Learning To Scale

Timo Berthold
FICO Xpress Optimization

Optimization Solutions
for use by Business Users and Business Analysts

Optimization Technology
4 Modules for use by Operations Researchers, Data Scientists and Solution Developers

Xpress Solutions

Xpress Mosel
Visualize, collaborate and decide using Solutions

Xpress Workbench
Develop models, services and solutions

Xpress Executor
Deploy and execute models

Xpress Solver
Optimization technologies

FICO Professional Services

Make Decisions that meet your Objectives
FICO Xpress Optimization – Optimization Technology

FICO Xpress Optimization

Xpress Solutions

Xpress Insight
Visualize, collaborate and decide using Solutions

Xpress Workbench
Develop models, services and solutions

Xpress Executor
Deploy and execute models

Xpress Solver
Optimization technologies

Performance
Widest breadth of industry-leading optimization algorithms and technologies

Features
Highest degree of customizability through advanced callbacks, permissive controls and flexible modelling

Robustness
Highest degree of determinism and a variety of features for numeric stability
Motivation: ML for Mathematical Optimization
<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning to Search in Branch and Bound Algorithms</td>
<td>2010</td>
</tr>
<tr>
<td>Online Mixed-Integer Optimization in Milliseconds</td>
<td>2019</td>
</tr>
<tr>
<td>Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks</td>
<td>2018</td>
</tr>
<tr>
<td>Learning for constrained optimization: identifying optimal active constraint sets</td>
<td>2018</td>
</tr>
<tr>
<td>The Voice of Optimization</td>
<td>2018</td>
</tr>
<tr>
<td>Reinforcement Learning for Integer Programming: Learning to Cut</td>
<td>2019</td>
</tr>
<tr>
<td>Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks</td>
<td>2018</td>
</tr>
<tr>
<td>Reinforcement Learning for Variable Selection in a Branch and Bound Algorithm</td>
<td>2020</td>
</tr>
<tr>
<td>Parameterizing Branch-and-Bound Search Trees to Learn Branching Policies</td>
<td>2020</td>
</tr>
<tr>
<td>Learning Generalized Strong Branching for Set Covering, Set Packing, and D-1 Knapsack Problems</td>
<td>2020</td>
</tr>
<tr>
<td>Learning to Search via Retrospective Imitation</td>
<td>2019</td>
</tr>
<tr>
<td>Exact Combinatorial Optimization with Graph Convolutional Neural Networks</td>
<td>2019</td>
</tr>
<tr>
<td>Learning to Branch: Accelerating Resource Allocation in Wireless Networks</td>
<td>2019</td>
</tr>
<tr>
<td>Cuts, Primal Heuristics, and Learning to Branch for the Time-Dependent Traveling Salesman Problem</td>
<td>2018</td>
</tr>
<tr>
<td>Learning to branch</td>
<td>2018</td>
</tr>
<tr>
<td>Accelerating the branch-and-price algorithm using machine learning</td>
<td>2018</td>
</tr>
<tr>
<td>Deep Learning Assisted Heuristic Tree Search for the Container Pre-marshalling Problem</td>
<td>2017</td>
</tr>
<tr>
<td>On learning and branching: a survey</td>
<td>2017</td>
</tr>
<tr>
<td>A Machine Learning-Based Approximation of Strong Branching</td>
<td>2017</td>
</tr>
<tr>
<td>Online Learning for Strong Branching Approximation in Branch-and-Bound</td>
<td>2016</td>
</tr>
<tr>
<td>Learning to branch in mixed integer programming</td>
<td>2016</td>
</tr>
<tr>
<td>Machine Learning for Integer Programming</td>
<td>2016</td>
</tr>
<tr>
<td>Learning to Search in Branch and Bound Algorithms</td>
<td>2014</td>
</tr>
<tr>
<td>A Supervised Machine Learning Approach to Variable Branching in Branch-And-Bound</td>
<td>2014</td>
</tr>
<tr>
<td>Dynamic Approach for Switching Heuristics</td>
<td>2013</td>
</tr>
<tr>
<td>Guiding Combinatorial Optimization with UCT</td>
<td>2012</td>
</tr>
</tbody>
</table>

Huge Activity in Machine Learning for Mathematical Optimization

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning to Search via Retrospective Imitation</td>
<td>2019</td>
</tr>
<tr>
<td>Learning to Run Heuristics in Tree Search</td>
<td>2017</td>
</tr>
<tr>
<td>Algorithm Runtime Prediction: Methods &amp; Evaluation</td>
<td>2012</td>
</tr>
<tr>
<td>Automated Configuration of Mixed Integer Programming Solvers</td>
<td>2010</td>
</tr>
<tr>
<td>MIPaal: Mixed Integer Program as a Layer</td>
<td>2019</td>
</tr>
<tr>
<td>End to end learning and optimization on graphs</td>
<td>2019</td>
</tr>
<tr>
<td>SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver</td>
<td>2019</td>
</tr>
<tr>
<td>Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization</td>
<td>2018</td>
</tr>
<tr>
<td>Smart “Predict, then Optimize”</td>
<td>2017</td>
</tr>
<tr>
<td>Attention, Learn to Solve Routing Problems!</td>
<td>2018</td>
</tr>
<tr>
<td>Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search</td>
<td>2018</td>
</tr>
<tr>
<td>Learning Combinatorial Optimization Algorithms over Graphs</td>
<td>2017</td>
</tr>
<tr>
<td>Neural Combinatorial Optimization with Reinforcement Learning</td>
<td>2016</td>
</tr>
<tr>
<td>Generation techniques for linear programming instances with controllable properties</td>
<td>2017</td>
</tr>
<tr>
<td>Stress testing mixed integer programming solvers through new test instance generation methods</td>
<td>2019</td>
</tr>
<tr>
<td>How to Evaluate Machine Learning Approaches for Combinatorial Optimization: Application to the Travelling Salesman Problem</td>
<td>2019</td>
</tr>
<tr>
<td>Learning MILP Resolution Outcomes Before Reaching Time-Limit</td>
<td>2018</td>
</tr>
<tr>
<td>Learning to steer nonlinear interior-point methods</td>
<td>2019</td>
</tr>
<tr>
<td>Learning when to use a decomposition</td>
<td>2018</td>
</tr>
<tr>
<td>Machine Learning for Combinatorial Optimization: a Methodological Tour d’Horizon</td>
<td>2018</td>
</tr>
<tr>
<td>Adaptive Large Neighborhood Search for Mixed Integer Programming</td>
<td>2018</td>
</tr>
<tr>
<td>Learning a Classification of Mixed-Integer Quadratic Programming</td>
<td>2017</td>
</tr>
<tr>
<td>OptNet: Differentiable Optimization as a Layer in Neural Networks</td>
<td>2017</td>
</tr>
<tr>
<td>Global Deterministic Optimization with Artificial Neural Networks</td>
<td>2018</td>
</tr>
<tr>
<td>Embedded</td>
<td>2018</td>
</tr>
<tr>
<td>Large Scale Learning To Rank</td>
<td>2020</td>
</tr>
<tr>
<td>A General Large Neighborhood Search Framework for Solving Integer Programs</td>
<td>2020</td>
</tr>
</tbody>
</table>
Huge Activity in Machine Learning for Mathematical Optimization

Only very few of these implemented in general purpose MIP solvers! (and activated by default)
Complex decisions, no easy answers

Sophisticated rules already in place

We don't even know good features
Huge Activity in Machine Learning for Mathematical Optimization

Scaling!!!

Either-or decision

We know meaningful features

Currently no rule in place
Numerical Stability

• Numerical stability is a crucial topic in many applications
  • Recent blog series on Numerics, visit https://community.fico.com/
    • Numerics I: Solid Like a Rock or Fragile Like a Flower?
    • Numerics II: Zoom Into the Unknown
    • Numerics III: Learning to Scale
    • tbc...

• Real-life applications often complex and numerically challenging to handle:
  • More than half of client problems seen in the past year had some mild numeric issues.
  • After performance, numeric failures are the most common support request.
    • Unexpected solution status
    • Inconsistent results
    • Performance issues (e.g. simplex cycling)
Information on numeric stability

• Since Xpress 8.6, we provide numeric analysis tools

• A priori: distribution of matrix, objective, rhs coefficients

<table>
<thead>
<tr>
<th>Coefficient range</th>
<th>original</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficients</td>
<td>[min, max] : [ 2.00e-06, 2.34e+02] / [ 1.25e-01, 1.67e+00]</td>
<td></td>
</tr>
<tr>
<td>RHS and bounds</td>
<td>[min, max] : [ 1.67e-01, 9.23e+03] / [ 1.67e-01, 8.21e+02]</td>
<td></td>
</tr>
<tr>
<td>Objective</td>
<td>[min, max] : [ 2.00e-06, 2.34e+02] / [ 2.00e-06, 2.34e+02]</td>
<td></td>
</tr>
</tbody>
</table>

comprises effects of presolving AND scaling

• A posteriori: report on numerical failures that the solver encountered

Numerical issues encountered:
- Dual failures : 78 out of 2194 (ratio: 0.0356)
- Primal failures: 5 out of 247  (ratio: 0.0202)
- Singular bases : 5 out of 11180 (ratio: 0.0004)
- Nodes w/LP fails : 9 out of 70  (ratio: 0.1286)
Condition Number

• The condition number $\kappa$ of a matrix $A$ provides a bound on how much a small change in $b$ can affect $x$.

• For a square, invertible matrix $B$

$$\kappa = \|B\| \cdot \|B^{-1}\|$$

• One purpose of scaling is to reduce the condition number.

• Sampling the condition number is an optional feature (MIPKAPPAFREQ=1)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Count</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>kappa stable</td>
<td>3757</td>
<td>0.0051</td>
</tr>
<tr>
<td>kappa suspicious</td>
<td>8476</td>
<td>0.0115</td>
</tr>
<tr>
<td>kappa unstable</td>
<td>723171</td>
<td>0.9831</td>
</tr>
<tr>
<td>kappa ill-posed</td>
<td>193</td>
<td>0.0003</td>
</tr>
<tr>
<td>Largest kappa</td>
<td>4.959805e+14</td>
<td></td>
</tr>
<tr>
<td>Attention level</td>
<td>0.2953</td>
<td></td>
</tr>
</tbody>
</table>

• Summarized in a single attention level from 0.0 (all stable) to 1.0 (anything goes).
The condition number $\kappa$ of a matrix $A$ provides a bound on how much a small change in $b$ can affect $x$.

For a square, invertible matrix $B$

$$\kappa = \|B\| \cdot \|B^{-1}\|$$

One purpose of scaling is to reduce the condition number.

Sampling the condition number is an optional feature.

Summarized in a single attention level from 0.0 (all stable) to 1.0 (anything goes).
What is Scaling?

• Scaling is a widely used preconditioning technique, used by various kinds of algorithms
  • to reduce the condition number of the constraint matrix
    • to reduce error propagation
  • to improve the numerical behavior of the algorithms
  • to reduce the number of iterations required to solve the problem

• More precisely, LP scaling refers to the (iterative) multiplication of rows and columns by scalars
  • to reduce the absolute magnitude of nonzero coefficients in matrix, rhs and objective
  • to reduce the relative difference of nonzero coefficients in matrix, rhs and objective
Scaling in Linear Programming

• Basic Linear Program (LP):
  \[
  \begin{align*}
  \text{max} & \quad cx \\
  \text{s.t.} & \quad Ax \leq b
  \end{align*}
  \]

• Scaling multiplies rows and columns to bring coefficients “on one scale”.
  • Typically, close to 1

• Two scaling methods:
  • **Standard**: Divide rows by largest coefficient and then divide columns by largest coefficient. Repeat.
  • **Curtis-Reid**: Minimize least-squares deviation from 1 (logarithmically).

\[
\begin{align*}
\text{max} & \quad (cD^C)(D^{C^{-1}}x) \\
\text{s.t.} & \quad (D^RAD^C)(D^{C^{-1}}x) \leq D^Rb \\
& \quad c' = cD^C, A' = D^RAD^C, b' = D^Rb \\
& \quad x' = D^{C^{-1}}x \\
\text{max} & \quad c'x' \\
\text{s.t.} & \quad A'x' \leq b'
\end{align*}
\]
Example

- We want to set up our home business to make boxes or chess pieces
  - We want to maximize profit [\$5/box, \$10/chess piece]
  - We have a limited amount of wood [100]
  - We have to buy tools [\$30 for boxes, \$500 for chess sets]

- A mixed integer programming (MIP) problem:

\[
\begin{align*}
\text{max} & \quad 5x_{\text{box}} + 10x_{\text{chess}} - 30b_{\text{box}} - 500b_{\text{chess}} \\
\text{s.t.} & \quad x_{\text{box}} + x_{\text{chess}} \leq 100 \\
& \quad x_{\text{box}} \leq 100b_{\text{box}} \\
& \quad x_{\text{chess}} \leq 100b_{\text{chess}} \\
& \quad b_{\text{box}}, b_{\text{chess}} \in \{0,1\}
\end{align*}
\]

- Coefficient matrix:

\[
\begin{bmatrix}
1 & 1 \\
1 & -100 \\
1 & -100
\end{bmatrix}
\]

... with a potential basis matrix
Example - Continued

- Unscaled:

\[
\begin{align*}
    x^{box} + x^{chess} & \leq 100 \\
    x^{box} & \leq 100 b^{box} \\
    x^{chess} & \leq 100 b^{chess}
\end{align*}
\]

\[
\begin{bmatrix}
    -\frac{1}{100} & -\frac{1}{100} & 1 \\
    -1 & 1 & 1
\end{bmatrix}
\]  

\[\kappa \approx 245\]
Example - Continued

• Unscaled:

\[
\begin{align*}
\begin{bmatrix}
\times^{box} \\
\times^{chess}
\end{bmatrix} + \begin{bmatrix}
1 \\
100
\end{bmatrix} & \leq 100 \\
\times^{box} & \leq 100b^{box} \\
\times^{chess} & \leq 100b^{chess}
\end{align*}
\]

\[
\begin{bmatrix}
\frac{1}{100} & -1 & 1 \\
-1 & 100 & 1
\end{bmatrix}
\]

\[\kappa \approx 245\]

• Standard scaling:

\[
\begin{align*}
\begin{bmatrix}
\frac{1}{100}\times^{box} \\
\frac{1}{100}\times^{chess}
\end{bmatrix} + \begin{bmatrix}
1 \\
100
\end{bmatrix} & \leq 100 \\
\frac{1}{100}\times^{box} & \leq b^{box} \\
\frac{1}{100}\times^{chess} & \leq b^{chess}
\end{align*}
\]

\[
\begin{bmatrix}
-1 & -1 & \frac{1}{100} \\
-100 & 100 & 1
\end{bmatrix}
\]

\[\kappa \approx 245\]

Same!
Example - Continued

• Unscaled:

\[ \begin{align*}
\chi_{\text{box}} + \chi_{\text{chess}} & \leq 100 \\
\chi_{\text{box}} & \leq 100b_{\text{box}} \\
\chi_{\text{chess}} & \leq 100b_{\text{chess}}
\end{align*} \]

\[ \begin{bmatrix}
-\frac{1}{100} & -\frac{1}{100} & \frac{1}{100} \\
\frac{1}{100} & 0 & \frac{1}{100} \\
-1 & 0 & 1
\end{bmatrix} \quad \kappa \approx 245 \]

• Standard scaling:

\[ \begin{align*}
\frac{1}{100}\chi_{\text{box}} + \chi_{\text{chess}} & \leq 100 \\
\frac{1}{100}\chi_{\text{box}} & \leq b_{\text{box}} \\
\frac{1}{100}\chi_{\text{chess}} & \leq b_{\text{chess}}
\end{align*} \]

\[ \begin{bmatrix}
-1 & -1 & \frac{1}{100} \\
0 & -100 & 1 \\
0 & 100 & 1
\end{bmatrix} \quad \kappa \approx 245 \quad \text{Same!} \]

• “Best” scaling

\[ \begin{align*}
\chi_{\text{box}} + \chi_{\text{chess}} & \leq 1 \\
\chi_{\text{box}} & \leq b_{\text{box}} \\
\chi_{\text{chess}} & \leq b_{\text{chess}}
\end{align*} \]

\[ \begin{bmatrix}
-1 & -1 & 1 \\
-1 & 0 & 1 \\
0 & 1 & 1
\end{bmatrix} \quad \kappa \approx 25 \quad \text{Best!} \]
Learning To Scale
Learning to Scale

- New approach: Learn to Scale
  - Try each scaling method: Standard and Curtis-Reid.
  - One fixed method not always best.
  - Use an ML model based on linear regression to predict which method will result in the smallest attention level.
  - Features drawn from coefficient distributions.
- Trained on more than 1000 MIP instances
- Validation outcome:
Learning to Scale

• New approach: Learn to Scale
  • Try each scaling method: Standard and Curtis-Reid.
  • One fixed method not always best.
  • Use an ML model based on linear regression to predict which method will result in the smallest attention level.
  • Features drawn from coefficient distributions.

• Trained on more than 1000 MIP instances.

• Validation outcome:
Learning to Scale

- New approach: Learn to Scale
  - Try each scaling method: Standard and Curtis-Reid.
  - One fixed method not always best.
  - Use an ML model based on linear regression to predict which method will result in the smallest attention level.
  - Features drawn from coefficient distributions.
- Trained on more than 1000 numerically challenging instances.
- Validation outcome:
Learning to Scale

• New approach: Learn to Scale
  • Try each scaling method: Standard and Curtis-Reid.
  • One fixed method not always best.
  • Use an ML model based on linear regression to predict which method will result in the smallest attention level.
  • Features drawn from coefficient distributions.

• Trained on more than 1000 numerically challenging instances.

• Validation outcome:
Computational Results

• On our set of Numerically Challenging instances:
  • Tremendous improvements in all stability criteria

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Fails</td>
<td>-64%</td>
<td>Primal Fails</td>
<td>-67%</td>
<td>Singular Inverts</td>
</tr>
<tr>
<td>Infeasibilities</td>
<td>-26%</td>
<td>Inconsistencies</td>
<td>-35%</td>
<td>Violated Sols</td>
</tr>
<tr>
<td>Kappa Stable</td>
<td>+148%</td>
<td>Kappa Max</td>
<td>-979%</td>
<td>Attn. Level</td>
</tr>
</tbody>
</table>

• ≈10% performance improvements on our simplex test sets.

• On our MIP Performance set: performance-neutral

• New control: **AUTOSCALE**
  • Setting **SCALING** control will override **AUTOSCALE**.
Conclusion

Machine Learning for general MIP

Works best:

- For categorical decisions,
- With suitable features,
- With established label that connects to the features,
- When you are not competing against a rule that has been finetuned over decades.

Learning to scale:

- ML module to predict scaling method for MIP and LP solving
- Drastically improves numerical stability
- Does not deteriorate performance
- One of many recent components in Xpress that address numeric stability
Sneak peek: Learning the Attention Level

- A-priori prediction: Will the current solve lead to a high attention level?
  - Called after initial LP relaxation
  - Prints a warning for the user

- Similar features as in “Learning to scale”
  - Additionally use conditioning of matrix w.r.t. right-hand side

- Uses random forest
  - Accuracy > 95%
  - False negative rate < 2%
  - Threshold biased towards false positives

- To be released with the next major Xpress version
Thank You!

Timo Berthold