

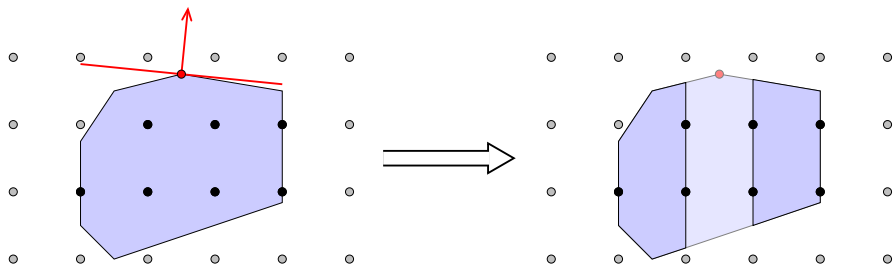
Improving Strong Branching by Domain Propagation

Gerald Gamrath

Zuse Institute Berlin · Berlin Mathematical School



Branching on Variables for MIPs

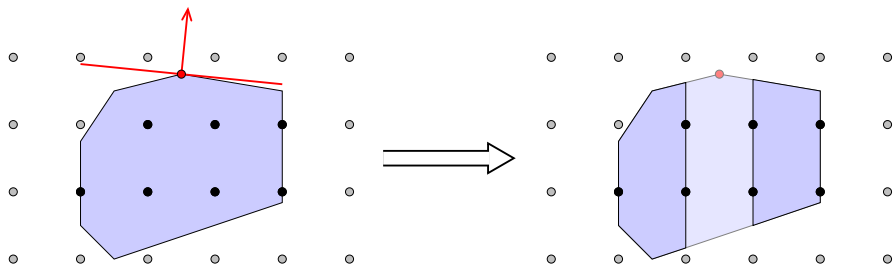


Goals

- ▷ increase child nodes' dual bounds
- ▷ prove optimality fast

Tools

- ▷ LP solution value
- ▷ history information
- ▷ strong branching



Goals

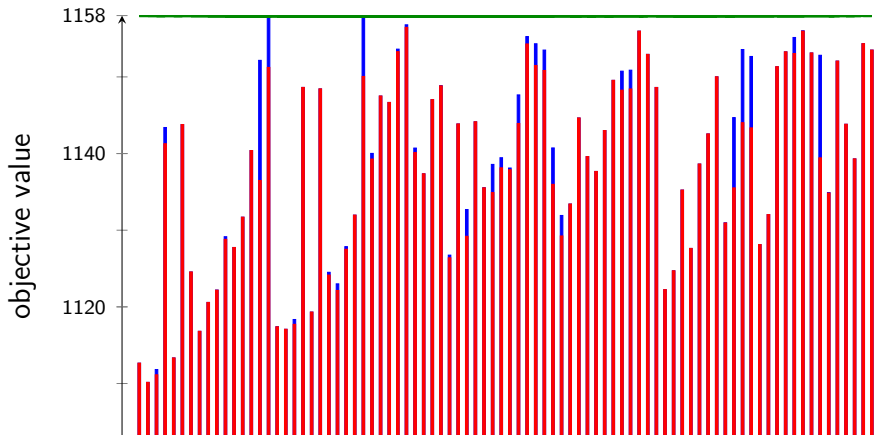
- ▷ increase child nodes' dual bounds
- ▷ prove optimality fast

Tools

- ▷ LP solution value
- ▷ history information
- ▷ **strong branching**

Quality of Strong Branching Results

strong branching dual bounds vs. LP value during node processing

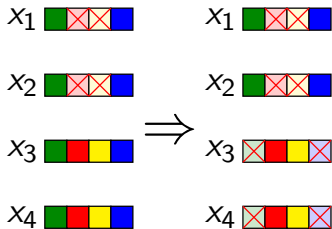


instance: aflow30a (MIPLIB 2003), full strong branching,
optimum value provided, solved after 87 nodes

Domain Propagation (Node Preprocessing)

Idea:

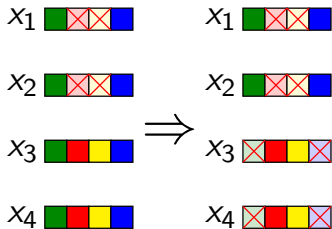
- ▷ analyze local constraints and domains
- ▷ tighten local variable domains
- ▷ first step in node processing
- ▷ better dual bounds



Domain Propagation (Node Preprocessing)

Idea:

- ▷ analyze local constraints and domains
- ▷ tighten local variable domains
- ▷ first step in node processing
- ▷ better dual bounds



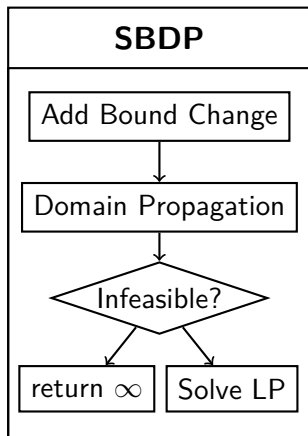
Idea of this talk

Strong branching should consider the effects of domain propagation!

Strong Branching with Domain Propagation

Strong Branching with Domain Propagation (SBDP):

- ▷ add branching bound
- ▷ perform “default” domain propagation
- ▷ solve LP



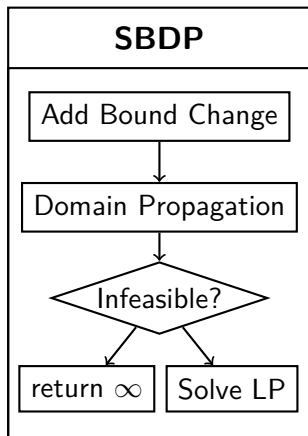
Strong Branching with Domain Propagation

Strong Branching with Domain Propagation (SBDP):

- ▷ add branching bound
- ▷ perform “default” domain propagation
- ▷ solve LP

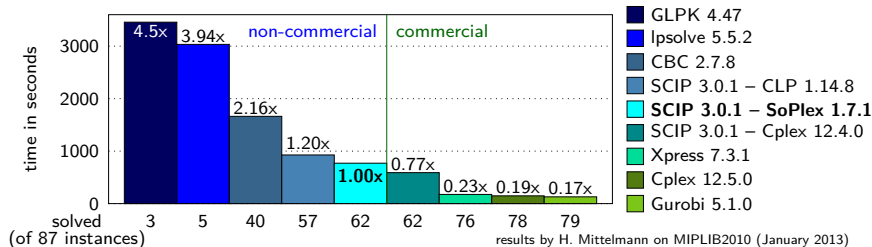
Open Questions:

- ▷ better predictions?
- ▷ change in strong branching effort?
- ▷ overall impact?



SCIP: Solving Constraint Integer Programs

- ▷ standalone solver / branch-cut-and-price-framework
- ▷ modular structure via plugins
- ▷ free for academic use: <http://scip.zib.de>
- ▷ available in source code
- ▷ very fast non-commercial MIP solver



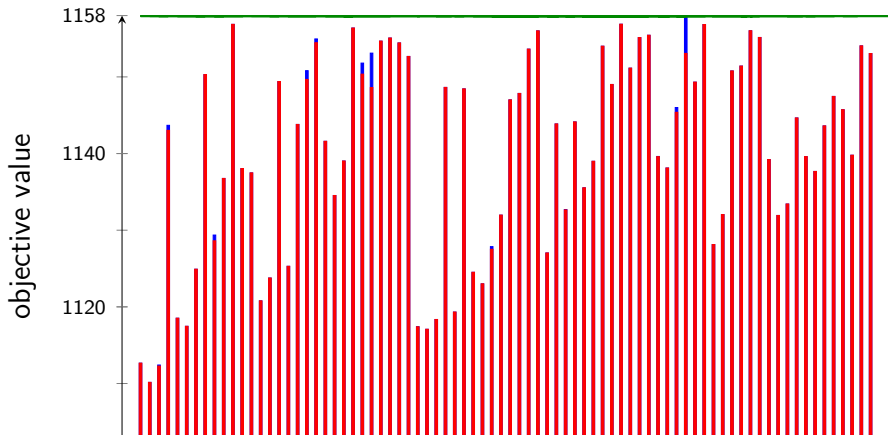
- ▷ 168 instances (MIPLIB 3, 2003, 2010 benchmark)
- ▷ excluded infeasible instances + instances without branching
- ▷ 147 instances left
- ▷ time limit: 2 hours
- ▷ full strong branching
- ▷ optimal solution value provided
- ▷ heuristics, separation turned off
- ▷ SBDP in addition to “standard” strong branching
- ▷ only “standard” strong branching results used for branching

Compare single strong branching (SB) calls

- ▷ SB children detected **infeasible**: 14% → 17%
 - ▶ 15% during propagation
- ▷ SB children with **better dual bound**: 5.6%
 - ▶ gap closed by 20% on average
- ▷ **LP iterations**:
 - ▶ infeasible children: - 18%
 - ▶ better dual bounds: + 30%
 - ▶ all other SB children: - 4%
- ▷ **Domain propagation time**: about 5%
- ▷ **SB time** increased by about 1%

Update: Quality of Strong Branching Results

SBDP dual bounds vs. LP value during node processing



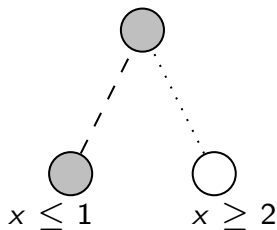
instance: aflow30a (MIPLIB 2003), full strong branching,
optimum value provided, solved after 81 nodes

Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts

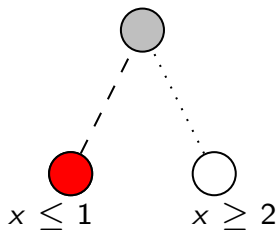
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers



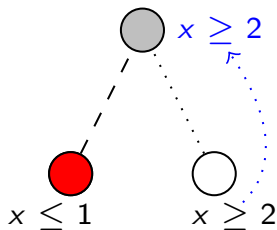
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers



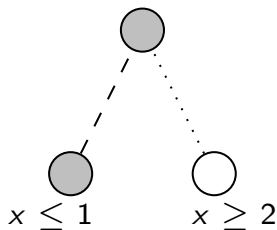
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers



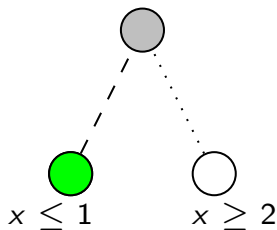
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers
- ▷ check SB solutions for integrality
 - ▶ simple rounding heuristic



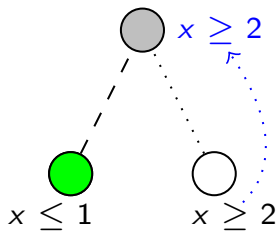
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers
- ▷ check SB solutions for integrality
 - ▶ simple rounding heuristic



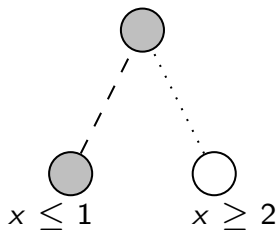
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers
- ▷ check SB solutions for integrality
 - ▶ simple rounding heuristic



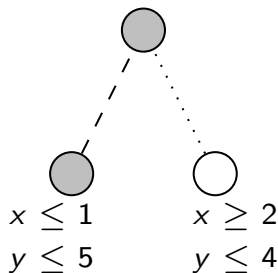
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers
- ▷ check SB solutions for integrality
 - ▶ simple rounding heuristic
- ▷ probing-like bound tightening



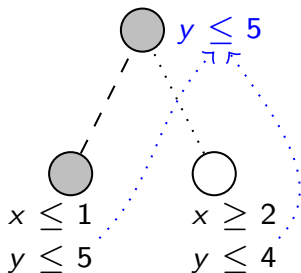
Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers
- ▷ check SB solutions for integrality
 - ▶ simple rounding heuristic
- ▷ probing-like bound tightening



Let's add some more things:

- ▷ additional statistics:
 - ▶ implications, cutoffs, conflicts
- ▷ stop when infeasible
 - ▶ up-child first
 - ▶ use locking numbers
- ▷ check SB solutions for integrality
 - ▶ simple rounding heuristic
- ▷ probing-like bound tightening



Computational Experiment: Proving Optimality

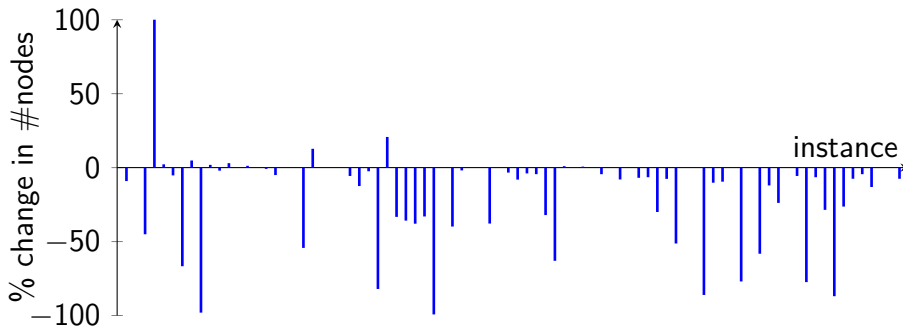
- ▷ all 168 instances (MIPLIB 3, 2003, 2010 benchmark)
- ▷ time limit: 2 hours
- ▷ original + 3 random permutations to reduce performance variability
- ▷ arithmetic mean of solving time and node number per instance
- ▷ instances solved iff all four permutations were solved

- ▷ full strong branching
- ▷ optimal solution value provided
- ▷ heuristics, separation turned off

- ▷ two individual runs
- ▷ compare B&B tree and solving time

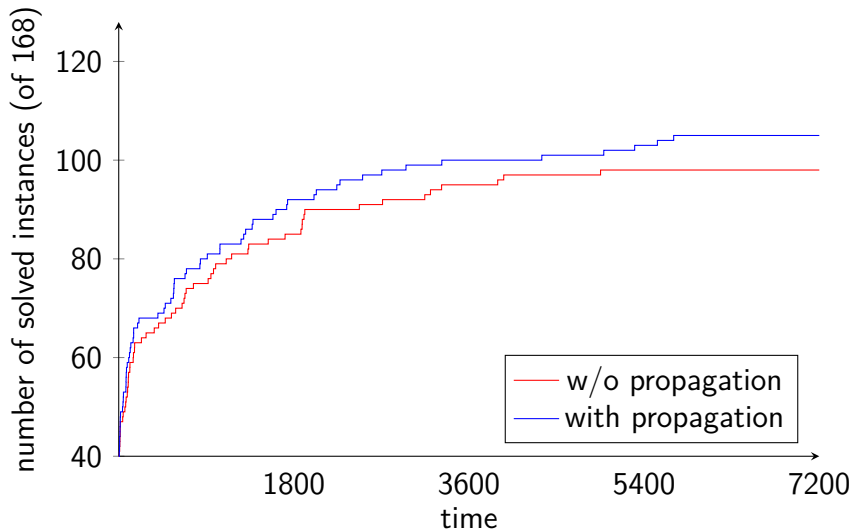
Number of Nodes: Proving Optimality

- ▷ relative change of number of nodes when using SBDP
- ▷ only instances solved to optimality by both variants



- ▷ shifted geometric mean reduced by 25%

Performance Diagram: Proving Optimality



▷ result: 7 more instances solved, shifted geom. time reduced by 13%

- ▷ extension of reliability pseudo-cost branching
- ▷ + other history information
 - ▶ inferences, conflicts, cutoffs
- ▷ strong branching LP iteration limit
- ▷ “lookahead” strategy

What does this mean for SDBP?

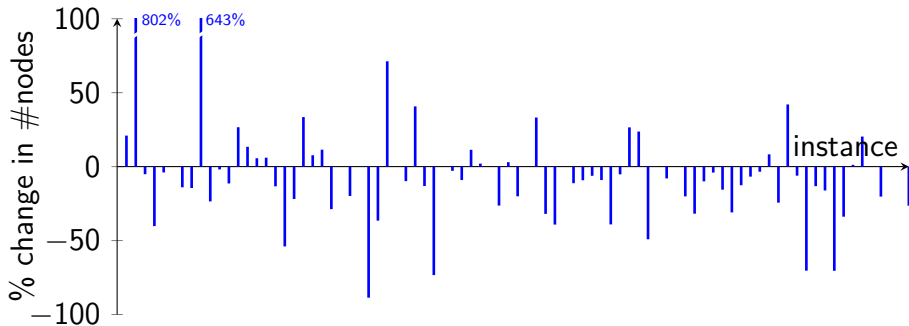
- ▷ smaller strong branching impact
- ▷ but: first branchings most important
- ▷ initializes other history information as well
- ▷ SBDP reaches LP iteration limit twice as often: 10% instead of 5%
- ▷ less “stable”

Computational Experiment: Hybrid Branching

- ▷ all 168 instances (MIPLIB 3, 2003, 2010 benchmark)
- ▷ time limit: 2 hours
- ▷ original + 3 random permutations to reduce performance variability
- ▷ arithmetic mean of solving time and node number per instance
- ▷ instances solved iff all four permutations were solved
- ▷ hybrid branching
- ▷ SCIP default settings
- ▷ two individual runs
- ▷ compare B&B tree and solving time

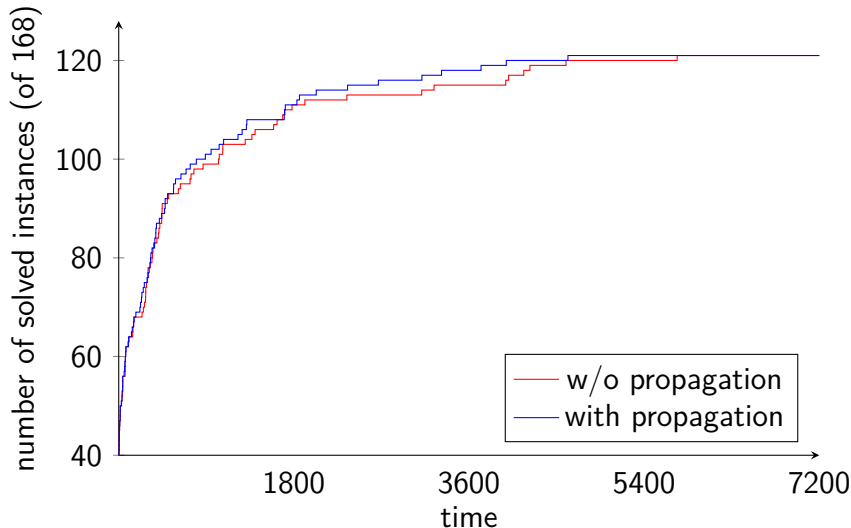
Number of Nodes: Hybrid Branching

- ▷ relative change of number of nodes when using SBDP
- ▷ only instances solved to optimality by both variants



- ▷ shifted geometric mean reduced by 7%

Performance Diagram: Hybrid Branching



▷ result: shifted geom. time reduced by 4%

Domain propagation in strong branching...

- ▷ increases strong branching time marginally
- ▷ improves dual bound predictions
- ▷ decreases tree size significantly
- ▷ improves performance of two common branching rules
- ▷ default in SCIP 3.1

Domain propagation in strong branching...

- ▷ increases strong branching time marginally
- ▷ improves dual bound predictions
- ▷ decreases tree size significantly
- ▷ improves performance of two common branching rules
- ▷ default in SCIP 3.1

Next Steps

- ▷ test on other problem types
- ▷ dynamic (de)activation of domain propagation
- ▷ nonchimerical + cloud + propagation = ?

Improving Strong Branching by Domain Propagation

Gerald Gamrath

Zuse Institute Berlin · Berlin Mathematical School

