An abstract model for branching and its application to Mixed Integer Programming

Pierre Le Bodic
Joint work with George Nemhauser

School of Industrial and Systems Engineering
Georgia Institute of Technology

Grant FA9550-12-1-0151 of the Air Force Office of Scientific Research
Grant CCF-1415460 of the National Science Foundation

Discrete Maths Seminar, Monash University
August 1 2016

Linear Programming (LP)

$$\begin{aligned} z_{LP} = \max\ & c^t x \\ \text{s.t. } & Ax \le b \\ & x \in \mathbb{R}^n_+ \end{aligned}$$

LP is in **P**.

where $\begin{cases} A \text{ is a } m \times n \text{ matrix,} \\ c \text{ is a } n \text{ vector,} \\ b \text{ is a } m \text{ vector,} \end{cases}$ and all data are rational.

Linear Programming (LP) Integer Programming (IP)

$$\begin{aligned} z_{LP} = \max \ & c^t x \\ \text{s.t. } & Ax \le b \\ & x \in \mathbb{R}^n_+ \end{aligned}$$

LP is in **P**.

$$\begin{aligned} z_{IP} = \max \ & c^t x \\ \text{s.t. } & Ax \le b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

IP is **NP-hard**.

where $\begin{cases} A \text{ is a } m \times n \text{ matrix,} \\ c \text{ is a } n \text{ vector,} \\ b \text{ is a } m \text{ vector,} \end{cases}$ and all data are rational.

Linear Programming (LP) Integer Programming (IP)

$$\begin{aligned} z_{LP} = \max\ & c^t x \\ \text{s.t.}\ & Ax \le b \\ & x \in \mathbb{R}^n_+ \end{aligned}$$

LP is in **P**.

$$\begin{aligned} z_{IP} = \max\ & c^t x \\ \text{s.t.}\ & Ax \le b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

IP is **NP-hard**.

$$\boxed{z_{LP} \ge z_{IP}}$$

where $\begin{cases} A \text{ is a } m \times n \text{ matrix,} \\ c \text{ is a } n \text{ vector,} \\ b \text{ is a } m \text{ vector,} \end{cases}$ and all data are rational.

Linear Programming (LP) Integer Programming (IP)

$$z_{LP} = \max \; c^t x \quad \text{s.t. } Ax \leq b, \; x \in \mathbb{R}^n_+$$

LP is in **P**.

$$z_{IP} = \max \; c^t x \quad \text{s.t. } Ax \leq b, \; x \in \mathbb{Z}^n_+$$

IP is **NP-hard**.

$$\boxed{z_{LP} \geq z_{IP}}$$

$$\boxed{x \in Z^n_+ \text{ optimal for } LP \Rightarrow z_{IP} = z_{LP}}$$

where $\begin{cases} A \text{ is a } m \times n \text{ matrix,} \\ c \text{ is a } n \text{ vector,} \\ b \text{ is a } m \text{ vector,} \end{cases}$ and all data are rational.

Example: an IP formulation for edge coloring

Given $G = (V, E)$, let $C = \{1, \ldots, \Delta + 1\}$ be the set of possible colors, and

- Variable $c^i = 1$ iff color $i \in C$ is used,
- Variable $x_e^i = 1$ iff color $i \in C$ is assigned to edge $e \in E$.

$$
\begin{array}{llllr}
\min_{x,c} & \sum_{i \in C} c^i & & & (1) \\
& \sum_{i \in C} x_e^i = 1 & \forall e \in E & & (2) \\
& \sum_{u \in V, e = uv} x_e^i \leq c^i & \forall v \in V, & \forall i \in C & (3) \\
& c^i \in \{0, 1\} & & \forall i \in C & (4) \\
& x_e^i \in \{0, 1\} & \forall e \in E, & \forall i \in C & (5)
\end{array}
$$

- (1) minimizes the number of colors used.
- (2) ensures each edge is assigned a color.
- (3) enforces a proper coloring.
- (4) and (5) enforce integrality

(Mixed) Integer Programming solvers main components

Presolvers

- Simplify problem (e.g. eliminate redundancy)
- Tighten LP bound (e.g. change coefficients)

Primal heuristics

- Find a feasible solution (e.g. starting from LP or IP solution)

Cutting planes

- Tighten LP bound

Branch & Bound

- Implicit enumeration using primal and dual bounds to prune nodes

Outline of the B&B algorithm for MIP solving

Input: a MIP instance

Add the root node to the list of nodes to process
while the list of nodes is non empty **do**
Select* the node to process
Solve the node's LP
if the LP solution is integral **then**
Add the solution to the pool of solutions
else
if the node's LP bound is better than the primal bound **then**
Select** an *integer* variable x with a *fractional* value x_{LP}
Create two children where $x \leq \lfloor x_{LP} \rfloor$ or $x \geq \lceil x_{LP} \rceil$
Add the children to the list of nodes
end if
end if
end while
Output the best primal solution

* using a node selector

** using a *branching rule*

Discrete maths in two papers related to branching in MIP

anuscript

Operations Research Letters 43 (2015) 273–278

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

How important are branching decisions: Fooling MIP solvers

Pierre Le Bodic *, George L. Nemhauser

H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive, NW, Atlanta, GA 30332-0205, United States

ARTICLE INFO

Article history:
Received 14 April 2014
Received in revised form 24 February 2015
Accepted 3 March 2015
Available online 11 March 2015

Keywords:
Mixed integer programming solvers
Branch and bound
Tree size
Edge coloring
Chromatic index

ABSTRACT

We show the importance of selecting good branching variables by exhibiting a family of instances for which an optimal solution is both trivial to find and provably optimal by a fixed-size branch-and-bound tree, but for which state-of-the-art Mixed Integer Programming solvers need an increasing amount of resources. The instances encode the edge-coloring problem on a family of graphs containing a small subgraph requiring four colors, while the rest of the graph requires only three.

Published by Elsevier B.V.

1. Introduction

Mixed Integer Programming (MIP) solvers depend on branching rules to implicitly search the solution space. Numerous experimental results (see e.g. [2]) provide a good notion of their performances. However, little literature has been dedicated to theoretical results on MIP branching. One notable exception is Jeroslow's IP instance [9], for which a pure branch-and-bound algorithm provably requires a treesize that is exponential in the number of variables. By contrast, branching in satisfiability (SAT) solvers has been studied in a theoretical setting. Liberatore [12] has proven that choosing a branching candidate that minimizes the tree size is NP-hard. Ouyang [15] provides a family of instances of

size. Finally, we explain this behavior for SCIP, a state-of-the-art open-source MIP solver.

2. Instances

We build IP instances encoding the *chromatic index problem* on specific *input graphs* using a simple *mathematical model.*

2.1. The chromatic index problem

Let G be a simple graph. A *proper edge coloring* (we suppose all colorings are proper throughout the article) of G is such that no two adjacent edges are assigned the same color. The *chromatic index* χ'

Noname manuscript No.
(will be inserted by the editor)

An Abstract Model for Branching and its Application to Mixed Integer Programming

Pierre Le Bodic · George Nemhauser

the date of receipt and acceptance should be inserted later

Abstract The selection of branching variables is a key component of branch-and-bound algorithms for solving Mixed-Integer Programming (MIP) problems since the quality of the selection procedure is likely to have a significant effect on the size of the enumeration tree. State-of-the-art procedures base the selection of variables on their "LP gains", which is the dual bound improvement obtained after branching on a variable. There are various ways of selecting variables depending on their LP gains. However, all methods are evaluated empirically. In this paper we present a theoretical model for the selection of branching variables. It is based upon an abstraction of MIPs to a simpler setting in which it is possible to analytically evaluate the dual bound improvement of choosing a given variable. We then discuss how the analytical results can be used to choose branching variables for MIPs, and we give experimental results that demonstrate the effectiveness of the method on MIPLIB 2010 "tree" instances where we achieve a 5% geometric average time and node improvement, over the default rule of SCIP, a state-of-the-art MIP solver.

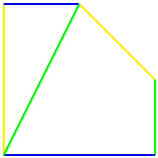
Keywords Branch and Bound, Abstract Model, Mixed Integer Programming, Computational Complexity, Algorithm Analysis

Fooling MIP solvers

Find a family of MIP instances for which:

- There exists a *small* Branch & Bound tree
- MIP solvers produce *big* Branch & Bound trees

Edge coloring problem

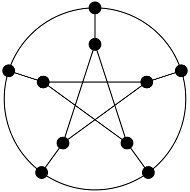


Without knowing Vizing, MIP solvers will still immediately find that they need

- At least Δ colors
- At most $\Delta + 1$ colors

Then they have to use branch-and-bound to decide between Δ and $\Delta + 1$.

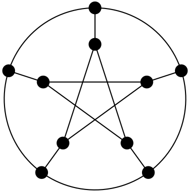
Petersen graph



- 10 vertices
- 15 edges
- degree 3
- chromatic index $\chi' = 4$

Good, but we want bigger graphs! And we can't add edges!

Petersen graph



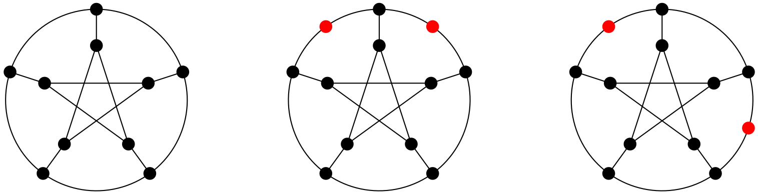
- 10 vertices
- 15 edges
- degree 3
- chromatic index $\chi' = 4$

Good, but we want bigger graphs! And we can't add edges!

Find a family of snarks that

- has arbitrarily large graphs
- is easily constructible

Modifying the Petersen graph

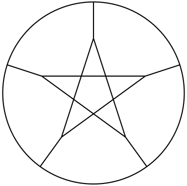


(a) Petersen graph
$\chi' = 4$

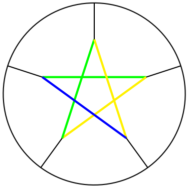
(b) Graph P_1
$\chi' = 4$

(c) Graph P_2
$\chi' = 3$

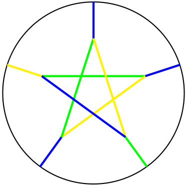
Proof



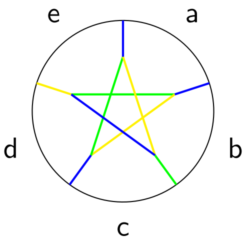
Proof



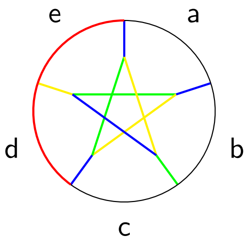
Proof



Proof

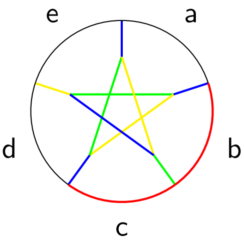


Proof



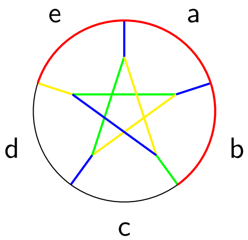
Based *only* on the coloring of the inside edges,
the "path" {d,e} cannot be blue or yellow

Proof



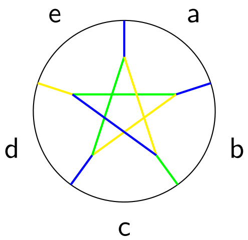
Based *only* on the coloring of the inside edges,
the “path” {d,e} cannot be blue or yellow
the “path” {b,c} cannot be blue or green

Proof



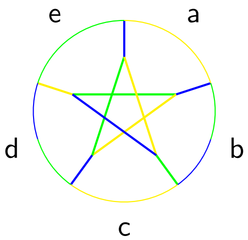
Based *only* on the coloring of the inside edges,
the “path” $\{$d,e$\}$ cannot be blue or yellow
the “path” $\{$b,c$\}$ cannot be blue or green
the “path” $\{$a,b,e$\}$ cannot be blue

Proof



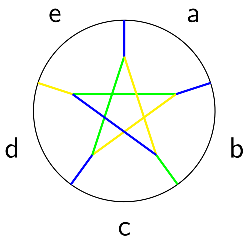
Based *only* on the coloring of the inside edges,
the "path" {d,e} cannot be blue or yellow
the "path" {b,c} cannot be blue or green
the "path" {a,b,e} cannot be blue
We want to *split* these "paths" by adding two vertices!

Proof



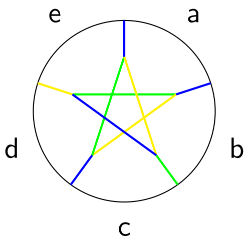
Based *only* on the coloring of the inside edges,
the "path" {d,e} cannot be blue or yellow
the "path" {b,c} cannot be blue or green
the "path" {a,b,e} cannot be blue
We want to *split* these "paths" by adding two vertices!
Split two edges: {b,d} or {b,e} or {c,e} $\rightarrow$ non-adjacent edges!

Proof



Based *only* on the coloring of the inside edges,
the "path" $\{$d,e$\}$ cannot be blue or yellow
the "path" $\{$b,c$\}$ cannot be blue or green
the "path" $\{$a,b,e$\}$ cannot be blue
We want to *split* these "paths" by adding two vertices!
Split two edges: $\{$b,d$\}$ or $\{$b,e$\}$ or $\{$c,e$\}$ $\rightarrow$ non-adjacent edges!
P_2 can be colored using three colors but P_1 cannot!

Proof



Based *only* on the coloring of the inside edges,
the "path" {d,e} cannot be blue or yellow
the "path" {b,c} cannot be blue or green
the "path" {a,b,e} cannot be blue
We want to *split* these "paths" by adding two vertices!
Split two edges: {b,d} or {b,e} or {c,e} $\rightarrow$ non-adjacent edges!
P_2 can be colored using three colors but P_1 cannot!

Input graphs

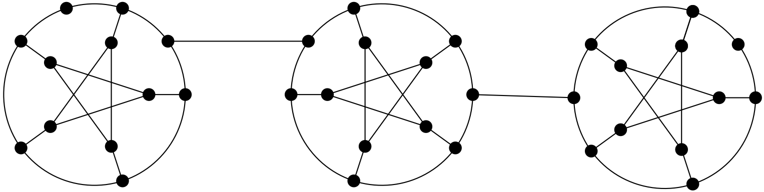


Figure: the graph G_3

$G_k = P_1 + (k-1)P_2$
G_k has $\Delta = 3$ and $\chi' = 4$.

A fixed-size Branch & Bound tree

Theorem

Given an optimal solution, there is a fixed-size Branch & Bound tree for any $k \geq 1$.

Proof

- Solve instance $I_1 \rightarrow$ Branch & Bound tree T_1
- Solve I_k by following T_1 in the Branch & Bound tree T_k
- Note that the global dual bound of T_1 is 4
- All constraints of I_1 are contained in or implied by I_k, thus the global dual bound of the tree T_k is 4

Experimental results

Size (k)	CPLEX s	CPLEX n	CPLEX t	GUROBI s	GUROBI n	GUROBI t	SCIP s	SCIP n	SCIP t
1	10	11	0	10	21	0	10	12	0
2	10	15	0	10	23	0	10	19	1
4	10	38	0	10	30	1	10	41	4
8	10	59	0	10	50	3	10	79	10
16	9	302	3	10	84	15	10	263	23
32	7	213	11	10	175	47	10	419	48
64	9	50	26	10	1921	424	9	1328	178
128	8	276	79	7	1470	1098	10	6542	808
256	6	1366	564	7	699	4182	8	6225	2041
512	2	3265	1700	7	198	3586	6	6125	6347
1024	2	1509	5501	3	112	16943	0	-	-

Number of instances solved (s), and, for the instances solved, the geometric means of the number of nodes (n) and time in seconds (t)

An abstract model for branching and its application to Mixed Integer Programming

Noname manuscript No.
(will be inserted by the editor)

An Abstract Model for Branching and its Application to Mixed Integer Programming

Pierre Le Bodic · George Nemhauser

the date of receipt and acceptance should be inserted later

Abstract The selection of branching variables is a key component of branch-and-bound algorithms for solving Mixed-Integer Programming (MIP) problems since the quality of the selection procedure is likely to have a significant

State-of-the-art branching rule in MIP solvers

At a given node, a *branching rule* picks the variable to branch on.

The state-of-the-art branching rule is a *hybrid* of two branching rules that aim at improving the dual bound.

State-of-the-art branching rule in MIP solvers

At a given node, a *branching rule* picks the variable to branch on.
The state-of-the-art branching rule is a *hybrid* of two branching rules that aim at improving the dual bound.

Strong branching

For all fractional variables x, *strong branching* computes the LP values at the children that would be created by branching on x.
Example: $x_1 = 0.2, x_2 = 0.5$ in the LP relaxation.

State-of-the-art branching rule in MIP solvers

At a given node, a *branching rule* picks the variable to branch on.

The state-of-the-art branching rule is a *hybrid* of two branching rules that aim at improving the dual bound.

Strong branching

For all fractional variables x, *strong branching* computes the LP values at the children that would be created by branching on x.

Example: $x_1 = 0.2, x_2 = 0.5$ in the LP relaxation.

Pseudocost branching

For all fractional variables x, *pseudocost branching* imitates strong branching using historical information provided by strong branching and (actual) branching.

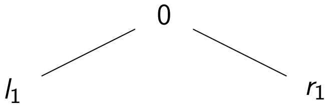
Branch & Bound abstract model for MIP solving

Variable = pair (l, r) of two > 0 integers with $l \leq r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node

[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

Branch & Bound abstract model for MIP solving

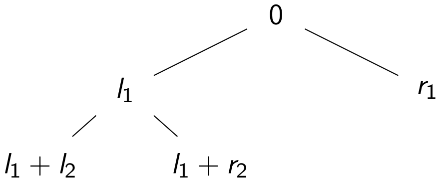
Variable = pair (l, r) of two > 0 integers with $l \leq r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node



[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

Branch & Bound abstract model for MIP solving

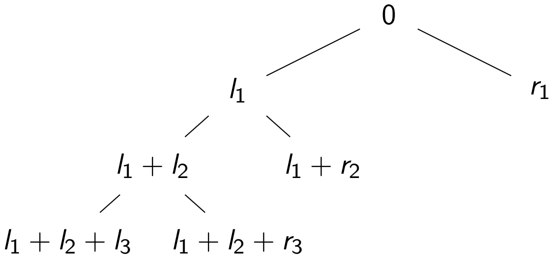
Variable = pair (l, r) of two > 0 integers with $l \leq r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node



[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

Branch & Bound abstract model for MIP solving

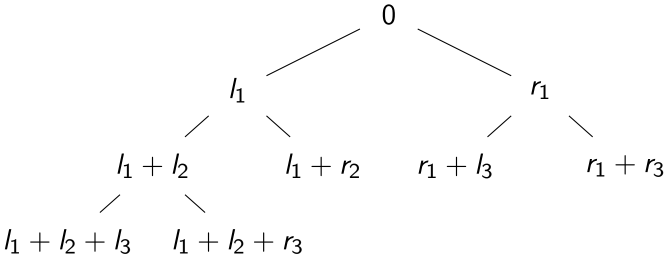
Variable = pair (l, r) of two > 0 integers with $l \leq r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node



[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

Branch & Bound abstract model for MIP solving

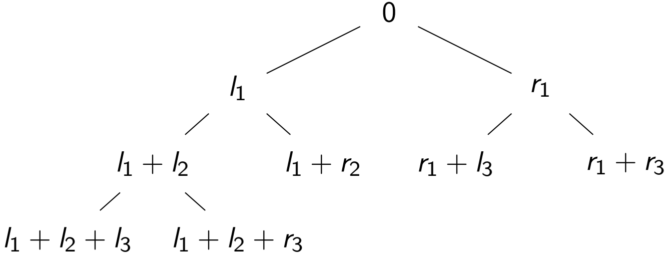
Variable = pair (l, r) of two > 0 integers with $l \leq r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node



[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

Branch & Bound abstract model for MIP solving

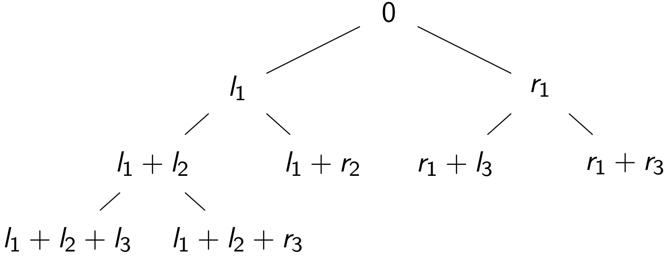
Variable = pair (l, r) of two > 0 integers with $l \leq r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node
Gap closed by the B&B tree = minimum gap at a leaf



[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

Branch & Bound abstract model for MIP solving

Variable = pair (l, r) of two > 0 integers with $l \le r$
B&B tree = binary tree with a variable at each inner node
(Absolute) Gap[1] closed at a node = value at that node
Gap closed by the B&B tree = minimum gap at a leaf



Tree-size = 9

[1] In MIP solvers, the (absolute) gap is the difference between the primal and the dual bound.

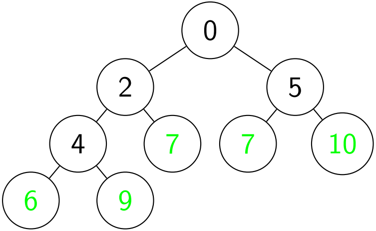
Single Variable Branching (SVB)

Input: one variable (l, r), an integer G and an integer k, all positive.
Question: Is the size of the Branch & Bound tree that closes the gap G, repeatedly using the given variable, at most k?

Single Variable Branching (SVB)

Input: one variable (l, r), an integer G and an integer k, all positive.
Question: Is the size of the Branch & Bound tree that closes the gap G, repeatedly using the given variable, at most k?

Example: variable $(2, 5)$ and gap $G = 6$.



Treesize = 9

Motivation: state-of-the-art scoring functions

At a given node, we have to branch on a variable given mutliple variables x with gains (l_x, r_x). $\Rightarrow$ *scoring* functions

Motivation: state-of-the-art scoring functions

At a given node, we have to branch on a variable given mutliple variables x with gains (l_x, r_x). $\Rightarrow$ *scoring* functions

Linear function:

$$(1 - \mu) \times l_x + \mu \times r_x \qquad \left(\mu = \frac{1}{6}\right)$$

Motivation: state-of-the-art scoring functions

At a given node, we have to branch on a variable given mutliple variables x with gains (l_x, r_x). $\Rightarrow$ *scoring* functions

Linear function:

$$(1-\mu) \times l_x + \mu \times r_x \qquad \left(\mu = \frac{1}{6}\right)$$

Product function:

$$l_x \times r_x$$

Motivation: state-of-the-art scoring functions

At a given node, we have to branch on a variable given mutliple variables x with gains (l_x, r_x). $\Rightarrow$ *scoring* functions

Linear function:

$$(1-\mu) \times l_x + \mu \times r_x \qquad \left(\mu = \frac{1}{6}\right)$$

Product function:

$$l_x \times r_x$$

-34% nodes
-14% time

Motivation: state-of-the-art scoring functions

At a given node, we have to branch on a variable given mutliple variables x with gains (l_x, r_x). $\Rightarrow$ *scoring* functions

Linear function:

$$(1 - \mu) \times l_x + \mu \times r_x \qquad \left(\mu = \frac{1}{6}\right)$$

Product function:

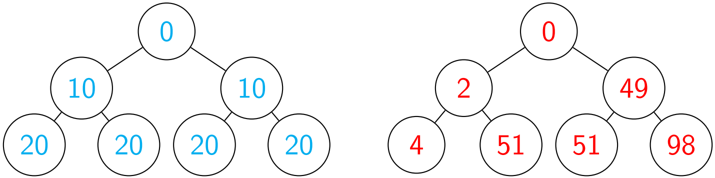
$$l_x \times r_x$$

-34% nodes
-14% time

Variables $(10, 10)$ and $(2, 50)$ have the same score for both functions!

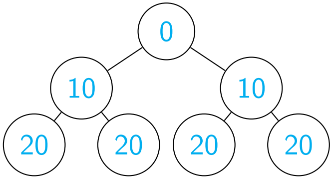
Motivation: variable $(10, 10)$ vs $(2, 49)$ for SVB

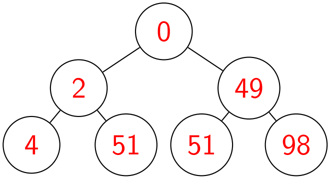
The linear and product functions both score $(10, 10)$ higher than $(2, 49)$.

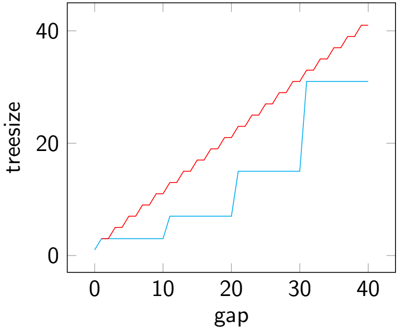


Motivation: variable $(10, 10)$ vs $(2, 49)$ for SVB

The linear and product functions both score $(10, 10)$ higher than $(2, 49)$.

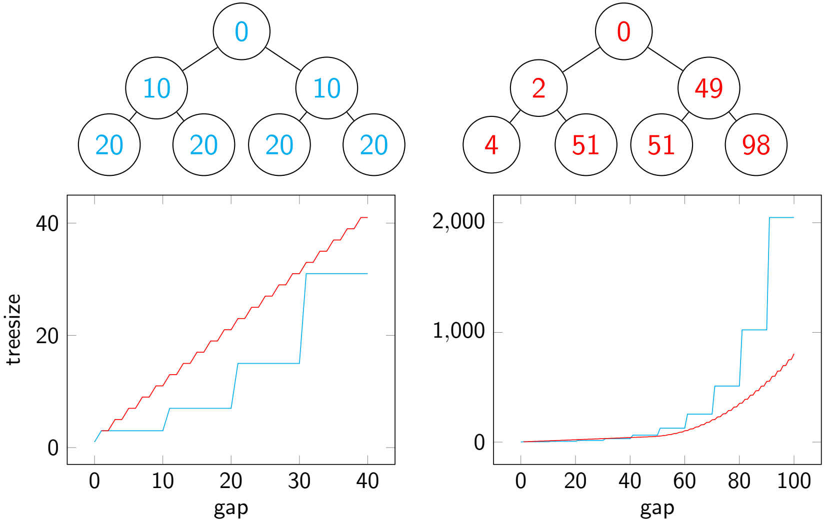






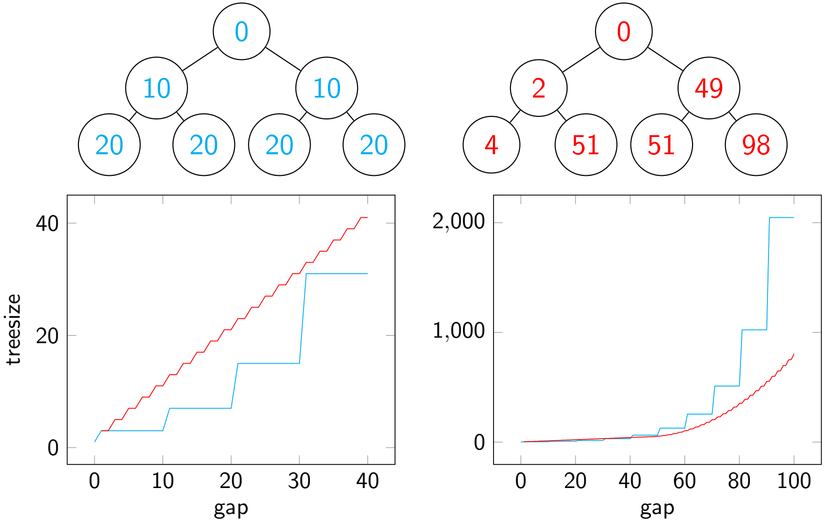
Motivation: variable $(10, 10)$ vs $(2, 49)$ for SVB

The linear and product functions both score $(10, 10)$ higher than $(2, 49)$.



Motivation: variable $(10, 10)$ vs $(2, 49)$ for SVB

The linear and product functions both score $(10, 10)$ higher than $(2, 49)$.



At gap $G = 1000$, the relative difference in treesize is 323 *millions*.

Complexity results for SVB

Notation

$t(G)$ = treesize to close G with (l, r)

Is $t(G) \leq k$?

Complexity results for SVB

Notation

$t(G)$ = treesize to close G with (l, r)

Is $t(G) \leq k$?

Recursion

$$t(G) = \begin{cases} 1 & \text{if } G \leq 0 \\ 1 + t(G - l) + t(G - r) & \text{otherwise} \end{cases}$$

Pseudo-polynomial!

Complexity results for SVB

Notation

$t(G)$ = treesize to close G with (l, r)
Is $t(G) \leq k$?

Recursion

$$t(G) = \begin{cases} 1 & \text{if } G \leq 0 \\ 1 + t(G-l) + t(G-r) & \text{otherwise} \end{cases}$$

Pseudo-polynomial!

Closed-form formula (CFF)

$$t(G) = 1 + 2 \times \sum_{h=1}^{\lceil \frac{G}{r} \rceil} \binom{h + \lceil \frac{G-(h-1)\times r}{l} \rceil - 1}{h}$$

Proof: group leaves that are reached by "turning" right h times together.

$O(\log^2(k))$ is polynomial, but still big in practice!

Asymptotic case results for SVB

- For variable $(1, 2)$, the growth rate $\frac{t(G+1)}{t(G)}$ of the tree tends to $\frac{1+\sqrt{5}}{2}$

Asymptotic case results for SVB

- For variable $(1,2)$, the growth rate $\frac{t(G+1)}{t(G)}$ of the tree tends to $\frac{1+\sqrt{5}}{2}$
- The sequence $\frac{t(G+1)}{t(G)}$ does not converge for all variables. We use:

$$\varphi = \lim_{G\to\infty} \sqrt[l]{\frac{t(G+l)}{t(G)}}$$

φ is the unique > 1 root of the polynomial $p(x) = x^r - x^{r-l} - 1$

Asymptotic case results for SVB

- For variable $(1, 2)$, the growth rate $\frac{t(G+1)}{t(G)}$ of the tree tends to $\frac{1+\sqrt{5}}{2}$
- The sequence $\frac{t(G+1)}{t(G)}$ does not converge for all variables. We use:

$$\varphi = \lim_{G \to \infty} \sqrt[l]{\frac{t(G+l)}{t(G)}}$$

  φ is the unique > 1 root of the polynomial $p(x) = x^r - x^{r-l} - 1$
- We have the bounds $\sqrt[r]{2} \leq \varphi \leq \sqrt[l]{2}$

Asymptotic case results for SVB

- For variable $(1, 2)$, the growth rate $\frac{t(G+1)}{t(G)}$ of the tree tends to $\frac{1+\sqrt{5}}{2}$
- The sequence $\frac{t(G+1)}{t(G)}$ does not converge for all variables. We use:

$$\varphi = \lim_{G \to \infty} \sqrt[l]{\frac{t(G+l)}{t(G)}}$$

  φ is the unique > 1 root of the polynomial $p(x) = x^r - x^{r-l} - 1$
- We have the bounds $\sqrt[r]{2} \leq \varphi \leq \sqrt[l]{2}$
- A numerical approximation of φ^r is given by the fixed-point iteration

$$f(x) = 1 + \frac{1}{x^{\frac{l}{r}} - 1}$$

Asymptotic case results for SVB

- For variable $(1,2)$, the growth rate $\frac{t(G+1)}{t(G)}$ of the tree tends to $\frac{1+\sqrt{5}}{2}$
- The sequence $\frac{t(G+1)}{t(G)}$ does not converge for all variables. We use:

$$\varphi = \lim_{G\to\infty} \sqrt[l]{\frac{t(G+l)}{t(G)}}$$

  φ is the unique > 1 root of the polynomial $p(x) = x^r - x^{r-l} - 1$
- We have the bounds $\sqrt[r]{2} \leq \varphi \leq \sqrt[l]{2}$
- A numerical approximation of φ^r is given by the fixed-point iteration

$$f(x) = 1 + \frac{1}{x^{\frac{l}{r}} - 1}$$

- For $G \geq F$, we have

$$t(G) \approx \varphi^{G-F} t(F)$$

Asymptotic case results for SVB

- For variable $(1,2)$, the growth rate $\frac{t(G+1)}{t(G)}$ of the tree tends to $\frac{1+\sqrt{5}}{2}$
- The sequence $\frac{t(G+1)}{t(G)}$ does not converge for all variables. We use:

$$\varphi = \lim_{G\to\infty} \sqrt[l]{\frac{t(G+l)}{t(G)}}$$

  φ is the unique >1 root of the polynomial $p(x) = x^r - x^{r-l} - 1$
- We have the bounds $\sqrt[r]{2} \leq \varphi \leq \sqrt[l]{2}$
- A numerical approximation of φ^r is given by the fixed-point iteration

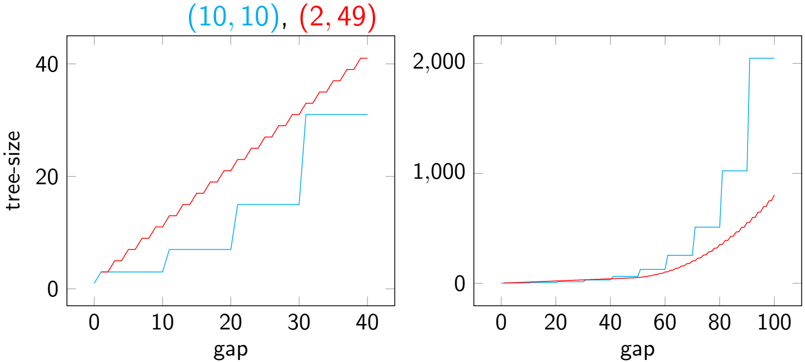
$$f(x) = 1 + \frac{1}{x^{\frac{l}{r}} - 1}$$

- For $G \geq F$, we have

$$t(G) \approx \varphi^{G-F} t(F)$$

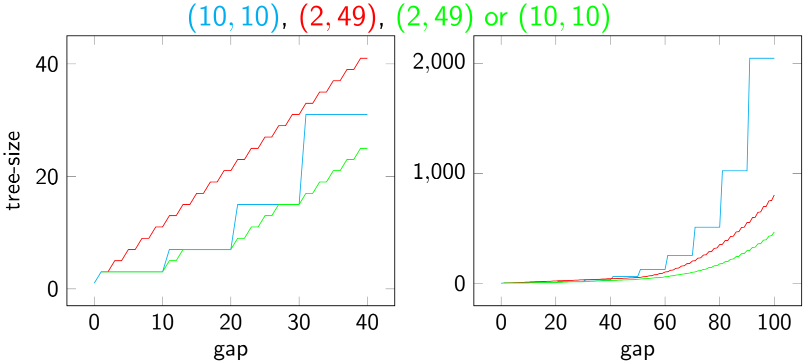
- Given two variables x and y and a "large" G, $\varphi_x < \varphi_y$ implies that branching on x leads to a smaller treesize.

Single Variable Branching



$$\varphi_{(10,10)} \approx 1.071, \quad \varphi_{(2,49)} \approx 1.049$$

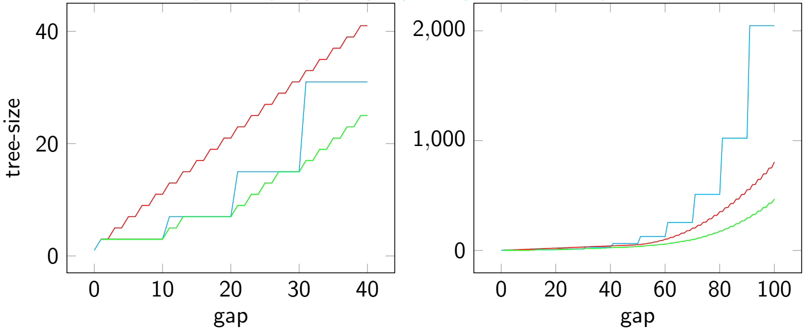
Multiple Variable Branching



$$\varphi_{(10,10)} \approx 1.071, \quad \varphi_{(2,49)} \approx 1.049$$

Multiple Variable Branching

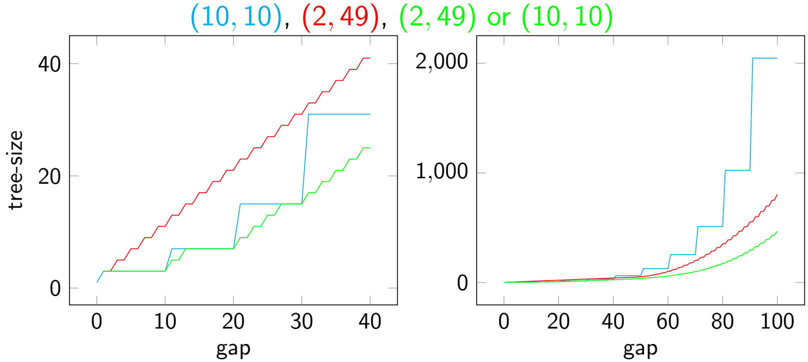
$(10, 10)$, $(2, 49)$, $(2, 49)$ or $(10, 10)$



$$\varphi_{(10,10)} \approx 1.071, \quad \varphi_{(2,49)} \approx 1.049$$

- For a gap of $G = 1000$, only a relative difference of 1.798 between red and green treesizes.

Multiple Variable Branching



$$\varphi_{(10,10)} \approx 1.071, \quad \varphi_{(2,49)} \approx 1.049$$

- For a gap of $G = 1000$, only a relative difference of 1.798 between red and green treesizes.
- Variable $(2, 49)$ is branched on at every node where the gap left to close is at least 31.

Multiple Variable Branching (MVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, using each variable as many times as needed?

$$t(G) = \begin{cases} 1 & \text{if } G \leq 0 \\ 1 + \min\limits_{1 \leq i \leq n} \left(t(G - l_i) + t(G - r_i) \right) & \text{otherwise} \end{cases}$$

We do not know if there exists a poly-time algorithm!

Multiple Variable Branching (MVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, using each variable as many times as needed?

$$t(G) = \begin{cases} 1 & \text{if } G \leq 0 \\ 1 + \min\limits_{1 \leq i \leq n} \left(t(G - l_i) + t(G - r_i)\right) & \text{otherwise} \end{cases}$$

We do not know if there exists a poly-time algorithm!
We define:

$$\varphi = \lim_{G \to \infty} \sqrt[z]{\frac{t(G+z)}{t(G)}}$$

where z is the least common multiple of all l_i and r_i.

Multiple Variable Branching (MVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, using each variable as many times as needed?

$$t(G) = \begin{cases} 1 & \text{if } G \leq 0 \\ 1 + \min_{1 \leq i \leq n} \left(t(G - l_i) + t(G - r_i)\right) & \text{otherwise} \end{cases}$$

We do not know if there exists a poly-time algorithm!
We define:

$$\varphi = \lim_{G \to \infty} \sqrt[z]{\frac{t(G+z)}{t(G)}}$$

where z is the least common multiple of all l_i and r_i.

$$\varphi = \min_i \varphi_i$$

where φ_i is the SVB ratio of variable i.

Multiple Variable Branching (MVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, using each variable as many times as needed?

$$t(G) = \begin{cases} 1 & \text{if } G \leq 0 \\ 1 + \min_{1 \leq i \leq n} \left(t(G - l_i) + t(G - r_i) \right) & \text{otherwise} \end{cases}$$

We do not know if there exists a poly-time algorithm!
We define:

$$\varphi = \lim_{G \to \infty} \sqrt[z]{\frac{t(G+z)}{t(G)}}$$

where z is the least common multiple of all l_i and r_i.

$$\varphi = \min_i \varphi_i$$

where φ_i is the SVB ratio of variable i.
For $G \geq F$, we have

$$t(G) \approx \varphi^{G-F} t(F)$$

General Variable Branching (GVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, branching on each variable i **at most once** on each path from the root to a leaf?

General Variable Branching (GVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, branching on each variable i **at most once** on each path from the root to a leaf?

GVB models B&B in MIP solvers under two hypotheses

- In GVB the primal gap is considered to be 0
- LP gains are fixed and known

General Variable Branching (GVB)

Input: n variables $(l_i, r_i), i = 1, \ldots, n$, an integer $G > 0$, an integer $k > 0$.
Question: Is there a Branch & Bound tree with at most k nodes that closes the gap G, branching on each variable i **at most once** on each path from the root to a leaf?

GVB models B&B in MIP solvers under two hypotheses

- In GVB the primal gap is considered to be 0
- LP gains are fixed and known

Complexity of GVB

- is #P-hard (#Knapsack reduction)
- Under the conjecture that the Polynomial Hierarchy is proper, this implies that GVB is not in NP or co-NP

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$
Variables $(100, 105), (100, 103), (100, 102)$

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$
Variables $(100, 105), (100, 103), (100, 102), (100, 100)$

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$
Variables $(100, 105), (100, 103), (100, 102), (100, 100)$ and $G = 305$

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

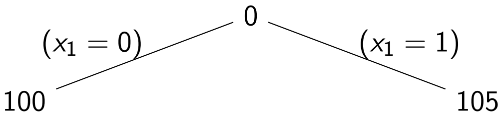
"Big" number $= 100 \geq 5 + 3 + 2$
Variables $(100, 105), (100, 103), (100, 102), (100, 100)$ and $G = 305$

0

#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$
Variables $(100, 105), (100, 103), (100, 102), (100, 100)$ and $G = 305$

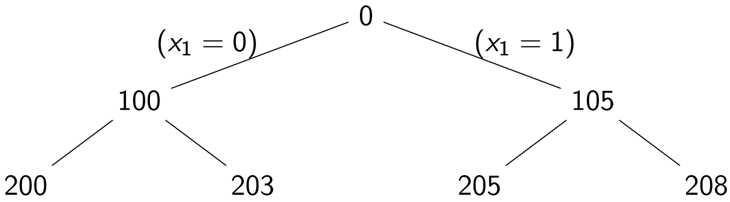


#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$

Variables $(100, 105), (100, 103), (100, 102), (100, 100)$ and $G = 305$

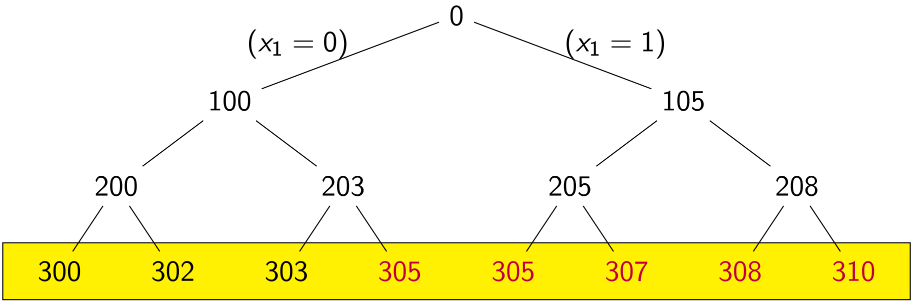


#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$

Variables $(100, 105), (100, 103), (100, 102), (100, 100)$ and $G = 305$

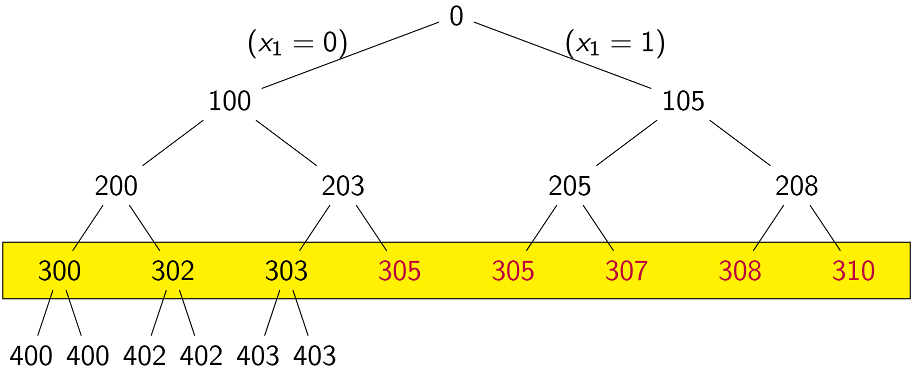


#P-hardness proof by example

$$5x_1 + 3x_2 + 2x_3 \geq 5$$

"Big" number $= 100 \geq 5 + 3 + 2$

Variables $(100, 105), (100, 103), (100, 102), (100, 100)$ and $G = 305$



The ratio (φ) scoring function

Simple version

Branch on the variable i with smallest φ_i (root of $p(x) = x^{r_i} - x^{r-l_i} - 1$)

The ratio (φ) scoring function

Simple version

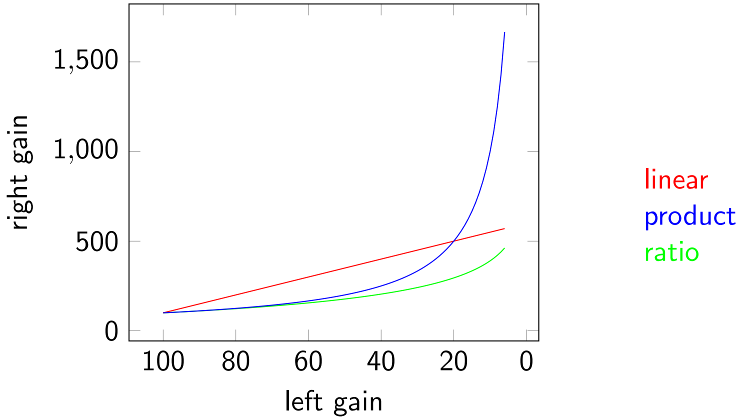
Branch on the variable i with smallest φ_i (root of $p(x) = x^{r_i} - x^{r-l_i} - 1$)

Faster version

- Filter out variables with “dominated” gains
- Compute φ^* as the ratio of the best variable according to product
- For each variable i, test $p_i(\varphi^*) > 0$
- If true, compute the root φ_i of p_i and update $\varphi^* = \varphi_i$

Note: the only parameter is the maximum number of iterations to approximate φ_i.

The ratio (φ) scoring function



Right gains as a function of the left gain such that the score is constant

Numerical results: summary

General improvements in time and number of nodes

- $\sim 5\%$ in B&B simulations for large gaps
- $\sim 2\%$ on MIPLIB "benchmark" instances
- $\sim 5\%$ on MIPLIB "tree" test set

Why read the paper?

- One of the first theoretical studies of B&B
- Open complexity and approximation problems
- Many possible extensions
- Theory that yields direct experimental improvements

http://arxiv.org/abs/1511.01818
To appear in *Mathematical Programming series A*.