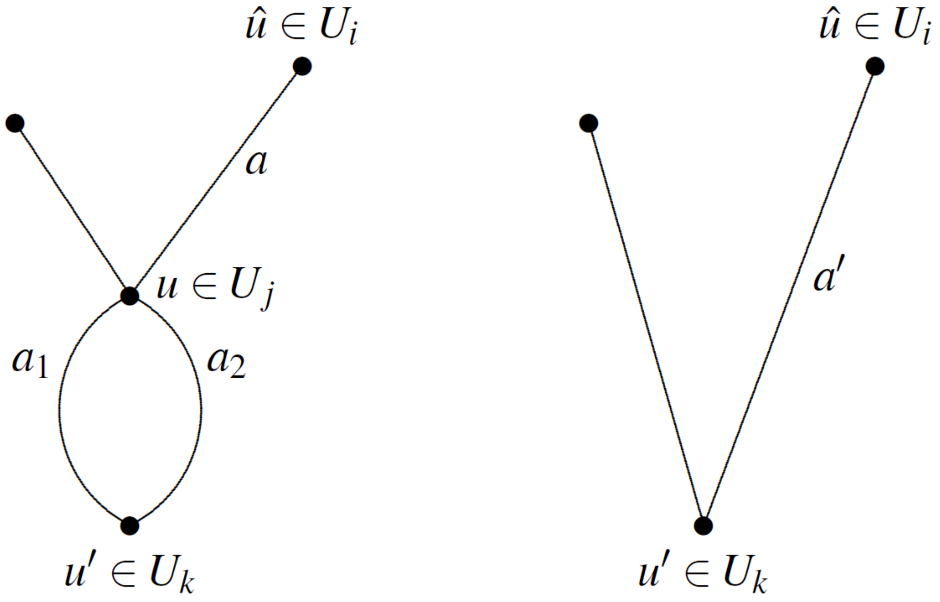
Bottom-up pass deletes one more arc.





Theorem. Arc contraction can delete more arcs while preserving soundness.

Contraction of arcs a_1, a_2



Separable Constraints

- Problem: Because $\bar{b}$ is a tuple, it is hard to prove equivalence.
  - Let a subset *S* of constraints be **separable** when the problem of finding equivalency ranges for the entire constraint set can be decomposed into finding ranges for *S* and its complement separately.
- Dividing constraints into separable subsets can help prove equivalence.
  - Particularly because constraints with continuous variables often have RHS equivalency range $[-\infty, \infty]$.
    - Their RHS states are always equivalent.

Separable Constraints

Theorem. If *S* has no continuous variables in common with other constraints, then *S* is separable.

Corollary. A pure integer constraint is separable and can therefore be analyzed separately.

Corollary. If all constraints in *S* have equivalency range $[-\infty, \infty]$, we can ignore *S* when computing equivalency ranges for the entire constraint set.

Dualizing Constraints

- Constraints that block equivalence can be **dualized**.
  - Given constraint $A_i x + B_i y \geq b_i$ add artificial variable to obtain $A_i x + B_i y + s_i \geq b_i$
  - Add $s_i \geq 0$ to the constraint set and $+M s_i$ to the objective function.
  - Constraint $A_i x + B_i y \geq b_i$ can now be ignored when checking for equivalence.

Theorem. For sufficiently large but bounded *M,* dualizing constraints preserves soundness.

Yet it results in more spurious solutions.

Sound Reduction

- Sound reduction can be defined in parallel with IP.
  - However, the test for sound reduction is harder to pass.
  - It relies on LP bounds rather than path lengths.
  - Finding weaker conditions for sound reduction is a current research issue.

Research Issues

- Applications to:
  - Multiobjective optimization
    - Original application!
  - General mixed discrete/continuous programming
    - Not just MILP

Research Issues

- Applications to:
  - Multiobjective optimization
    - Original application!
  - General mixed discrete/continuous programming
    - Not just MILP
- How to combine DD-based solution with DD-based postoptimality?
  - Partially analogous to "**1-tree**" method for generating near-optimal MILP solutions.
    - Search for all near-optimal solutions using the **same tree**.
  - Use a "**1-DD**" method.
    - Search for all near-optimal solutions in the **same DD**.
    - Result is a **sound DD** representing the solutions, rather than just a **list** as in MILP.