Learning to Branch in Mixed Integer Programming

Elias Khalil

School of Computational Science & Engineering
Georgia Institute of Technology

AAAI 2016

Joint work with Pierre Le Bodic, Le Song, George Nemhauser and Bistra Dilkina
Goal: solve Mixed Integer Linear Programs (MIP) “faster”

Branch-and-Bound (B&B): a general framework for solving MIPs

B&B components: presolve, cutting planes, primal heuristics, **branching** (this talk)

**Figure**: A B&B tree. A node is a sub-problem, an edge is an additional constraint.
Motivation

- Branching on the “right” variables can have a dramatic impact on size of B&B tree (i.e. number of nodes)

\[ \bar{z} = 4 \]

LP relaxation value

▶ Branching on \( x_j \) prunes off the child sub-trees
Motivation

- Branching on the “right” variables can have a dramatic impact on size of B&B tree (i.e. number of nodes)
- Example: suppose we are minimizing and the incumbent value is 5
Motivation

- Branching on the “right” variables can have a dramatic impact on size of B&B tree (i.e. number of nodes)
- Example: suppose we are minimizing and the incumbent value is 5

\[
\begin{align*}
\text{LP relaxation value} & \quad \hat{z} = 4 \\
\hat{x}_j & \leq \lfloor \hat{x}_j \rfloor \\
\hat{x}_j & \geq \lceil \hat{x}_j \rceil \\
\hat{x}_j^- & = 6 \\
\hat{x}_j^+ & = 7
\end{align*}
\]

- Branching on \( x_j \) prunes off the child sub-trees
Quality of a Branch

Measuring the quality of a branch:

\[ x_j \leq \lceil \tilde{x}_j \rceil \quad x_j \geq \lfloor \tilde{x}_j \rfloor \]

- Change in objective value: \( \Delta_j^- = \tilde{z}_j - \tilde{z} \) and \( \Delta_j^+ = \tilde{z}_j^+ - \tilde{z} \)
- Quality of the branch:

\[
\text{score}(\Delta_j^-, \Delta_j^+) = \Delta_j^- \cdot \Delta_j^+
\]

- Higher score is better
Two Classes of Branching Strategies

- Strong Branching (SB)

For some of the fractional integer variables at a node, simulate their branching quality by solving LPs, branch on var. with best product score.

Pseudocost Branching (PC)

Maintain some historical averages of the increase in obj. value, one for each of upwards and downwards branchings; combine by product. Imitate SB without solving many LP sub-problems.

Elias Khalil (Georgia Tech)
Two Classes of Branching Strategies

- **Strong Branching (SB)**
  - For some of the fractional integer variables at a node, *simulate* their branching quality by solving LPs, branch on var. with best product score

- **Pseudocost Branching (PC)**
  - Maintain some historical averages of the increase in obj. value, one for each of upwards and downwards branchings; combine by product.
  - Imitate SB without solving many LP sub-problems.
Two Classes of Branching Strategies

- **Strong Branching (SB)**
  - For some of the fractional integer variables at a node, *simulate* their branching quality by solving LPs, branch on var. with best product score

- **Pseudocost Branching (PC)**
Two Classes of Branching Strategies

- **Strong Branching (SB)**
  - For some of the fractional integer variables at a node, *simulate* their branching quality by solving LPs, branch on var. with best product score

- **Pseudocost Branching (PC)**
  - Maintain some historical averages of the increase in obj. value, one for each of upwards and downwards branchings; combine by product.
Two Classes of Branching Strategies

- **Strong Branching (SB)**
  - For some of the fractional integer variables at a node, *simulate* their branching quality by solving LPs, branch on var. with best product score

- **Pseudocost Branching (PC)**
  - Maintain some historical averages of the increase in obj. value, one for each of upwards and downwards branchings; combine by product.
  - Imitate SB without solving many LP sub-problems.
Two Classes of Branching Strategies

- **Strong Branching (SB)**
  - For some of the fractional integer variables at a node, *simulate* their branching quality by solving LPs, branch on var. with best product score.

- **Pseudocost Branching (PC)**
  - Maintain some historical averages of the increase in obj. value, one for each of upwards and downwards branchings; combine by product.
  - Imitate SB without solving many LP sub-problems.
Contribution

- This talk: *can we get the best of both worlds (SB and PC)*?
  - small number of nodes
  - small time
  - as little manual tuning as possible
This talk: *can we get the best of both worlds (SB and PC)?*
- small number of nodes
- small time
- as little manual tuning as possible
Contribution

- This talk: *can we get the best of both worlds (SB and PC)?*
  - small number of nodes
  - small time
  - as little manual tuning as possible

Proposed Method

A Machine Learning (ML) framework for branching that imitates SB well, at a fraction of the computation cost
Regression:
- Tempted to set up learning task as fitting a regression model to estimate SB scores directly.
Regression:

- Tempted to set up learning task as fitting a regression model to estimate SB scores directly

Key observation:

- SB scores are only used to select the single best variable
Regression:
  ▶ Tempted to set up learning task as fitting a regression model to estimate SB scores directly

Key observation:
  ▶ SB scores are only used to select the single best variable

Goal: learn to imitate the correct ranking of the candidate variables at each node
  ▶ Collect training data on SB ranking & fit a ranking model
Learning of Branching Strategies

- **On-the-fly**: no upfront offline learning required
- **Instance-specific**: different ML model for each instance
- **No lost work**: data collection is part of the search tree

**Parameters**:
- $\theta$: maximum number of SB nodes
- $\kappa$: maximum size of variable candidate set at each node
On-the-fly Learning of Branching Strategies

**data collection**: run SB for $\theta$ nodes, and use SB scores as *labels* for variables within each node; compute *features* describing each variable (dataset $D$)
On-the-fly Learning of Branching Strategies

data collection: run SB for $\theta$ nodes, and use SB scores as labels for variables within each node; compute features describing each variable (dataset $D$)

model learning: learn a model (function $f$) that ranks variables in $D$ s.t. good ones are ranked better than bad ones
On-the-fly Learning of Branching Strategies

MIP instance → data collection → model learning → ML-based branching

Parameters

0

θ

until termination

Nodes

**data collection**: run SB for θ nodes, and use SB scores as *labels* for variables within each node; compute *features* describing each variable (dataset $\mathcal{D}$)

**model learning**: learn a model (function $f$) that *ranks* variables in $\mathcal{D}$ s.t. good ones are ranked better than bad ones

**ML-based branching**: as of the $(\theta + 1)^{st}$ node, use learned function $f$ to rank variables based on their features
Data Collection

For the first $\theta$ nodes, $N_i \in \mathcal{N} = \{N_1, \ldots, N_\theta\}$, run SB on some fractional integer variables $C_i$, $|C_i| \leq \kappa$, and collect dataset $\mathcal{D}$ formed of:

- features describing each candidate variable at that node
- labels $y_{ij}$, based on SB scores, s.t. higher is better

Example dataset:

$\theta = 2, \kappa = 4, p = 4$ features

<table>
<thead>
<tr>
<th>feature 1</th>
<th>feature 2</th>
<th>feature 3</th>
<th>feature 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>$x_2$</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>$x_4$</td>
<td>1</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_6$</td>
<td>1</td>
<td>0.5</td>
<td>0.8</td>
</tr>
<tr>
<td>$N_2$</td>
<td>$x_1$</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>0.4</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Data Collection

For the first $\theta$ nodes, $N_i \in \mathcal{N} = \{N_1, \ldots, N_\theta\}$, run SB on some fractional integer variables $C_i$, $|C_i| \leq \kappa$, and collect dataset $\mathcal{D}$ formed of:

- **features** describing each candidate variable at that node
Data Collection

For the first $\theta$ nodes, $N_i \in \mathcal{N} = \{N_1, \ldots, N_{\theta}\}$, run SB on some fractional integer variables $C_i$, $|C_i| \leq \kappa$, and collect dataset $\mathcal{D}$ formed of:

- **features** describing each candidate variable at that node
- **labels** $y_j^i$, based on SB scores, s.t. higher is better

Example dataset:

\[
\begin{align*}
\theta &= 2, \kappa = 4, \ p = 4 \\
&
\begin{array}{cccc}
\text{feature 1} & \text{feature 2} & \text{feature 3} & \text{feature 4} \\
N_1 & 0 & 0.4 & 0.4 & 0.3 & 0.9 \\
N_2 & 1 & 0.2 & 0.9 & 0.4 & 0.4 \\
N_3 & 0 & 0.2 & 0.6 & 0.7 \\
N_4 & 1 & 0.5 & 0.8 & 0.4 \\
\end{array}
\end{align*}
\]
Data Collection

For the first $\theta$ nodes, $N_i \in \mathcal{N} = \{N_1, \ldots, N_\theta\}$, run SB on some fractional integer variables $C_i$, $|C_i| \leq \kappa$, and collect dataset $\mathcal{D}$ formed of:

- **features** describing each candidate variable at that node
- **labels** $y^i_j$, based on SB scores, s.t. higher is better

Example dataset: $\theta = 2$, $\kappa = 4$, $p = 4$ features

<table>
<thead>
<tr>
<th>$y^i_j$</th>
<th>feature 1</th>
<th>feature 2</th>
<th>feature 3</th>
<th>feature 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>0.4</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>$x_4$</td>
<td>1</td>
<td>0.3</td>
<td>0.9</td>
<td>0.4</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>$x_6$</td>
<td>1</td>
<td>0.5</td>
<td>0.8</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y^i_j$</th>
<th>feature 1</th>
<th>feature 2</th>
<th>feature 3</th>
<th>feature 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>0.5</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>0.4</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Features

- Feature map $\phi : \mathcal{X} \times \mathcal{N} \rightarrow [0, 1]^p$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$. 
  $\phi(x_j, N_i)$ describes variable $x_j$ at node $N_i$ with $p$ features.
Features

- Feature map $\phi : \mathcal{X} \times \mathcal{N} \rightarrow [0, 1]^p$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$. $\phi(x_j, N_i)$ describes variable $x_j$ at node $N_i$ with $p$ features.

- **Atomic features:**
Features

- Feature map $\phi : \mathcal{X} \times \mathcal{N} \rightarrow [0, 1]^p$, where $\mathcal{X} = \{x_1, ..., x_n\}$. $\phi(x_j, N_i)$ describes variable $x_j$ at node $N_i$ with $p$ features.

- **Atomic features:**
  - static & dynamic, taking into account the structure of the node subproblem
Feature map $\phi : \mathcal{X} \times \mathcal{N} \to [0, 1]^p$, where $\mathcal{X} = \{x_1, ..., x_n\}$.

$\phi(x_j, N_i)$ describes variable $x_j$ at node $N_i$ with $p$ features.

**Atomic features:**

- *static & dynamic*, taking into account the structure of the node subproblem
- *easy to compute*, with time complexity $O(nnz(A))$
Features

- Feature map $\phi : \mathcal{X} \times \mathcal{N} \rightarrow [0, 1]^p$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$. $\phi(x_j, N_i)$ describes variable $x_j$ at node $N_i$ with $p$ features.

- **Atomic features:**
  - *static & dynamic*, taking into account the structure of the node subproblem
  - *easy to compute*, with time complexity $O(\text{nnz}(A))$

- **Interaction features:**
Features

- Feature map \( \phi : \mathcal{X} \times \mathcal{N} \rightarrow [0, 1]^p \), where \( \mathcal{X} = \{x_1, ..., x_n\} \).
  \( \phi(x_j, N_i) \) describes variable \( x_j \) at node \( N_i \) with \( p \) features.

- **Atomic features:**
  - *static & dynamic*, taking into account the structure of the node subproblem
  - *easy to compute*, with time complexity \( O(\text{nnz}(A)) \)

- **Interaction features:**
  - create new features by multiplying each pair of atomic features.
Features

- Feature map $\phi : \mathcal{X} \times \mathcal{N} \rightarrow [0, 1]^p$, where $\mathcal{X} = \{x_1, ..., x_n\}$. $\phi(x_j, N_i)$ describes variable $x_j$ at node $N_i$ with $p$ features.

- **Atomic features:**
  - *static & dynamic*, taking into account the structure of the node subproblem
  - *easy to compute*, with time complexity $O(\text{nnz}(A))$

- **Interaction features:**
  - create new features by multiplying each pair of atomic features.

- Scale and normalize all features, *per node*, to $[0, 1]$. 
We currently use 72 atomic features.
Features

We currently use 72 atomic features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Features</strong> (18)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective function coefficients</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Number of constraints</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Statistics for constraint degrees</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Statistics for constraint coefficients</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
We currently use 72 atomic features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Features (18)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective function coefficients</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Number of constraints</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Statistics for constraint degrees</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Statistics for constraint coefficients</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><strong>Dynamic Features (54)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slack and ceil distances</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Pseudocosts</td>
<td>5</td>
<td>[Achterberg, 2009]</td>
</tr>
<tr>
<td>Infeasibility statistics</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Statistics for constraint degrees</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Min/max for ratios of constraint coefficients to RHS</td>
<td>4</td>
<td>[Alvarez et al., 2014]</td>
</tr>
<tr>
<td>Min/max for one-to-all coefficient ratios</td>
<td>8</td>
<td>[Alvarez et al., 2014]</td>
</tr>
<tr>
<td>Stats. for active constraint coefficients</td>
<td>24</td>
<td>[Patel and Chinneck, 2007]</td>
</tr>
</tbody>
</table>
How do we obtain the *labels* $y_j$? Some alternatives:

- The SB scores themselves:
  - Valid, but sensitive to variables with low SB scores (noisy)

- A label $1$ to the variable with max. SB score, and a label $0$ otherwise:
  - Too strict, ignores the fact that there may be multiple "good" variables that deserve a label of $1$.

**Relaxed Binary Labels**:

$$ y_j = \begin{cases} 1, & \text{if SB score of } x_j \text{ is within } \alpha \text{ of max. SB score at } N_i \geq 0, \\ 0, & \text{otherwise} \end{cases} $$

where $\alpha \in (0, 1)$ is small.
How do we obtain the *labels* $y_j$? Some alternatives:

- *The SB scores themselves:*

\[ y_j = \begin{cases} 1, & \text{if SB score of } x_j \text{ is within } \alpha \text{ of max. SB score at } N_i \\ 0, & \text{otherwise} \end{cases} \]

where $\alpha \in (0, 1)$ is small.
Labels

How do we obtain the *labels* $y_j^i$? Some alternatives:

- *The SB scores themselves:*
  - Valid, but sensitive to variables with low SB scores (noisy)

- *Relaxed Binary Labels*
  
  $y_j^i = \begin{cases} 
  1, & \text{if SB score of } x_j \text{ is within } \alpha \text{ of max. SB score at } N_i \\
  0, & \text{otherwise} \end{cases}$

  where $\alpha \in (0, 1)$ is small.
How do we obtain the labels $y^j$? Some alternatives:

- *The SB scores themselves*:
  - Valid, but sensitive to variables with low SB scores (noisy)

- *A label 1 to the variable with max. SB score, and a label 0 otherwise*:
How do we obtain the labels $y^j_i$? Some alternatives:

- **The SB scores themselves:**
  - Valid, but sensitive to variables with low SB scores (noisy)

- **A label 1 to the variable with max. SB score, and a label 0 otherwise:**
  - Too strict, ignores the fact that there may be multiple “good” variables that deserve a label of 1.
Labels

How do we obtain the labels $y_j^i$? Some alternatives:

- **The SB scores themselves:**
  - Valid, but sensitive to variables with low SB scores (noisy)

- **A label 1 to the variable with max. SB score, and a label 0 otherwise:**
  - Too strict, ignores the fact that there may be multiple “good” variables that deserve a label of 1.

- **Relaxed Binary Labels:**
How do we obtain the labels $y_j^i$? Some alternatives:

- **The SB scores themselves:**
  - Valid, but sensitive to variables with low SB scores (noisy)

- **A label 1 to the variable with max. SB score, and a label 0 otherwise:**
  - Too strict, ignores the fact that there may be multiple “good” variables that deserve a label of 1.

- **Relaxed Binary Labels:**
  
  $$y_j^i = \begin{cases} 1, & \text{if SB score of } x_j \text{ is within } \alpha \text{ of max. SB score at } N_i \\ 0, & \text{otherwise} \end{cases}$$

  where $\alpha \in (0, 1)$ is small.
Model Learning

**Goal**

Learn a function of the features that ranks variables with better labels higher than other variables.
Model Learning

**Goal**

Learn a function of the features that ranks variables with better labels higher than other variables.

- “Learning to rank with pairwise loss”, popular in web search [Joachims, 2002]
Model Learning

Goal

Learn a function of the features that ranks variables with better labels higher than other variables.

- “Learning to rank with pairwise loss”, popular in web search [Joachims, 2002]
- Assume the ranking function is linear in the features, i.e. \( f : \mathbb{R}^p \rightarrow \mathbb{R} \), and \( w \in \mathbb{R}^p \):
  \[
  f(\phi(x_j, N_i)) = w^T \cdot \phi(x_j, N_i)
  \]
Goal

Learn a function of the features that ranks variables with better labels higher than other variables.

- “Learning to rank with pairwise loss”, popular in web search [Joachims, 2002]
- Assume the ranking function is linear in the features, i.e. \( f : \mathbb{R}^p \rightarrow \mathbb{R} \), and \( \mathbf{w} \in \mathbb{R}^p \):
  \[
  f(\phi(x_j, N_i)) = \mathbf{w}^T \cdot \phi(x_j, N_i)
  \]
- Convex SVM optimization problem; can be solved efficiently with cutting plane algorithm of [Joachims, 2006].
Model Learning

Goal
Learn a function of the features that ranks variables with better labels higher than other variables.

- “Learning to rank with pairwise loss”, popular in web search [Joachims, 2002]
- Assume the ranking function is linear in the features, i.e. \( f : \mathbb{R}^p \rightarrow \mathbb{R} \), and \( \mathbf{w} \in \mathbb{R}^p \):
  \[
  f(\phi(x_j, N_i)) = \mathbf{w}^T \cdot \phi(x_j, N_i)
  \]
- Convex SVM optimization problem; can be solved efficiently with cutting plane algorithm of [Joachims, 2006].
- Open-source implementation SVM\(^\text{rank} \):
  http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
ML-based Branching

- After collecting data for the first $\theta$ nodes, and learning the ranking model, we start branching based on the learned model.
- No SB anymore.
After collecting data for the first \( \theta \) nodes, and learning the ranking model, we start branching based on the learned model.

No SB anymore.

Complexity (per node):
- \( O(\text{nnz}(A)) \) for feature computation (line 2)
- \( O(p \cdot \kappa) \) for scoring by dot product (line 3)

**Algorithm: ML-based Branching**

**Input:** node \( N_i \), candidate set \( C_i \), \(|C_i| \leq \kappa\)

**Output:** variable \( x_k \) s.t. \( k \in C_i \)

1. for each \( j \in C_i \) do
2.  Compute the features by \( \phi(x_j, N_i) \)
3.  Compute the score \( s_j = f(\phi(x_j, N_i)) \)
4. end
5. Branch on \( x_k \), where \( s_k = \max_{j \in C_i} \{s_j\} \)
Experimental Setup

- All algorithms implemented using control callbacks in CPLEX 12.6.1
- Cuts applied at the root, disabled afterwards
- Optimal value provided as upper cutoff
- Number of simplex iterations for any SB call is 50
- MIPLIB2010 “Benchmark” set with a time cutoff of 5 hours; default gap tolerances
Experimental Setup

- Competing Strategies:
  - sb+ml ($\theta = 500, \kappa = 10$): our method with $\alpha = 0$.2 and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
  - pc: pseudocost branching with SB initialization
  - sb+pc ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after

Other Strategies:
  - cplex-d: enter branching callback and branch as CPLEX suggests
  - sb ($\kappa = 10$): SB at each node

MIPLIB2010 "Benchmark" set; 84 feasible instances, each with 10 random seeds. From the 840 instances, 523 remain after filtering (too easy, too hard, etc.)

Similar experimental procedures used in [Fischetti and Monaci, 2012, Kılınç-Karzan et al., 2009].
Competing Strategies:

- **SB+ML** ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$. Other Strategies:

- **pc**: pseudocost branching with SB initialization
- **sb+pc** ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after
Experimental Setup

- Competing Strategies:
  - **SB+ML** \((\theta = 500, \kappa = 10)\): our method with \(\alpha = 0.2\) and SVM parameter \(C = 0.1\). Results are consistent for other values of \(\alpha\) and \(C\).
  - **PC**: pseudocost branching with SB initialization
Competing Strategies:

- **SB+ML** ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
- **PC**: pseudocost branching with SB initialization
- **SB+PC** ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after
Experimental Setup

- Competing Strategies:
  - SB+ML ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
  - PC: pseudocost branching with SB initialization
  - SB+PC ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after

- Other Strategies:

Elias Khalil (Georgia Tech)
Experimental Setup

- Competing Strategies:
  - SB+ML ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
  - PC: pseudocost branching with SB initialization
  - SB+PC ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after

- Other Strategies:
  - CPLEX-D: enter branching callback and branch as CPLEX suggests
Experimental Setup

- **Competing Strategies:**
  - **SB + ML** ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
  - **PC**: pseudocost branching with SB initialization
  - **SB + PC** ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after

- **Other Strategies:**
  - **CPLEX-D**: enter branching callback and branch as CPLEX suggests
  - **SB** ($\kappa = 10$): SB at each node
Experimental Setup

- **Competing Strategies:**
  - **SB+ML** ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
  - **PC**: pseudocost branching with SB initialization
  - **SB+PC** ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after

- **Other Strategies:**
  - **CPLEX-D**: enter branching callback and branch as CPLEX suggests
  - **SB** ($\kappa = 10$): SB at each node

- **MIPLIB2010** “Benchmark” set; 84 feasible instances, each with 10 random seeds. From the 840 instances, 523 remain after filtering (too easy, too hard, etc.)
Experimental Setup

- **Competing Strategies:**
  - SB+ML ($\theta = 500, \kappa = 10$): our method with $\alpha = 0.2$ and SVM parameter $C = 0.1$. Results are consistent for other values of $\alpha$ and $C$.
  - PC: pseudocost branching with SB initialization
  - SB+PC ($\theta = 500, \kappa = 10$): SB for first $\theta$ nodes, PC after

- **Other Strategies:**
  - CPLEX-D: enter branching callback and branch as CPLEX suggests
  - SB ($\kappa = 10$): SB at each node

- MIPLIB2010 “Benchmark” set; 84 feasible instances, each with 10 random seeds. From the 840 instances, 523 remain after filtering (too easy, too hard, etc.)

- Similar experimental procedures used in [Fischetti and Monaci, 2012, Kılınç-Karzan et al., 2009].
Results

<table>
<thead>
<tr>
<th>Unsolved Instances</th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
<th>SB+ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (523)</td>
<td>11</td>
<td>129</td>
<td>66</td>
<td>63</td>
<td>52</td>
</tr>
<tr>
<td>Easy (255)</td>
<td>0</td>
<td>12</td>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>Medium (120)</td>
<td>2</td>
<td>43</td>
<td>22</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>Hard (148)</td>
<td>9</td>
<td>74</td>
<td>29</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

- An instance is “Easy” if CPLEX-D solves in $\leq 50,000$ nodes, “Medium” in $\leq 500,000$ nodes, “Hard” otherwise.
- PC, SB+PC and SB+ML are competing, lower is better for all three criteria, and best is in bold.
## Results

<table>
<thead>
<tr>
<th>Unsolved Instances</th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
<th>SB+ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (523)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>Easy (255)</td>
<td>0</td>
<td>12</td>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>Medium (120)</td>
<td>2</td>
<td>43</td>
<td>22</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>Hard (148)</td>
<td>9</td>
<td>74</td>
<td>29</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Num. nodes</th>
<th>All (523)</th>
<th>Easy (255)</th>
<th>Medium (120)</th>
<th>Hard (148)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>All (523)</td>
<td>46,633</td>
<td>3,255</td>
<td>173,417</td>
<td>1,570,891</td>
<td></td>
</tr>
<tr>
<td>Easy (255)</td>
<td>33,072</td>
<td>3,610</td>
<td>121,923</td>
<td>519,878</td>
<td></td>
</tr>
<tr>
<td>Medium (120)</td>
<td>92,662</td>
<td>7,931</td>
<td>395,199</td>
<td>1,971,333</td>
<td></td>
</tr>
<tr>
<td>Hard (148)</td>
<td>70,455</td>
<td>5,224</td>
<td>288,916</td>
<td>1,979,660</td>
<td></td>
</tr>
</tbody>
</table>

- An instance is “Easy” if CPLEX-D solves in $\leq 50,000$ nodes, “Medium” in $\leq 500,000$ nodes, “Hard” otherwise
- PC, SB+PC and SB+ML are competing, lower is better for all three criteria, and best is in bold
An instance is “Easy” if CPLEX-D solves in $\leq 50,000$ nodes, “Medium” in $\leq 500,000$ nodes, “Hard” otherwise

PC, SB+PC and SB+ML are competing, lower is better for all three criteria, and best is in bold
Head-to-head Comparisons: Win/Tie/Loss

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5/264/0/125/123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>8/164/0/285/63</td>
<td>68/63/0/326/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>8/227/0/225/60</td>
<td>72/66/7/315/6</td>
<td>15/320/0/125/12</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>8/267/0/196/49</td>
<td>82/96/7/286/5</td>
<td>21/355/0/95/7</td>
<td>17/300/58/96/6</td>
</tr>
</tbody>
</table>

**Table**: Each cell has the numbers of: Abs. Win/Win/Tie/Loss/Abs. Loss

Compare branching strategy $A$ to $B$:

- **absolute win** for $A$ vs. $B$ on instance $I$ $⇔$ $A$ solves $I$, $B$ does not
- **win** for $A$ vs. $B$ on $I$ $⇔$ $A$ and $B$ solve $I$, and $A$ does so in fewer nodes than $B$
- **tie** between $A$ and $B$ on $I$ $⇔$ $A$ and $B$ solve $I$ in same num. of nodes
### Head-to-head Comparisons: Win/Tie/Loss

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5/264/0/125/123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>8/164/0/285/63</td>
<td>68/63/0/326/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>8/227/0/225/60</td>
<td>72/66/7/315/6</td>
<td>15/320/0/125/12</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>8/267/0/196/49</td>
<td>82/96/7/286/5</td>
<td>21/355/0/95/7</td>
<td>17/300/58/96/6</td>
</tr>
</tbody>
</table>

**Table:** Each cell has the numbers of: Abs. Win/Win/Tie/Loss/Abs. Loss

**Compare branching strategy $\mathcal{A}$ to $\mathcal{B}$:**

- **absolute win** for $\mathcal{A}$ vs. $\mathcal{B}$ on instance $I$ $\iff$ $\mathcal{A}$ solves $I$, $\mathcal{B}$ does not
- **win** for $\mathcal{A}$ vs. $\mathcal{B}$ on $I$ $\iff$ $\mathcal{A}$ and $\mathcal{B}$ solve $I$, and $\mathcal{A}$ does so in fewer nodes than $\mathcal{B}$
- **tie** between $\mathcal{A}$ and $\mathcal{B}$ on $I$ $\iff$ $\mathcal{A}$ and $\mathcal{B}$ solve $I$ in same num. of nodes
### Head-to-head Comparisons: Win/Tie/Loss

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5/264/0/125/123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>8/164/0/285/63</td>
<td>68/63/0/326/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>8/227/0/225/60</td>
<td>72/66/7/315/6</td>
<td>15/320/0/125/12</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>8/267/0/196/49</td>
<td>82/96/7/286/5</td>
<td>21/355/0/95/7</td>
<td>17/300/58/96/6</td>
</tr>
</tbody>
</table>

**Table:** Each cell has the numbers of: Abs. Win/Win/Tie/Loss/Abs. Loss

Compare branching strategy $\mathcal{A}$ to $\mathcal{B}$:

- **absolute win** for $\mathcal{A}$ vs. $\mathcal{B}$ on instance $\mathcal{I} \Leftrightarrow \mathcal{A}$ solves $\mathcal{I}$, $\mathcal{B}$ does not
- **win** for $\mathcal{A}$ vs. $\mathcal{B}$ on $\mathcal{I} \Leftrightarrow \mathcal{A}$ and $\mathcal{B}$ solve $\mathcal{I}$, and $\mathcal{A}$ does so in fewer nodes than $\mathcal{B}$
- **tie** between $\mathcal{A}$ and $\mathcal{B}$ on $\mathcal{I} \Leftrightarrow \mathcal{A}$ and $\mathcal{B}$ solve $\mathcal{I}$ in same num. of nodes
Head-to-head Comparisons: Win/Tie/Loss

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5/264/0/125/123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>8/164/0/285/63</td>
<td>68/63/0/326/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>8/227/0/225/60</td>
<td>72/66/7/315/6</td>
<td>15/320/0/125/12</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>8/267/0/196/49</td>
<td>82/96/7/286/5</td>
<td>21/355/0/95/7</td>
<td>17/300/58/96/6</td>
</tr>
</tbody>
</table>

Table: Each cell has the numbers of: Abs. Win/Win/Tie/Loss/Abs. Loss

Compare branching strategy $A$ to $B$:

- **absolute win** for $A$ vs. $B$ on instance $I$ $\Leftrightarrow$ $A$ solves $I$, $B$ does not
- **win** for $A$ vs. $B$ on $I$ $\Leftrightarrow$ $A$ and $B$ solve $I$, and $A$ does so in fewer nodes than $B$
- **tie** between $A$ and $B$ on $I$ $\Leftrightarrow$ $A$ and $B$ solve $I$ in same num. of nodes
### Head-to-head Comparisons: Win/Tie/Loss

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5/264/0/125/123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>8/164/0/285/63</td>
<td>68/63/0/326/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>8/227/0/225/60</td>
<td>72/66/7/315/6</td>
<td>15/320/0/125/12</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>8/267/0/196/49</td>
<td>82/96/7/286/5</td>
<td>21/355/0/95/7</td>
<td>17/300/58/96/6</td>
</tr>
</tbody>
</table>

**Table:** Each cell has the numbers of: Abs. Win/Win/Tie/Loss/Abs. Loss

Compare branching strategy $\mathcal{A}$ to $\mathcal{B}$:

- **absolute win** for $\mathcal{A}$ vs. $\mathcal{B}$ on instance $\mathcal{I}$ ⇔ $\mathcal{A}$ solves $\mathcal{I}$, $\mathcal{B}$ does not
- **win** for $\mathcal{A}$ vs. $\mathcal{B}$ on $\mathcal{I}$ ⇔ $\mathcal{A}$ and $\mathcal{B}$ solve $\mathcal{I}$, and $\mathcal{A}$ does so in fewer nodes than $\mathcal{B}$
- **tie** between $\mathcal{A}$ and $\mathcal{B}$ on $\mathcal{I}$ ⇔ $\mathcal{A}$ and $\mathcal{B}$ solve $\mathcal{I}$ in same num. of nodes
Future Directions

- Adaptive learning w.r.t. the evolution of the search
Future Directions

- Adaptive learning w.r.t. the evolution of the search
- Fully online or offline learning

Figure: Cut Selection Strategy in SCIP (slide from [Wolter, 2006])

- **Efficacy**, i.e., distance of the hyperplane to the LP solution
- **Orthogonality** with respect to the other cuts
- **Parallelism** with respect to the objective function

⇒ Select cuts with largest value of $e_r + w_0 \times o_r + w_p \times p_r$

Weights of the criteria can be adjusted

- $\text{ORTHOFAC} = 1.0$
- $\text{OBJPARALFAC} = 0.0001$

---

Elias Khalil (Georgia Tech)  Learning to Branch in MIP  February 16, 2016
Future Directions

- Adaptive learning w.r.t. the evolution of the search
- Fully online or offline learning
- Most decisions in the MIP solver are inherently heuristic

Figure: Cut Selection Strategy in SCIP (slide from [Wolter, 2006])

- Efficacy, i.e., distance of the hyperplane to the LP solution
- Orthogonality with respect to the other cuts
- Parallelism with respect to the objective function

⇒ Select cuts with largest value of $e_r + w_o \ast o_r + w_p \ast p_r$

Weights of the criteria can be adjusted
- $\text{ORTHOFAC} = 1.0$
- $\text{OBJPARALFAC} = 0.0001$

8 / 36
Future Directions

- Adaptive learning w.r.t. the evolution of the search
- Fully online or offline learning
- Most decisions in the MIP solver are inherently heuristic
  - Can we use data and ML to make better informed decisions?

Figure: Cut Selection Strategy in SCIP (slide from [Wolter, 2006])

Cut Selection Strategy

- **Efficacy**, i.e., distance of the hyperplane to the LP solution
- **Orthogonality** with respect to the other cuts
- **Parallelism** with respect to the objective function

⇒ Select cuts with largest value of \( e_r + w_o o_r + w_p p_r \)

Weights of the criteria can be adjusted
- \( \text{ORTHOFAC} = 1.0 \)
- \( \text{OBJPARALFAC} = 0.0001 \)
Thanks!
Questions?

Elias Khalil
elias.khalil@cc.gatech.edu
## Features Description

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Count</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Features (18)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective function coeffs.</td>
<td>Value of the coefficient (raw, positive only, negative only)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Num. constraints</td>
<td>Number of constraints that the variable participates in (with a non-zero coefficient)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Stats. for constraint degrees</td>
<td>The degree of a constraint is the number of variables that participate in it. A variable may participate in multiple constraints, and statistics over those constraints' degrees are used. The constraint degree is computed on the root LP (mean, stdev., min, max)</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Stats. for constraint coeffs.</td>
<td>A variable’s positive (negative) coefficients in the constraints it participates in (count, mean, stdev., min, max)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><strong>Dynamic Features (54)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slack and ceil distances</td>
<td>$\min{x_j^i - \lceil x_j^i \rceil, \lceil x_j^i \rceil - \lfloor x_j^i \rfloor, \lfloor x_j^i \rfloor - \lceil x_j^i \rceil}$</td>
<td>2</td>
<td>(Achterberg 2009)</td>
</tr>
<tr>
<td>Pseudocosts</td>
<td>Upwards and downwards values, and their corresponding ratio, sum and product, weighted by the fractionality of $x_j$.</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Infeasibility statistics</td>
<td>Number and fraction of nodes for which applying SB to variable $x_j$ led to one (two) infeasible children (during data collection)</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Stats. for constraint degrees</td>
<td>A dynamic variant of the static version above. Here, the constraint degrees are on the current node’s LP. The ratios of the static mean, maximum and minimum to their dynamic counterparts are also features.</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Min/max for ratios of constraint coeffs. to RHS</td>
<td>Minimum and maximum ratios across positive and negative right-hand-sides (RHS)</td>
<td>4</td>
<td>(Alvarez, Louveaux, and Wehenkel 2014)</td>
</tr>
<tr>
<td>Min/max for one-to-all coefficient ratios</td>
<td>The statistics are over the ratios of a variable’s coefficient, to the sum over all other variables’ coefficients, for a given constraint. Four versions of these ratios are considered: positive (negative) coefficient to sum of positive (negative) coefficients</td>
<td>8</td>
<td>(Alvarez, Louveaux, and Wehenkel 2014)</td>
</tr>
<tr>
<td>Stats. for active constraint coefficients</td>
<td>An active constraint at a node LP is one which is binding with equality at the optimum. We consider 4 weighting schemes for an active constraint: unit weight, inverse of the sum of the coefficients of all variables in constraint, inverse of the sum of the coefficients of only candidate variables in constraint, dual cost of the constraint. Given the absolute value of the coefficients of $x_j$ in the active constraints, we compute the sum, mean, stdev., max. and min. of those values, for each of the weighting schemes. We also compute the weighted number of active constraints that $x_j$ is in, with the same 4 weightings</td>
<td>24</td>
<td>(Patel and Chinneck 2007)</td>
</tr>
</tbody>
</table>
Performance Profiles

Performance profile for Nodes

Performance profile for Time

Fraction of instances

Ratio to best

Graphs showing performance profiles for different configurations:
- cplex-d
- sb
- pc
- sb+pc
- sb+ml

Legend:
- cplex-d
- sb
- pc
- sb+pc
- sb+ml
## Head-to-head Comparisons: Node Ratios

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
<th>SB+ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td>1.39 (389)</td>
<td>0.64 (449)</td>
<td>0.84 (452)</td>
<td>0.97 (463)</td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>0.72 (389)</td>
<td>0.47 (389)</td>
<td>0.61 (388)</td>
<td>0.76 (389)</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>1.56 (449)</td>
<td>2.11 (389)</td>
<td>1.34 (445)</td>
<td>1.59 (450)</td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>1.20 (452)</td>
<td>1.63 (388)</td>
<td>0.75 (445)</td>
<td>1.22 (454)</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>1.03 (463)</td>
<td>1.32 (389)</td>
<td>0.63 (450)</td>
<td>0.82 (454)</td>
<td></td>
</tr>
</tbody>
</table>
Head-to-head Comparisons: Win/Tie/Loss (averaging over seeds)

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>45/0/20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>20/0/45</td>
<td>5/0/60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>28/0/37</td>
<td>8/0/57</td>
<td>51/0/14</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>33/0/32</td>
<td>10/0/55</td>
<td>55/0/10</td>
<td>46/6/13</td>
</tr>
</tbody>
</table>

**Table:** Win-tie-loss counts for every pair of strategies, for the number of nodes. A triplet in a cell in row $A$ and column $B$ expresses the number of wins for $A$ over $B$, the number of ties, and the number of losses of $A$ to $B$, respectively, w.r.t. to the shifted geometric mean of the number of nodes. $A$ wins over $B$ on instance $I$ iff the mean number of nodes of $A$ over the random seeds on $I$ is strictly less than that of $B$; losses and ties are defined analogously.
Analysis of Feature Ranking Models

- Compare ranking models across instances
- For each model, consider the ranking of the features by their absolute weight in the linear model
- Similarity of two models measured by *Spearman’s rank correlation coefficient* between their respective feature weight rankings

![Box plot of Spearman's rank correlation coefficients]
The 10 features that appear the most in the top $K = 10$ features over models, sorted by that count (corresponding column is bold in the header). The counts for other values of $K$ are also shown. There are 2700 features in total. Static features are marked by (S); unmarked features are dynamic.

<table>
<thead>
<tr>
<th>Feature</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC Product</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>23</td>
<td>30</td>
<td>37</td>
</tr>
<tr>
<td>PC Product $\times$ PC Product</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>17</td>
<td>24</td>
<td>38</td>
</tr>
<tr>
<td>PC Product $\times$ Mean Constraint Degree</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>13</td>
<td>19</td>
<td>33</td>
</tr>
<tr>
<td>PC Product $\times$ Max. Constraint Degree</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>13</td>
<td>19</td>
<td>34</td>
</tr>
<tr>
<td>PC Product $\times$ Min. Constraint Degree</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>13</td>
<td>21</td>
<td>34</td>
</tr>
<tr>
<td>PC Product $\times$ Max. Absolute Coefficient in Active Constraints</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>13</td>
<td>27</td>
<td>37</td>
</tr>
<tr>
<td>PC Product $\times$ Mean Absolute Coefficient in Active Constraints</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>25</td>
<td>35</td>
</tr>
<tr>
<td>PC Product $\times$ Num. Constraints (S)</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>17</td>
<td>28</td>
</tr>
<tr>
<td>PC Product $\times$ Min. Positive Constraint Coefficient (S)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>18</td>
<td>32</td>
</tr>
<tr>
<td>PC Product $\times$ Max. Ratio of Constraint Degree (static/dynamic)</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>34</td>
</tr>
</tbody>
</table>
**Results with cuts and heuristics, no upper cutoff**

- “Easy” if CPLEX solves in \( \leq 50,000 \) nodes, “Medium” in \( \leq 500,000 \) nodes, “Hard” otherwise
- PC, SB+PC and SB+ML are competing, lower is better for all three criteria, and best is in bold

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
<th>SB+ML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instances unsolved</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All (538)</td>
<td>28</td>
<td>139</td>
<td>55</td>
<td>59</td>
<td>54</td>
</tr>
<tr>
<td>Easy (318)</td>
<td>3</td>
<td>23</td>
<td>21</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>Medium (140)</td>
<td>18</td>
<td>74</td>
<td>24</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>Hard (80)</td>
<td>7</td>
<td>42</td>
<td>10</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td><strong>Num. nodes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All (538)</td>
<td>25,407</td>
<td>17,909</td>
<td>43,120</td>
<td>36,235</td>
<td><strong>32,948</strong></td>
</tr>
<tr>
<td>Easy (318)</td>
<td>3,643</td>
<td>3,794</td>
<td>7,371</td>
<td>5,386</td>
<td><strong>5,358</strong></td>
</tr>
<tr>
<td>Medium (140)</td>
<td>165,188</td>
<td>91,279</td>
<td>260,851</td>
<td>266,497</td>
<td><strong>217,306</strong></td>
</tr>
<tr>
<td>Hard (80)</td>
<td>2,144,179</td>
<td>491,557</td>
<td>2,062,439</td>
<td>2,142,617</td>
<td><strong>1,649,930</strong></td>
</tr>
<tr>
<td><strong>Total time</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All (538)</td>
<td>681</td>
<td>2,273</td>
<td><strong>1,027</strong></td>
<td>1,229</td>
<td>1,423</td>
</tr>
<tr>
<td>Easy (318)</td>
<td>203</td>
<td>821</td>
<td><strong>385</strong></td>
<td>498</td>
<td>630</td>
</tr>
<tr>
<td>Medium (140)</td>
<td>2,978</td>
<td>9,942</td>
<td><strong>3,767</strong></td>
<td>4,077</td>
<td>4,210</td>
</tr>
<tr>
<td>Hard (80)</td>
<td>6,223</td>
<td>9,815</td>
<td><strong>5,192</strong></td>
<td>5,461</td>
<td>5,423</td>
</tr>
</tbody>
</table>
### Head-to-head Comparisons: Win/Tie/Loss

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>5/259/0/135/116</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>20/183/0/280/47</td>
<td>98/60/0/325/14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>21/216/0/242/52</td>
<td>94/96/1/288/14</td>
<td>22/264/0/193/26</td>
<td></td>
</tr>
</tbody>
</table>

**Definitions:**

- **absolute win** for \( A \) v/s \( B \) on instance \( I \) ⇔ \( A \) solves \( I \), \( B \) does not
- **win** for \( A \) v/s \( B \) on \( I \) ⇔ \( A \) and \( B \) solve \( I \), and \( A \) does so in \(<\) nodes than \( B \)
- **tie** between \( A \) and \( B \) on \( I \) ⇔ \( A \) and \( B \) solve \( I \) in same num. of nodes
### Head-to-head Comparisons: Node Ratios

Table: Ratios for the shifted geometric means (shift 10) over nodes on instances solved by both strategies. The first value in a cell in row $A$ and column $B$ is the ratio of the average number of nodes used by $A$ to that of $B$. The second value is the number of instances solved by both $A$ and $B$.

<table>
<thead>
<tr>
<th></th>
<th>CPLEX-D</th>
<th>SB</th>
<th>PC</th>
<th>SB+PC</th>
<th>SB+ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX-D</td>
<td>1.38 (394)</td>
<td>0.64 (463)</td>
<td>0.79 (458)</td>
<td>0.84 (462)</td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>0.72 (394)</td>
<td>0.48 (385)</td>
<td>0.65 (385)</td>
<td>0.67 (386)</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>1.56 (463)</td>
<td>2.09 (385)</td>
<td>1.24 (457)</td>
<td>1.32 (461)</td>
<td></td>
</tr>
<tr>
<td>SB+PC</td>
<td>1.27 (458)</td>
<td>1.55 (385)</td>
<td>0.81 (457)</td>
<td>1.08 (460)</td>
<td></td>
</tr>
<tr>
<td>SB+ML</td>
<td>1.20 (462)</td>
<td>1.49 (386)</td>
<td>0.76 (461)</td>
<td>0.92 (460)</td>
<td></td>
</tr>
</tbody>
</table>
Performance Profiles

Performance profile for Nodes

Performance profile for Time

- cplex-d
- sb
- pc
- sb+pc
- sb+ml
Pseudocost Branching (PC)

- **Goal:** Imitate SB without many simplex iterations.
- **Record change in objective value over time, and use historical averages as proxy for SB scores.**
- **Objective increase per unit change in** $x_j$ **at** $N_i$ **when branching down:**

\[
\varsigma_{i,j}^- = \frac{\Delta_j^-}{f_j^-}
\]

with $f_j^- = \ddot{x}_j^i - \lfloor \ddot{x}_j^i \rfloor$

- **Take average over those events:**

\[
\psi_j^- = \frac{\sum \varsigma_{i,j}^-}{\text{# times branched down on } x_j}
\]

- **PC score at some node:**

\[
s_j = \text{score}(f_j^- \psi_j^-, f_j^+ \psi_j^+)
\]


Related Work

- [Achterberg and Berthold, 2009],[Hendel, 2015]: recent PC-based strategies
- [Kılınç-Karzan, Savelsbergh and Nemhauser, 2009]: collect information on which variables are likely to lead to fathomed child nodes quickly, then restart solve and branch on those
- [He et al., 2014]: ML (classification) for node selection; offline, can prune optimum
- [Alvarez et al., 2014]: ML (regression) for variable selection; offline, slow and modest results