Hybrid Branching

Timo Berthold

Zuse Institute Berlin

joint work with Tobias Achterberg (ILOG/IBM)

DFG Research Center MATHEON
Mathematics for key technologies

Chicago, 08/27/2009

How do we solve disc. opt. problems?

Mixed **I**nteger **P**rogramming

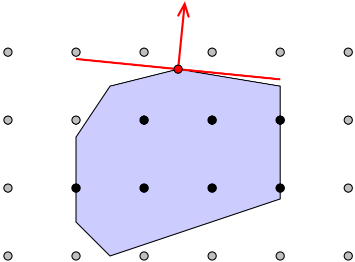
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint **P**rogramming

- Domain propagation
- Symmetry handling
- Branch-and-bound



How do we solve disc. opt. problems?

Mixed **I**nteger **P**rogramming

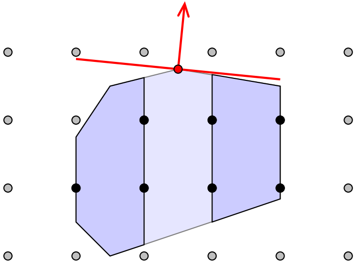
- ▷ LP relaxation
- ▷ Cutting planes
- ▷ Branch-and-bound

SATisfiability problems

- ▷ Conflict analysis
- ▷ Periodic restarts
- ▷ Branch-and-bound

Constraint **P**rogramming

- ▷ Domain propagation
- ▷ Symmetry handling
- ▷ Branch-and-bound



How do we solve disc. opt. problems?

Mixed Integer Programming

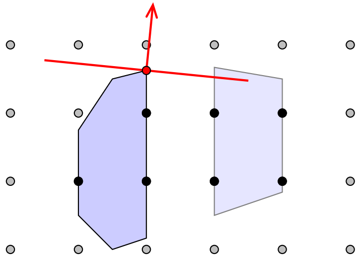
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint Programming

- Domain propagation
- Symmetry handling
- Branch-and-bound



How do we solve disc. opt. problems?

Mixed Integer Programming

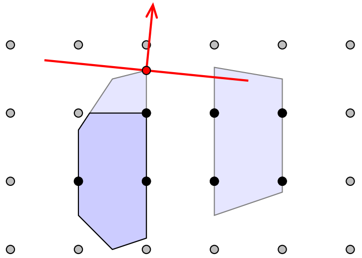
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint Programming

- Domain propagation
- Symmetry handling
- Branch-and-bound



How do we solve disc. opt. problems?

Mixed Integer Programming

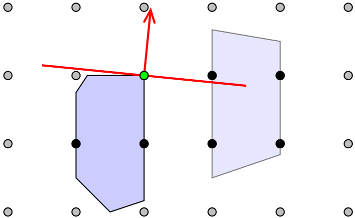
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint Programming

- Domain propagation
- Symmetry handling
- Branch-and-bound



How do we solve disc. opt. problems?

Mixed Integer Programming

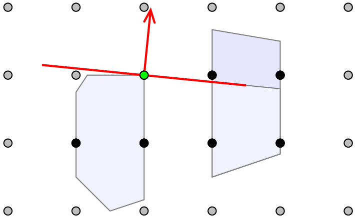
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint Programming

- Domain propagation
- Symmetry handling
- Branch-and-bound



How do we solve disc. opt. problems?

Mixed **I**nteger **P**rogramming

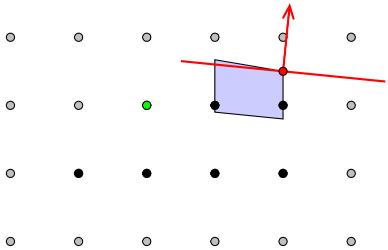
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint **P**rogramming

- Domain propagation
- Symmetry handling
- Branch-and-bound



Mixed Integer Programming

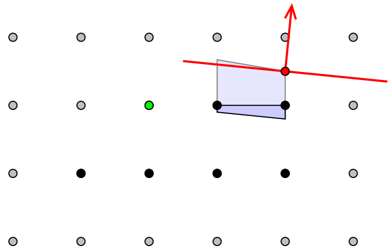
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint Programming

- Domain propagation
- Symmetry handling
- Branch-and-bound



Mixed Integer Programming

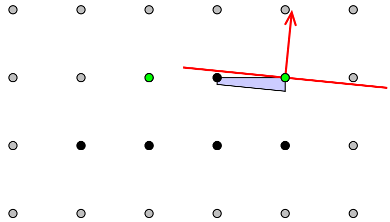
- LP relaxation
- Cutting planes
- Branch-and-bound

SATisfiability problems

- Conflict analysis
- Periodic restarts
- Branch-and-bound

Constraint Programming

- Domain propagation
- Symmetry handling
- Branch-and-bound



Same algorithm, different scope

Common goal: Keep branch-and-bound tree small!

MIP

- Improve LP value
- Reduce LP complexity

SAT

- Detect infeasibilities
- Generate short conflicts

CP

- Enable propagations
- Reduce domains

Same algorithm, different scope

Common goal: Keep branch-and-bound tree small!

MIP

- Improve LP value
- Reduce LP complexity

SAT

- Detect infeasibilities
- Generate short conflicts

CP

- Enable propagations
- Reduce domains

But

Also in MIP, sometimes...

- there's no objective
- instances are infeasible
- combinatorial structure

Same algorithm, different scope

Common goal: Keep branch-and-bound tree small!

MIP

- Improve LP value
- Reduce LP complexity

SAT

- Detect infeasibilities
- Generate short conflicts

CP

- Enable propagations
- Reduce domains

But

Also in MIP, sometimes...

- there's no objective
- instances are infeasible
- combinatorial structure

Standard MIP branching inferior in these cases

Most infeasible branching

- ▷ often referred to as a simple, standard rule
- ▷ computationally as bad as random branching!

Strong branching

- ▷ solve LP relaxations for some candidates, choose best
- ▷ effective w.r.t. number of nodes, expensive w.r.t. time

Pseudocost branching

- ▷ try to estimate LP values, based on history information
- ▷ effective, cheap, but weak in the beginning
- ▷ $\rightsquigarrow$ can/should be combined with strong branching

Branching rules in MIP

Most infeasible branching

- ▷ often referred to as a simple, standard rule
- ▷ computationally as bad as random branching!

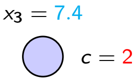
Strong branching

- ▷ solve LP relaxations for some candidates, choose best
- ▷ effective w.r.t. number of nodes, expensive w.r.t. time

Pseudocost branching

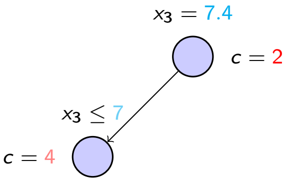
- ▷ try to estimate LP values, based on history information
- ▷ effective, cheap, but weak in the beginning
- ▷ $\rightsquigarrow$ can/should be combined with strong branching

Estimating the objective



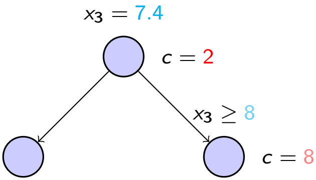
Estimating the objective

- objective gain per unit:
  - $\zeta^-(x_3) = \frac{4-2}{7.4-7} = \frac{2}{0.4} = 5$



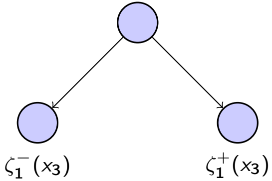
Estimating the objective

- objective gain per unit:
  - $\zeta^+(x_3) = \frac{8-2}{8-7.4} = \frac{6}{0.6} = 10$



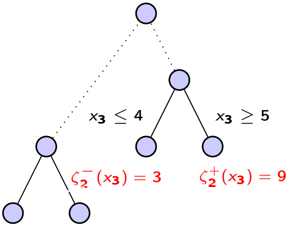
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$



Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes



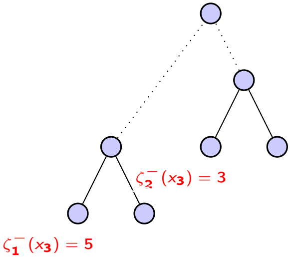
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes

- Pseudocosts:

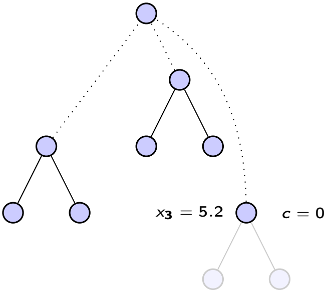
  average objective gain

  $\psi^-(x_3) = \frac{\zeta_1^-(x_3)+\ldots+\zeta_n^-(x_3)}{n} = \frac{5+3}{2} = 4$



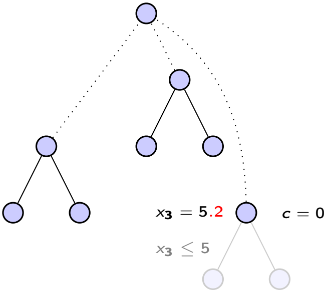
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- Pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- Estimate increase of objective by pseudocosts and fractionality:



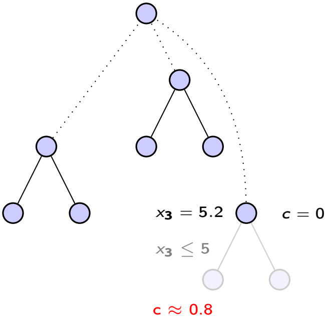
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- Pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- Estimate increase of objective
  by pseudocosts and fractionality:
  $\psi^-(x_3) \cdot \mathrm{frac}(x_3)$



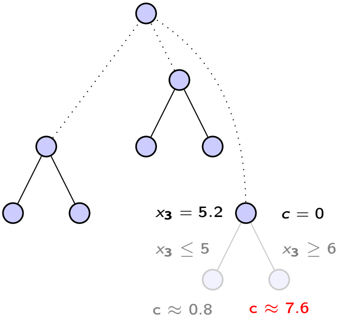
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- Pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- Estimate increase of objective
  by pseudocosts and fractionality:
  $\psi^-(x_3) \cdot \mathrm{frac}(x_3) = 4 \cdot 0.2 = 0.8$,



Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- Pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- Estimate increase of objective
  by pseudocosts and fractionality:
  $\psi^-(x_3) \cdot \mathrm{frac}(x_3) = 4 \cdot 0.2 = 0.8$,
  and $\psi^+(x_3)(1 - \mathrm{frac}(x_3)) = 7.6$



Early branchings are the most important ones!
Problem: In the beginning, pseudocosts are all zero

Pseudocost with strong branching initialization:

- Use strong branching, if pseudocosts have not been initialized yet

Reliability branching:

- Use strong branching, if pseudocosts are unreliable
- Unreliable: Pseudocosts have been updated less than k times
- Computational results: $k = 8$

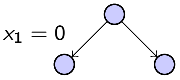
Inference branching:

- average number of applied domain deductions
- history based
- captures combinatorial structure
- estimates tightening of subproblems

Inference branching:

- average number of applied domain deductions
- history based
- captures combinatorial structure
- estimates tightening of subproblems

$$
\begin{aligned}
x_1 \qquad\quad + x_2 &= 1\\
x_1 + x_3 + x_4 &\le 1\\
-x_1 \qquad\quad + z &\ge 3\\
z &\in \mathbb{Z}_+\\
x_i &\in \{0,1\}
\end{aligned}
$$



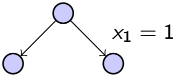
$$
\begin{aligned}
x_1 = 0 \Rightarrow x_2 &= 1\\
\Rightarrow z &\ge 3
\end{aligned}
$$

$$s_j^{\text{infer}}(-) = 2$$

Inference branching:

- average number of applied domain deductions
- history based
- captures combinatorial structure
- estimates tightening of subproblems

$$
\begin{aligned}
x_1 \quad\quad\quad + x_2 &= 1 \\
x_1 + x_3 + x_4 &\leq 1 \\
-x_1 \quad\quad\quad + z &\geq 3 \\
z &\in \mathbb{Z}_+ \\
x_i &\in \{0, 1\}
\end{aligned}
$$



$$
\begin{aligned}
x_1 = 1 \Rightarrow x_2 &= 0 \\
\Rightarrow x_3 &= 0 \\
\Rightarrow x_4 &= 0
\end{aligned}
$$

$$s_j^{\text{infer}}(+) = 3$$

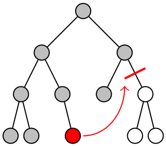
A general branching rule for CP

Inference branching:

- average number of applied domain deductions
- history based
- captures combinatorial structure
- estimates tightening of subproblems

- analogy to pseudocost values in MIP
- one value for upwards branch, one for downwards
- initialization: probing ($\approx$ strong branching)

A general branching rule for SAT

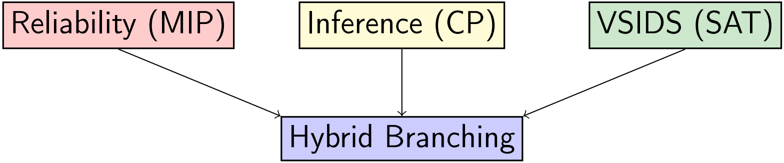


- ▷ important feature: conflict analysis / no-goods
- ▷ learning of small clauses which trigger infeasibility
- ▷ can be generalized to MIP, CP

VSIDS branching:

- ▷ largest number of (conflict) clauses, a variable appears in
- ▷ prefer "recent" conflicts
- ▷ "recent": exponentially decreasing importance
- ▷ works in particular well for infeasible problems
- ▷ state-of-the-art in SAT solving

Idea: Combine strategies to a new hybrid strategy for MIP



- use reliable pseudocosts, inference values, VSIDS
- additionally incorporate:
  - number of pruned subproblems
  - average length of conflict clauses

How the combination works:

- scaling: divide each value by average over all variables
- normalize each of the (scaled) values by $f : \mathbb{R}_{\geq 0} \rightarrow [0, 1), \; x \mapsto \frac{x}{x+1}$
- use a weighted sum of all criteria

How the combination works:

- scaling: divide each value by average over all variables
- normalize each of the (scaled) values by $f : \mathbb{R}_{\geq 0} \to [0, 1),\ x \mapsto \frac{x}{x+1}$
- use a weighted sum of all criteria

In formulae:

$$s_j = \omega^{\mathbf{pscost}} f\left(\frac{s_j^{\mathbf{pscost}}}{s_\varnothing^{\mathbf{pscost}}}\right) + \omega^{\mathbf{infer}} f\left(\frac{s_j^{\mathbf{infer}}}{s_\varnothing^{\mathbf{infer}}}\right) + \omega^{\mathbf{vsids}} f\left(\frac{s_j^{\mathbf{vsids}}}{s_\varnothing^{\mathbf{vsids}}}\right) + \omega^{\mathbf{prune}} f\left(\frac{s_j^{\mathbf{prune}}}{s_\varnothing^{\mathbf{prune}}}\right) + \omega^{\mathbf{conf}} f\left(\frac{s_j^{\mathbf{conf}}}{s_\varnothing^{\mathbf{conf}}}\right)$$

How the combination works:

- scaling: divide each value by average over all variables
- normalize each of the (scaled) values by $f : \mathbb{R}_{\geq 0} \to [0,1),\; x \mapsto \frac{x}{x+1}$
- use a weighted sum of all criteria

In formulae:

$$s_j = \omega^{\textbf{pscost}} f\left(\frac{s_j^{\textbf{pscost}}}{s_\varnothing^{\textbf{pscost}}}\right) + \omega^{\textbf{infer}} f\left(\frac{s_j^{\textbf{infer}}}{s_\varnothing^{\textbf{infer}}}\right) + \omega^{\textbf{vsids}} f\left(\frac{s_j^{\textbf{vsids}}}{s_\varnothing^{\textbf{vsids}}}\right) + \omega^{\textbf{prune}} f\left(\frac{s_j^{\textbf{prune}}}{s_\varnothing^{\textbf{prune}}}\right) + \omega^{\textbf{conf}} f\left(\frac{s_j^{\textbf{conf}}}{s_\varnothing^{\textbf{conf}}}\right)$$

Choice for the weights:

- high weight for pseudocosts: 1
- medium weight for VSIDS and conflict length: 10^{-2} and 10^{-3}, resp.
- low weight for inference and cutoff values: 10^{-4}

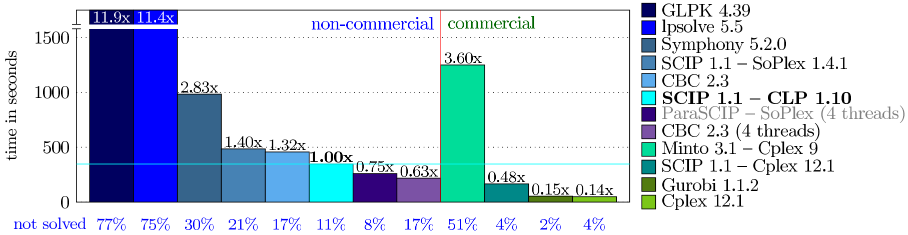
Branching score function

- yields two values: One for downwards, one for upwards branching
- need to combine them to a single value
- usually: convex sum
- includes minimum and maximum as extreme cases

- we use: multiplication
  $\text{score}(x_j) = \max\{s_j^-, \epsilon\} \cdot \max\{s_j^+, \epsilon\}$ $(\epsilon = 10^{-6})$
- computational results: 10% faster

SCIP: Solving Constraint Integer Programs

- ▷ Standalone solver / Branch-Cut-And-Price-Framework
- ▷ Combines methods from MIP, CP, SAT
- ▷ modular structure via plugins
- ▷ Free for academic use: `http://scip.zib.de`
- ▷ Very fast non-commercial MIP solver



Computational results

Test set:

- MIPLIB2003: Selection of 60 quite different, difficult instances
- Cor@l: Huge collection of 350 instances
- Cor@l-BP: The 118 pure 0/1-programs of the Cor@l test set
- Infeasible: 30 infeasible graph coloring instances

Comparison:

- geometric means of overall running time and number of branch-and-bound nodes
- ratio between hybrid branching and reliability branching

Computational results II

test set	MIPLIB2003		Cor@l	
	Time	Nodes	Time	Nodes
reliability	450.4	5091	803.6	4110
hybrid	445.6	5051	735.0	3575
ratio reli/hyb	1.01	1.01	1.09	1.15

Result: No difference / slight improvement for general MIPs

test set	Cor@l-BP		Infeasible	
	Time	Nodes	Time	Nodes
reliability	672.4	2145	290.7	5612
hybrid	577.2	1681	166.0	1998
ratio reli/hyb	1.16	1.28	1.75	2.81

Result: Medium / large improvement for special MIPs

Conclusion: Hybrid branching...

- is a successful integration of CP, SAT and MIP technologies
- works particular well for problem classes, where classical MIP branching “fails”
- is now used as default branching rule in SCIP

Outlook

- Use different weights for general MIP / BPs
- Switch weights if instance is suspected to be infeasible
- Generalize to MINLP

Hybrid Branching

Timo Berthold
Zuse Institute Berlin

joint work with Tobias Achterberg (ILOG/IBM)

DFG Research Center MATHEON
Mathematics for key technologies

Chicago, 08/27/2009