Strong(er) Branching for Mixed Integer Programming

Wasu Glankwamdee

Jeff Linderoth

ISE Department
COR@L Lab
Lehigh University
jtl3@lehigh.edu

pgflastimage

Workshop on Hybrid Methods and Branching Rules in Combinatorial Optimization
Centre de recherches mathématiques
Université de Montréal
September 18, 2006

Branch and Bound for MIP

MIP

$$
\begin{aligned}
z_{MIP} &\stackrel{\text{def}}{=} \max_{(x,y)\in S} \{c^T x + h^T y\} \\
S &= \{(x,y) \in \mathbb{Z}_+^{|I|} \times \mathbb{R}_+^{|C|} \mid Ax + Gy \leq b\} \\
R(S) &= \{(x,y) \in \mathbb{R}_+^{|I|} \times \mathbb{R}_+^{|C|} \mid Ax + Gy \leq b\}
\end{aligned}
$$

Branch and Bound for MIP

MIP

$$z_{MIP} \stackrel{\text{def}}{=} \max_{(x,y)\in S} \{c^T x + h^T y\}$$

$$S = \{(x,y) \in \mathbb{Z}_+^{|I|} \times \mathbb{R}_+^{|C|} \mid Ax + Gy \leq b\}$$

$$R(S) = \{(x,y) \in \mathbb{R}_+^{|I|} \times \mathbb{R}_+^{|C|} \mid Ax + Gy \leq b\}$$

Bounds

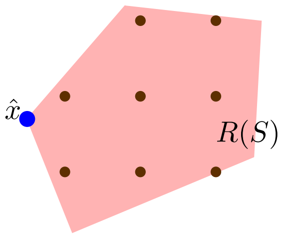
- Upper:

$$z_{LP} \stackrel{\text{def}}{=} \max_{(x,y)\in R(S)} \{c^T x + h^T y\} \geq z_{MIP}$$

- Lower:

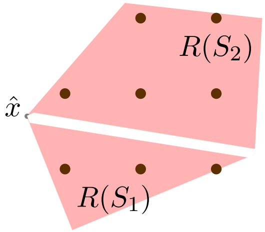
$$(\hat{x}, \hat{y}) \in S \Rightarrow c^T \hat{x} + h^T \hat{y} \leq z_{MIP}$$

Branch-and-Bound



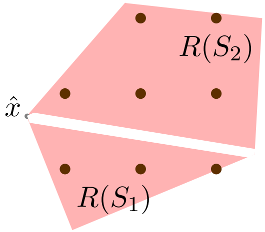
1. Solve for z_{LP}, $\hat{x}$

Branch-and-Bound



1. Solve for z_{LP}, $\hat{x}$
2. Branch: Exclude $\hat{x}$ but no points in S

Branch-and-Bound



1. Solve for z_{LP}, $\hat{x}$
2. Branch: Exclude $\hat{x}$ but no points in S
3. Lather, Rinse, Repeat!

On My High Horse

- There is very little one (specifically me) can say in the way of “proofs” for branching methods.
- What follows is mostly my own opinion, backed up with a little empirical experience.

On My High Horse

- There is very little one (specifically me) can say in the way of “proofs” for branching methods.
- What follows is mostly my own opinion, backed up with a little empirical experience.
- We branch only on variables
- For $j \in I$ with $f(\hat{x}_j) > 0$:

$$S = \{x \in S \mid x_j \leq \lfloor \hat{x}_j \rfloor\} \cup \{x \in S \mid x_j \geq \lceil \hat{x}_j \rceil\}$$

$$S = S_1 \cup S_2$$

Riding That High Horse

pgflastimage

- I believe for a majority of problems, this is the way to do it.
- The natural disjunctions are often along the axes of the decision problems.

Riding That High Horse

pgflastimage

- I believe for a majority of problems, this is the way to do it.
- The natural disjunctions are often along the axes of the decision problems.
- We can assume this to be true until Friday:
- Andrea Lodi, *About Branching on General Disjunctions*

Riding That High Horse

pgflastimage

- I believe for a majority of problems, this is the way to do it.
- The natural disjunctions are often along the axes of the decision problems.
- We can assume this to be true until Friday:
- Andrea Lodi, *About Branching on General Disjunctions*
    - Thankfully, I will be long gone!

Still on My High Horse

The Goal of Branching

Reduce the upper bound as much as possible

pgflastimage

Still on My High Horse

The Goal of Branching

Reduce the upper bound as much as possible

pgflastimage

- Branching to get feasible solutions (improving lower bound) is likely better done by another "heuristic" process (e.g "Local Branching")

Still on My High Horse

The Goal of Branching

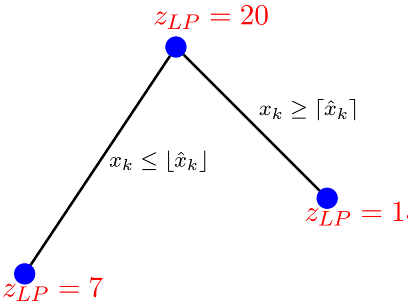
Reduce the upper bound as much as possible

pgflastimage

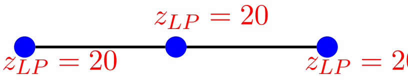
- Branching to get feasible solutions (improving lower bound) is likely better done by another “heuristic” process (e.g “Local Branching”)
- We can at least assume this to be true until tomorrow:
- John Chinneck, *Active-Constraint Variable Ordering for Faster Feasibility of Mixed Integer Linear Programs*

Some Branching Facts

1. An Example Branch

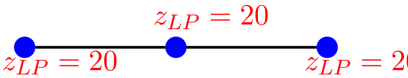


Some Branching Facts



1. An Example Branch
2. A bad branch.
   - The amount of work for this subtree has doubled

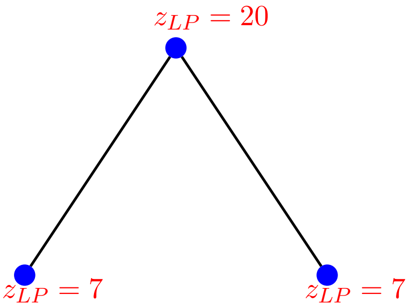
Some Branching Facts



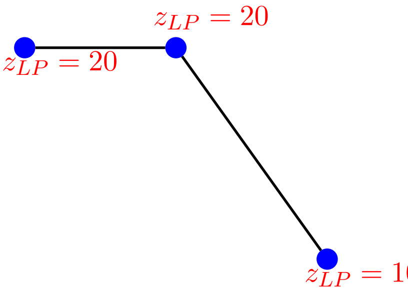
1. An Example Branch
2. A bad branch.
   - The amount of work for this subtree has doubled
3. Reducing upper bound vs. increasing lower bound:
   - These are somewhat conflicting goals

Proof By Picture

1. Improving Upper Bound: Make sure that your branching decision has a big impact on both children
   - Now our upper bound is 7



Proof By Picture



1. Improving Upper Bound: Make sure that your branching decision has a big impact on both children
   - Now our upper bound is 7
2. Improving Lower Bound: Make sure that your branching decision has little impact on at least one child
   - You still have "the same" amount of work to do on the left branch

A Natural Branching Idea

- To make bound go down on both branches, choose to branch on the "most fractional" variable

$$j \in \arg\min_I \{|f(\hat{x}_j) - 0.5|\}.$$

$f(z)$: Fractional part of z

A Natural Branching Idea

- To make bound go down on both branches, choose to branch on the "most fractional" variable

$$j \in \arg\min_{I} \{|f(\hat{x}_j) - 0.5|\}.$$

$f(z)$: Fractional part of z

Nature Is Bad!

Most fractional branching is no better than choosing a random fractional variable to branch on!

Alex Martin, MIP'06

A Better Branching Idea: Pseudocosts

- Keep track of the impact of branching on x_j:

$$z_j^- \stackrel{\text{def}}{=} \max_{x \in R(S) \,\cap\, x_j \leq \lfloor \hat{x}_j \rfloor} \{c^T x + h^T y\} \qquad z_j^+ \stackrel{\text{def}}{=} \max_{x \in R(S) \,\cap\, x_j \geq \lceil \hat{x}_j \rceil} \{c^T x + h^T y\}$$

$$P_j^- = \frac{z_{LP} - z_j^-}{f(\hat{x}_j)} \qquad P_j^+ = \frac{z_{LP} - z_j^+}{1 - f(\hat{x}_j)}$$

A Better Branching Idea: Pseudocosts

- Keep track of the impact of branching on x_j:

$$z_j^- \stackrel{\text{def}}{=} \max_{x \in R(S) \,\cap\, x_j \leq \lfloor \hat{x}_j \rfloor} \{c^T x + h^T y\} \qquad z_j^+ \stackrel{\text{def}}{=} \max_{x \in R(S) \,\cap\, x_j \geq \lceil \hat{x}_j \rceil} \{c^T x + h^T y\}$$

$$P_j^- = \frac{z_{LP} - z_j^-}{f(\hat{x}_j)} \qquad P_j^+ = \frac{z_{LP} - z_j^+}{1 - f(\hat{x}_j)}$$

- When you choose to branch on x_j (with value x'_j) again, compute estimated LP decreases as

$$D_j^- = P_j^- f(x'_j) \qquad D_j^+ = P_j^+ (1 - f(x'_j))$$

A Better Branching Idea: Pseudocosts

- Keep track of the impact of branching on x_j:

$$z_j^- \stackrel{\text{def}}{=} \max_{x \in R(S)\ \cap\ x_j \leq \lfloor \hat{x}_j \rfloor} \{c^T x + h^T y\} \qquad z_j^+ \stackrel{\text{def}}{=} \max_{x \in R(S)\ \cap\ x_j \geq \lceil \hat{x}_j \rceil} \{c^T x + h^T y\}$$

$$P_j^- = \frac{z_{LP} - z_j^-}{f(\hat{x}_j)} \qquad P_j^+ = \frac{z_{LP} - z_j^+}{1 - f(\hat{x}_j)}$$

- When you choose to branch on x_j (with value x'_j) again, compute estimated LP decreases as

$$D_j^- = P_j^- f(x'_j) \qquad D_j^+ = P_j^+ (1 - f(x'_j))$$

Problem!?

What do you use initially!

Just Do It

pgflastimage

- Initialize pseudocosts by <u>explicity</u> computing them for all yet-to-be-branched-on variables

Just Do It

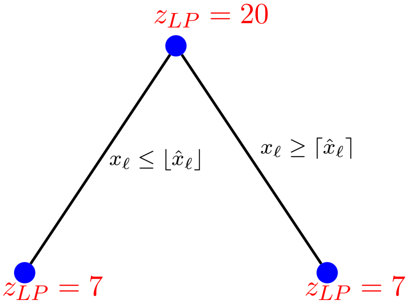
- Initialize pseudocosts by <u>explicity</u> computing them for all yet-to-be-branched-on variables
- With a little imagination, this is a branching method in and of itself: Strong Branching.

(Full) Strong Branching

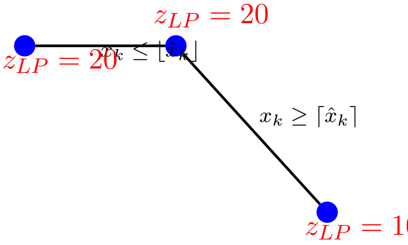
1. At each node n at which a branching decision must be made:

(Full) Strong Branching

1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+



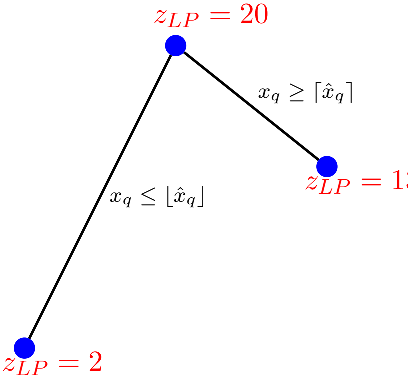
(Full) Strong Branching



1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+

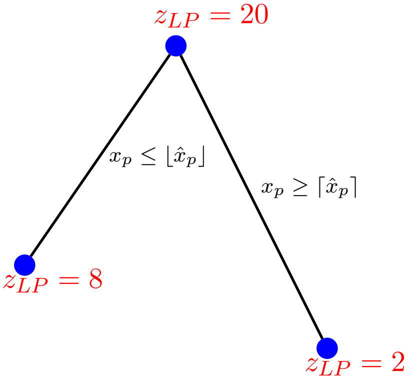
(Full) Strong Branching

1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+



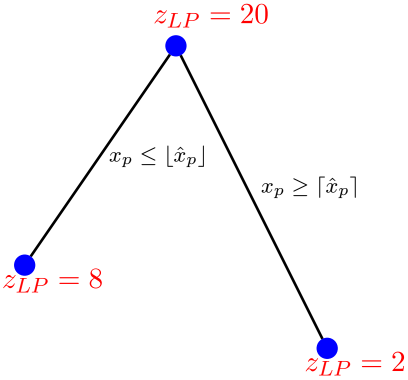
(Full) Strong Branching

1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+

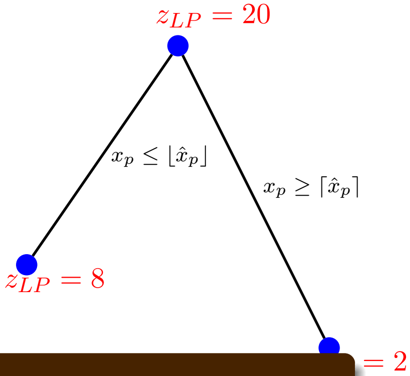


(Full) Strong Branching

1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+
3. Branch on $\max_{j \in \mathcal{F}_n} f(z_j^-, z_j^+)$



(Full) Strong Branching



1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+
3. Branch on $\max_{j \in \mathcal{F}_n} f(z_j^-, z_j^+)$

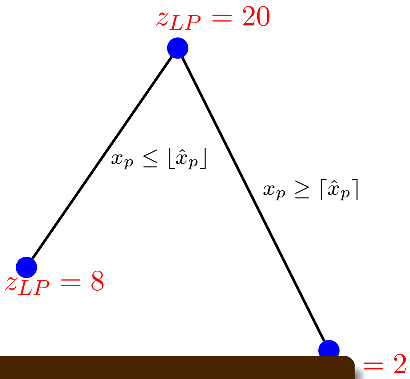
How To Combine?

- Try the weighting function $\mathcal{W}(z_{LP} - z_i^-, z_{LP} - z_i^+)$ for

$$\mathcal{W}(a, b) \stackrel{\text{def}}{=} \{\alpha_1 \min(a, b) + \alpha_2 \max(a, b)\},$$

- $\alpha_1 = 3.7214541, \alpha_2 = 1$ seems to work OK.

(Full) Strong Branching



1. At each node n at which a branching decision must be made:
2. For each $j \in \mathcal{F}_n$: Compute z_j^-, z_j^+
3. Branch on $\max_{j \in \mathcal{F}_n} f(z_j^-, z_j^+)$

How To Combine?

- Try the weighting function $\mathcal{W}(z_{LP} - z_i^-, z_{LP} - z_i^+)$ for

$$\mathcal{W}(a, b) \stackrel{\text{def}}{=} \{\alpha_1 \min(a, b) + \alpha_2 \max(a, b)\},$$

- $\alpha_1 = 3.7214541, \alpha_2 = 1$ seems to work OK.

Don't ignore the second child completely

Speeding up Strong Branching

Obvious Ideas

1. Limit number of pivots β
   - Like Driebeek/Tomlin "penalties"
2. Limit Candidate Set $|C|$

Speeding up Strong Branching

Obvious Ideas

1. Limit number of pivots β
   - Like Driebeek/Tomlin "penalties"
2. Limit Candidate Set $|C|$

Good Ideas!

1. $Q-$phase selection
   - $C_1 \supseteq C_2 \supseteq C_3 \supseteq \ldots \supseteq C_Q$
   - $\beta_1 \leq \beta_2 \leq \beta_3 \leq \ldots \leq \beta_Q$
2. Limit number of times that you perform strong branching on any variable, then "switch" to pseudocosts.
   - Reliability branching (Achterberg, Koch, Martin)

Strong Branching is Quite Effective

1. Branching Option in all high-quality MIP solvers
2. Being used for many (non-MILP) enumeration problems where bounds are computed

Strong Branching is Quite Effective

1. Branching Option in all high-quality MIP solvers
2. Being used for many (non-MILP) enumeration problems where bounds are computed

---

Non MILP Strong Branching Users

- Anstreicher *et al.*: Large quadratic assignment problem calculations
- Vandenbussche: Branching on complementarity condition for nonconvex QP
- Anstreicher & Fampa: Enumerating Steiner Topologies
  - Choose to include new node (to Steiner Tree) for which the number of (non-fathomed) child nodes (in the enumeration tree) is as small as possible.

The History of Strong Branching

```
From:  Jeff Linderoth <jtl3@lehigh.edu>
To:  bico@isye.gatech.edu
Subject:  Strong Branching
Date:  13 May 2003 13:18:02 -0400

Hello Bill,

I'm sorry to bother you, but I have a question, and I think you may be
the person who knows the answer...

A colleague of mine referred to strong branching as being
"invented" by CPLEX.  It has often troubled me that there seems to be
many different citations about the origins of the idea of strong
branching.  Can you set the record straight for me?  If you had to
provide one citation to strong branching, what would it be?

Thanks very much.  I hope all is well.
```

The Response

```
From:  Jeff Linderoth <jtl3@lehigh.edu>
To:  William Cook <bico@isye.gatech.edu>
Subject:  Re:  Strong Branching
Date:  14 May 2003 10:36:11 -0400
```

```
The way strong branching developed is that we were not happy with the choices of
branching edges we were finding in our TSP code, and I asked Bob whether he
could set up some limited LP solve that we could use to get a better indication
of whether the LP would move after setting the branching variable in both
directions.  (Dave and I had carried out some experiments showing that solving
the LP completely did give good information (keep in mind that we only have a
subset of edges in the LP, so I only mean solving with the active set not with
pricing).)  Bix then set up the interface for doing a limited number of pivots
and we played around a bit and came up with a set of paramters that seemed to
work well for the TSP.
```

The Response

```
From:  Jeff Linderoth <jtl3@lehigh.edu>
To:  William Cook <bico@isye.gatech.edu>
Subject:  Re:  Strong Branching
Date:  14 May 2003 10:36:11 -0400

The way strong branching developed is that we were not happy with the choices of
branching edges we were finding in our TSP code, and I asked Bob whether he
could set up some limited LP solve that we could use to get a better indication
of whether the LP would move after setting the branching variable in both
directions.  (Dave and I had carried out some experiments showing that solving
the LP completely did give good information (keep in mind that we only have a
subset of edges in the LP, so I only mean solving with the active set not with
pricing).)  Bix then set up the interface for doing a limited number of pivots
and we played around a bit and came up with a set of paramters that seemed to
work well for the TSP.
```

The Moral Of The Story

1. Don't cite CPLEX as the originator of "strong branching"

The Response

```
From:  Jeff Linderoth <jtl3@lehigh.edu>
To:  William Cook <bico@isye.gatech.edu>
Subject:  Re:  Strong Branching
Date:  14 May 2003 10:36:11 -0400

The way strong branching developed is that we were not happy with the choices of
branching edges we were finding in our TSP code, and I asked Bob whether he
could set up some limited LP solve that we could use to get a better indication
of whether the LP would move after setting the branching variable in both
directions.  (Dave and I had carried out some experiments showing that solving
the LP completely did give good information (keep in mind that we only have a
subset of edges in the LP, so I only mean solving with the active set not with
pricing).)  Bix then set up the interface for doing a limited number of pivots
and we played around a bit and came up with a set of paramters that seemed to
work well for the TSP.
```

The Moral Of The Story

1. Don't cite CPLEX as the originator of "strong branching"
2. But even more so, Don't cite Linderoth and Savelsbergh!

Good Thinking

1. Full Strong Branching can be very good
2. It takes too long

Good Thinking

1. Full Strong Branching can be very good
2. It takes too long

pgflastimage

A Smart Idea!

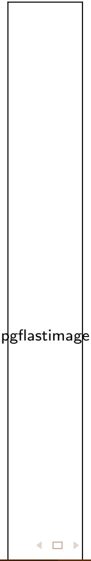
Let's Speed it up.

Bad Thinking

- Full Strong Branching can be very good
- It takes too long
- A Smart Idea: Let's Speed it up.

A Stupid Idea

Let's Slow it down!



Strong Branching Analogies

- "Strong branching is like dating many women before you finally decide to whom to commit." (Vasek Chvátal)

Strong Branching Analogies

- “Strong branching is like dating many women before you finally decide to whom to commit.” (Vasek Chvátal)
- This does not ring true with me, as...
    1. Given my general lack of pulchritude and savoir-faire, I was unable to date many women

Strong Branching Analogies

- "Strong branching is like dating many women before you finally decide to whom to commit." (Vasek Chvátal)
- This does not ring true with me, as...
  1. Given my general lack of pulchritude and savoir-faire, I was unable to date many women
  2. I've been happily married for seven years

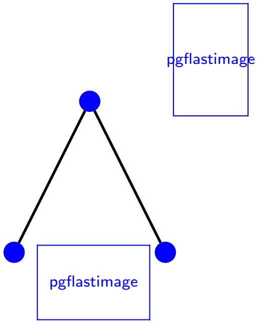
pgflastimage

pgflastimage

pgflastimage

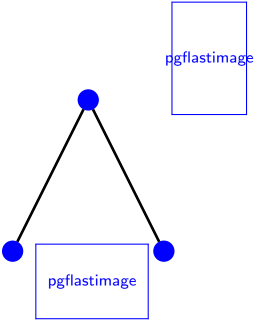
A Wishful Analogy From a New Parent

I (sometimes) wish the same opportunity was available for children



A Wishful Analogy From a New Parent

I (sometimes) wish the same opportunity was available for children

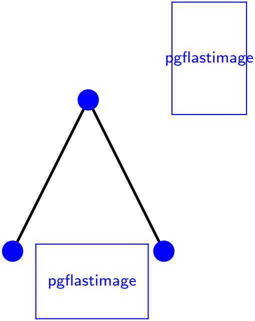


So if your child is...

- A Crybaby

A Wishful Analogy From a New Parent

I (sometimes) wish the same opportunity was available for children

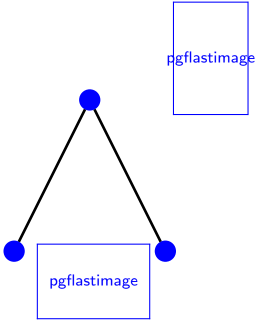


So if your child is...

- A Crybaby
- A Nosepicker

A Wishful Analogy From a New Parent

I (sometimes) wish the same opportunity was available for children

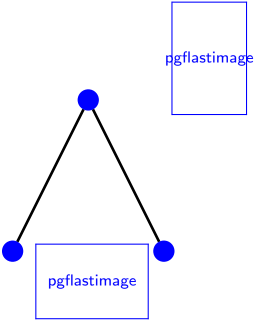


So if your child is...

- A Crybaby
- A Nosepicker
- A Giant Buffoon

A Wishful Analogy From a New Parent

I (sometimes) wish the same opportunity was available for children



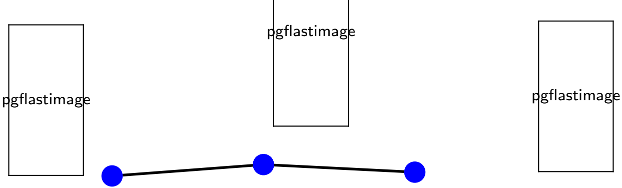
So if your child is...

- A Crybaby
- A Nosepicker
- A Giant Buffoon

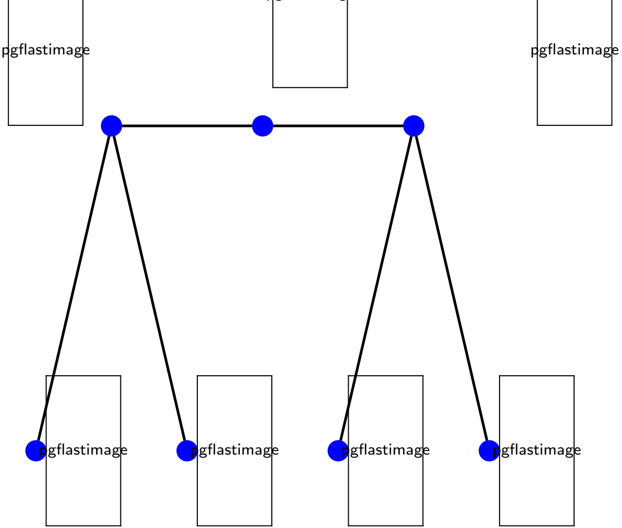
You Get To Pick Again!

Life Perspectives

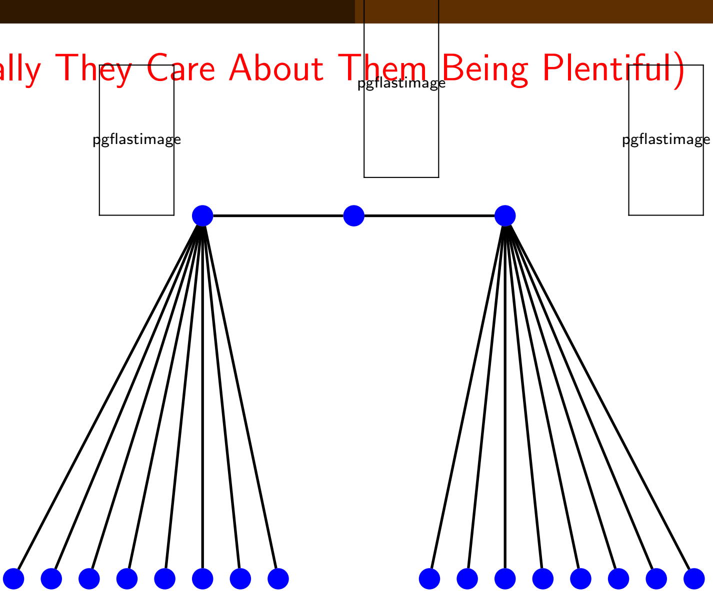
- I've come to realize that this perspective is short-sighted. Parents (specifically mine), don't care if their children are bad



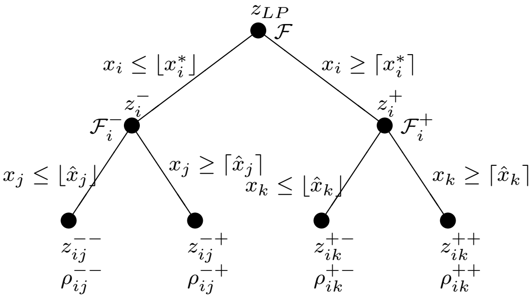
They Care Only That Their Grandchildren are GOOD!



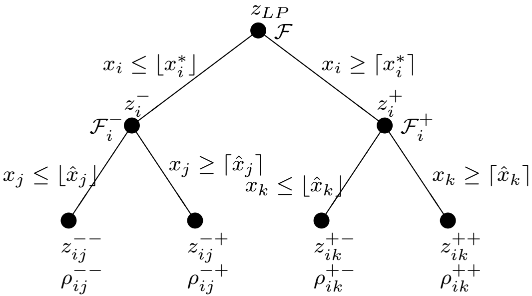
(Actually They Care About Them Being Plentiful)



Apply it To MIP: Grandchild Branching



Apply it To MIP: Grandchild Branching



- Let's just assume that all we care to find out first is if grandchild information can be helpful.
- We don't care how long it will take to generate the information

Why It Might Be Reasonable

- Maybe we only need to do this at the top of the tree
  - Recall: Branching decisions at the top of the tree are by far the most important

Why It Might Be Reasonable

- Maybe we only need to do this at the top of the tree
  - Recall: Branching decisions at the top of the tree are by far the most important
- Suppose you own a massively parallel machine, like Blue Gene or "The Grid"
  - Then during B&B "rampup", you don't have anything for most of the processors to do
  - You can afford to be ~~stupid~~ experimental

Why It Might Be Reasonable

- Maybe we only need to do this at the top of the tree
  - Recall: Branching decisions at the top of the tree are by far the most important
- Suppose you own a massively parallel machine, like Blue Gene or "The Grid"
  - Then during B&B "rampup", you don't have anything for most of the processors to do
  - You can afford to be ~~stupid~~ experimental

Don't Do It!!!

For the next few slides, no one is allowed to say:

- "It will talk too long"
- "Why didn't you try X?"

How To Use All This Information?

1. We've got four numbers for lots of pairs of variables (i, j). How do we use this to choose one variable to branch on?
2. Also, there are some pairs of variables that lead to infeasible grandchildren. How can we use this information?

How To Use All This Information?

1. We've got four numbers for lots of pairs of variables (i, j). How do we use this to choose one variable to branch on?
2. Also, there are some pairs of variables that lead to infeasible grandchildren. How can we use this information?

A Confusing Formula: Weighted Combination

$$i^* \in \arg\max_{i \in \mathcal{F}} \left\{ \mathcal{W}\left(\max_{j \in \mathcal{F}_i^-} \{\mathcal{W}(D_{ij}^{--}, D_{ij}^{-+})\}, \max_{j \in \mathcal{F}_i^+} \{\mathcal{W}(D_{ij}^{+-}, D_{ij}^{++})\} \right) + \lambda \eta_i \right\},$$

- η_i: Total number of infeasible grandchildren

$$\eta_i \stackrel{\text{def}}{=} \sum_{j \in \mathcal{F}_i^-} (\rho_{ij}^{--} + \rho_{ij}^{-+}) + \sum_{j \in \mathcal{F}_i^+} (\rho_{ij}^{+-} + \rho_{ij}^{++})$$

- λ: "coefficient of infeasibility"

Test Suite #1

Name	Rows	Columns	# Integer Variable	# Binary Variables	# Continuous Variables
bell3a	123	133	71	39	62
blend2	274	353	264	231	89
l152lav	97	1989	1989	ALL	0
p0548	176	548	548	ALL	0
rgn	24	180	100	ALL	80
stein45	331	45	45	ALL	0
vpm2	234	378	168	ALL	210
misc07	212	260	259	ALL	1
modglob	291	422	98	ALL	324
opt1217	64	769	768	ALL	1
p2756	755	2756	2756	ALL	0
pk1	45	86	55	ALL	31
pp08a	136	240	64	ALL	176
aflow30a	479	842	421	ALL	421
aflow40b	1442	2728	1364	ALL	1364
danoint	664	521	56	ALL	465
gesa2	1392	1224	408	240	816
qiu	1192	840	48	ALL	792
swath	884	6805	6724	ALL	81
timtab1	171	397	171	64	226

Preliminary Computational Results

pgflastimage

	Solved	Unsolved
	Avg # Evaluated Nodes	Avg Integrality Gap
MINTO Default	16974	20.34%
Full Strong	8471	45.36%
GrandChild	8004	43.02%

Preliminary Computational Results

pgflastimage

	Solved	Unsolved
	Avg # Evaluated Nodes	Avg Integrality Gap
MINTO Default	16974	20.34%
Full Strong	8471	45.36%
GrandChild	8004	43.02%

- Run all instances for eight hours (on slowish machine)

But it IS Doing Something Different

- Percentage of Time Grandchild Branching Makes a Different Selection Than Full Strong Branching

Name	% Diff
bell3a	0.27
blend2	0.67
l152lav	0.52
p0548	1
rgn	0.65
stein45	0.58
vpm2	0.54
misc07	0.60
modglob	0.49
opt1217	0.25

Name	% Diff
p2756	0
pk1	0.62
pp08a	0.76
aflow30a	0.62
aflow40b	0.46
danoint	0.84
gesa2	0.57
qiu	0.91
swath	0.72
timtab1	0.74

How Can We Improve It?

How Can We Improve It?

Fix All The Variables You Can!

Bound Fixing

Condition	Bound
$\xi_i^- = 1$	$x_i \geq \lceil x_i^* \rceil$
$\xi_i^+ = 1$	$x_i \leq \lfloor x_i^* \rfloor$
$\rho_{ij}^{--} = 1$ and $\rho_{ij}^{-+} = 1$	$x_i \geq \lceil x_i^* \rceil$
$\rho_{ij}^{+-} = 1$ and $\rho_{ij}^{++} = 1$	$x_i \leq \lfloor x_i^* \rfloor$

How Can We Improve It?

Fix All The Variables You Can!

Bound Fixing

Condition	Bound
$\xi_i^- = 1$	$x_i \geq \lceil x_i^* \rceil$
$\xi_i^+ = 1$	$x_i \leq \lfloor x_i^* \rfloor$
$\rho_{ij}^{--} = 1$ and $\rho_{ij}^{-+} = 1$	$x_i \geq \lceil x_i^* \rceil$
$\rho_{ij}^{+-} = 1$ and $\rho_{ij}^{++} = 1$	$x_i \leq \lfloor x_i^* \rfloor$

Deduce All The Implications You Can!

Implications

Condition	Implication
$\rho_{ij}^{--} = 0$	$(1 - x_i) + (1 - x_j) \leq 1$
$\rho_{ij}^{+-} = 0$	$(1 - x_i) + x_j \leq 1$
$\rho_{ij}^{+-} = 0$	$x_i + (1 - x_j) \leq 1$
$\rho_{ij}^{++} = 0$	$x_i + x_j \leq 1$

Computational Results: Using All Information

	Solved	Unsolved
	Avg # Evaluated Nodes (Old)	Avg Integrality Gap (Old)
MINTO Default	16974	20.34
Full Strong	2590 (8471)	14.1% (45.36%)
GrandChild	946 (8004)	9.22% (43.02%)

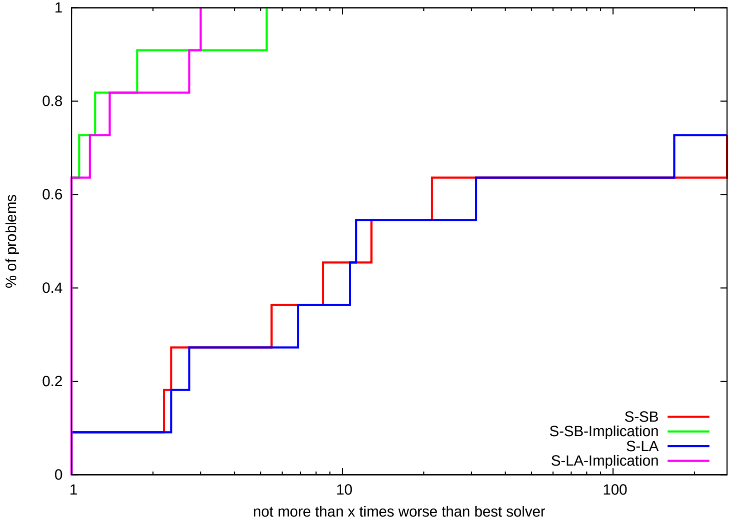
Amount of Additional Information

Name	Number of Implications	% Vars Fixed
l152lav	2408	0.13
p0548	46	0.55
rgn	658	0.55
stein45	455243	3.33
vpm2	3171	0.22
misc07	24556	1.06
modglob	841	0.29
opt1217	0	0.13
p2756	45	0.02
pk1	280189	1.04
pp08a	10058	0.42
aflow30a	1500	0.13
aflow40b	2892	0.04
danoint	47	0.18
qiu	603	0.43
swath	1877	0.02

Performance Profiles

- A relative measure of the effectiveness of a solver s when compared to a group of solvers $\mathcal{S}$ on a set of problem instances $\mathcal{P}$.
  - γ_{ps}: *quality measure* of solver $s \in \mathcal{S}$ when solving problem $p \in \mathcal{P}$
  - $r_{ps} = \gamma_{ps}/(\min_{s \in \mathcal{S}} \gamma_{ps})$
  - $\rho_s(\tau) = |\{p \in \mathcal{P} \mid r_{ps} \leq \tau\}| / |\mathcal{P}|$.
- $\rho_s(\tau)$: fraction of instances for which the performance of solver s was within a factor of τ of the best.
- A performance profile for solver s is the graph of $\rho_s(\tau)$.
- In general, the "higher" the graph of a solver, the better the relative performance.

Does it work? Solved Instances. Number of Nodes

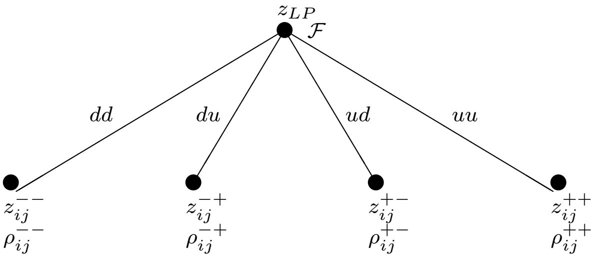


Can We Speed It Up?

Can We Speed It Up?

Duh!

- Just branch on two variables at once
- Subsequent experiments: Larger test suite, time only



$$dd = x_i \leq \lfloor x_i^* \rfloor, x_j \leq \lfloor x_j^* \rfloor$$
$$du = x_i \leq \lfloor x_i^* \rfloor, x_j \geq \lceil x_j^* \rceil$$
$$ud = x_i \geq \lceil x_i^* \rceil, x_j \leq \lfloor x_j^* \rfloor$$
$$uu = x_i \geq \lceil x_i^* \rceil, x_j \geq \lceil x_j^* \rceil$$

First, A Reasonable Strong Branching

SB(α, β)

1. Limit the size of the candidate set to

$$|C| = \max\{\alpha|\mathcal{F}|, 10\}$$

   (Ranked by fractionality)

2. Then do β pivots on both children
3. Choose best variable based on $\mathcal{W}(z_{LP} - z_i^-, z_{LP} - z_i^+)$

First, A Reasonable Strong Branching

SB(α, β)

1. Limit the size of the candidate set to

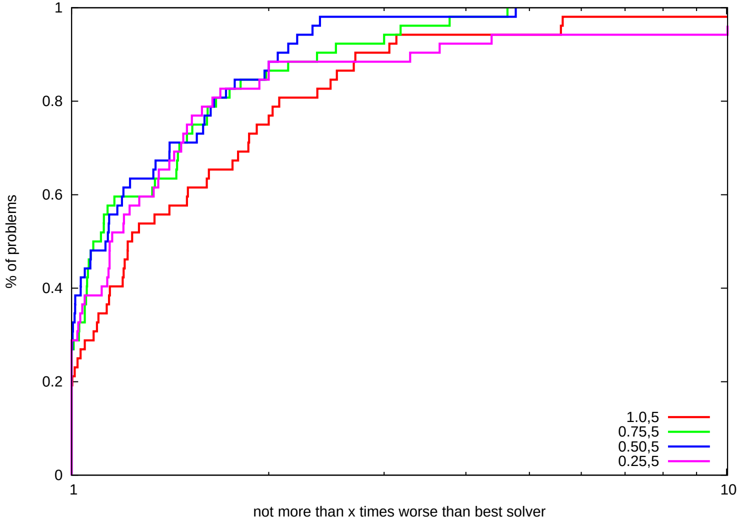
$$|C| = \max\{\alpha|\mathcal{F}|, 10\}$$

   (Ranked by fractionality)
2. Then do β pivots on both children
3. Choose best variable based on $\mathcal{W}(z_{LP} - z_i^-, z_{LP} - z_i^+)$

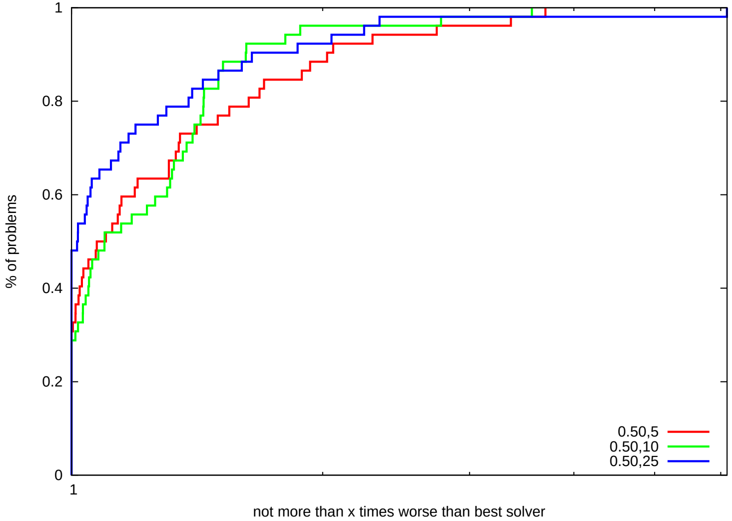
The $.64 Question for SB

What are "good" values of α, β?

$\beta = 5$, Find Good α



$\alpha = 0.5$, Find Good β



Parameters for Grandchild Branching

LA($\alpha, \beta, \gamma, \delta$)

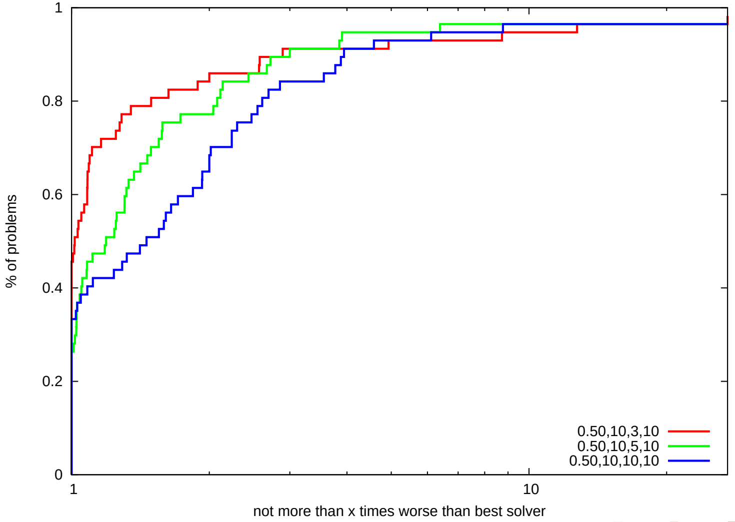
1. Do SB(α, β).
2. Choose best γ from this strong branching e.g. $(x_1, x_2, \ldots, x_\gamma)$
3. For each pair of variables in $x_1, x_2, \ldots, x_\gamma$ do δ dual simplex iterations on each of the four possible grandchildren

---

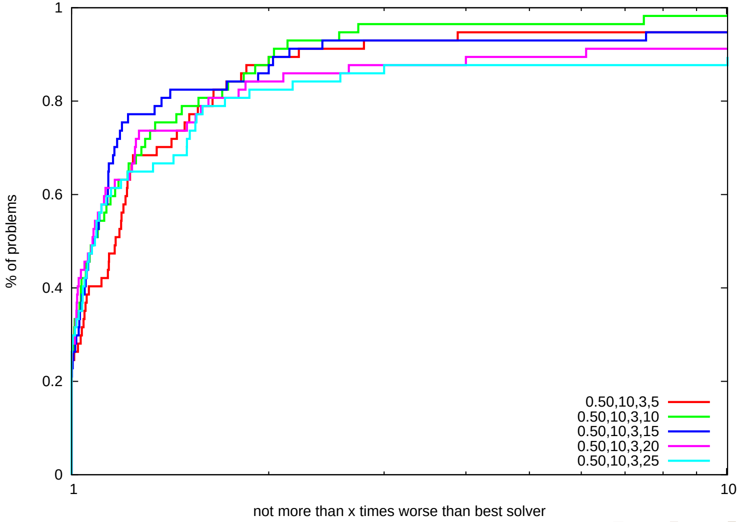
The $.064 Question

What Are Reasonable Values for $\alpha, \beta, \gamma, \delta$?

$\alpha = 0.5, \beta = 10, \gamma = ??, \delta = 10$



$\alpha = 0.5, \beta = 10, \gamma = 3, \delta = ??$



The Final Verdict

Reasonable Parameters

$$\alpha_{\mathsf{LA}} = 0.5, \beta_{\mathsf{LA}} = 10, \gamma = 3, \delta = 15$$

The Final Verdict

Reasonable Parameters

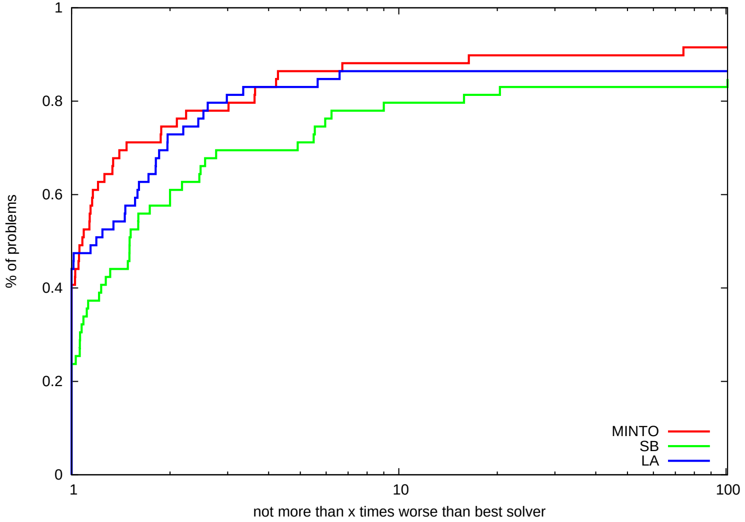
$$\alpha_{\mathsf{LA}} = 0.5, \beta_{\mathsf{LA}} = 10, \gamma = 3, \delta = 15$$

- To see if "lookahead" really makes any difference, we should see for a fixed number of pivots if it is significantly better than strong branching
- Compare SB(α_1, β_1) against strategies LA$(\alpha_2, \beta_2, \gamma, \delta)$, for parameter values such that

$$2|C_1|\beta_1 = 2|C_2|\beta_2 + 4\frac{\gamma(\gamma-1)\delta}{2}.$$

- Also compare to MINTO

Ta Da!!!!!!!!!


1
0.8
0.6
0.4
0.2
0
% of problems
1
10
100
not more than x times worse than best solver
MINTO
SB
LA

The End

pgflastimage

Conclusions

- There is often some useful information you can get big digging more than one level deep
- It will take some work to make it a "default" branching method
- For very hard problems, maybe it will be worth it.

There's Hard Work To Do!

pgflastimage

- Really use implications found. (Add to conflict graph)
    - "Automatic" triggering of further implications
    - Reduced Cost Fixing
    - Stronger Cuts
- A complete "mipping" of the branching decision.
- Better ranking mechanism for strong branching candidates (don't use fractionality)
- Introduce ideas from Reliability branching: Try the whole set $\mathcal{F}$ until the "winner" hasn't changed for η trials.
- Insert your own ideas here...