Machine Learning in MIP Solvers

Timo Berthold
TU Berlin, FICO, MODAL

image source: DALL-E

Agenda

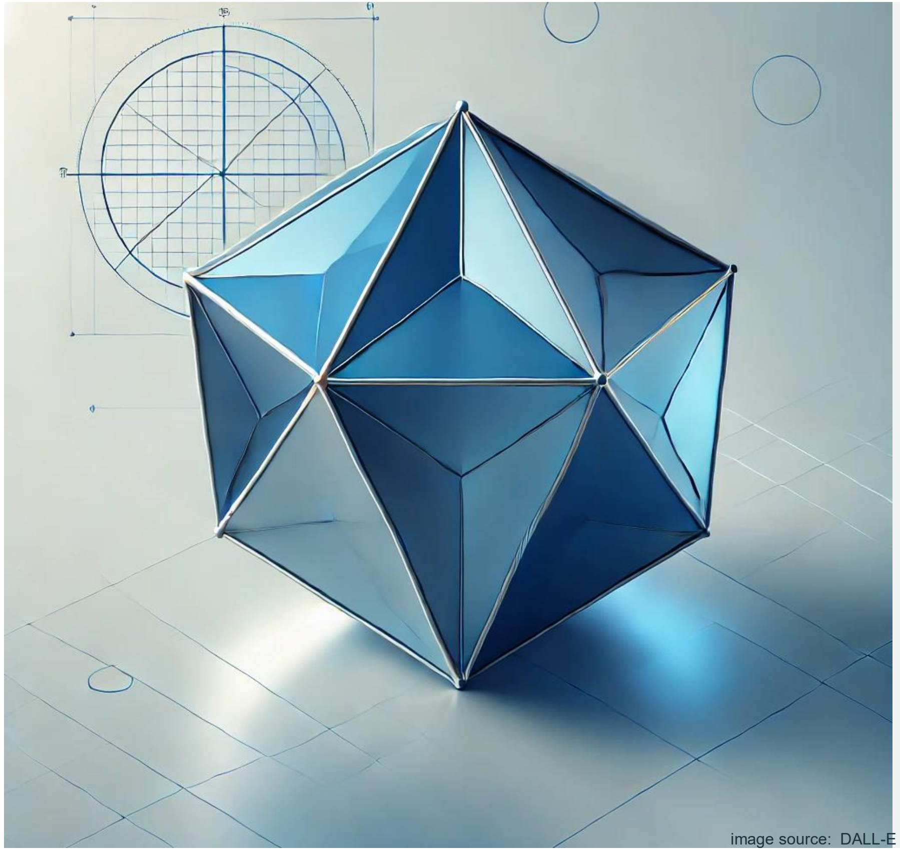


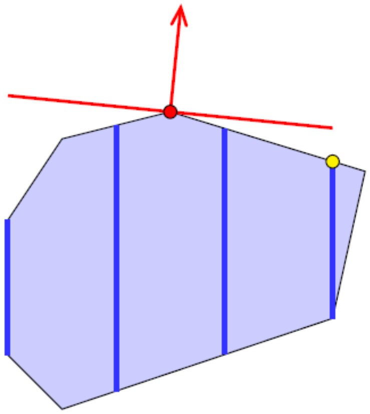
image source: DALL-E

1. What do I mean by „Machine Learning in MIP Solvers"?
2. Learning to Scale
3. Learning the Attention Level
4. Learning to Use Local Cuts
5. Learning to Select Cuts

What do I mean by ML in MIP solvers?

What do I mean by MIP?

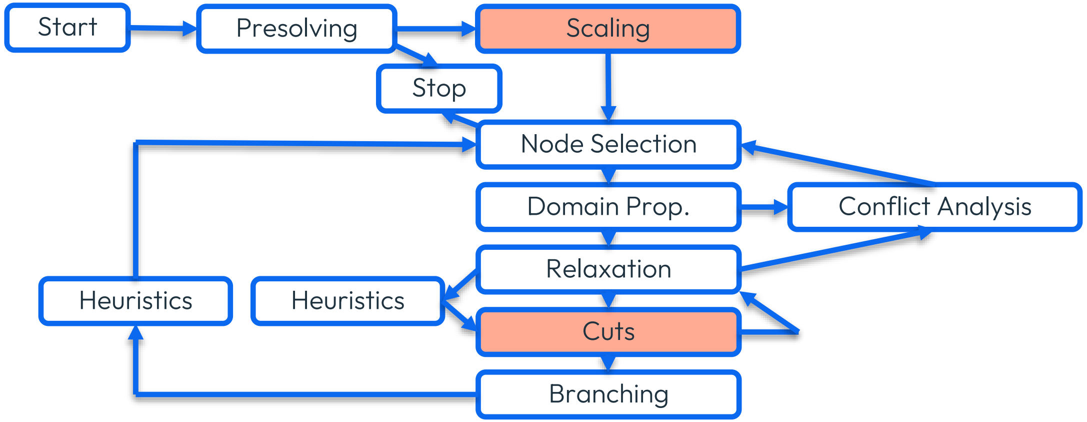
- The obvious:
  - maximizing a linear objective function over a set of linear constraints
  - Some or all variables have to take integral values, some or all are bounded
  - $$\max c^T x$$
    $$s.t.\ Ax \leq b$$
    $$x \in \mathbb{Z}^I \times \mathbb{R}^{N \setminus I}$$
    $$\ell \leq x \leq u$$



- More specifically:
  - General MIP, not specific problem classes
  - Extremely heterogeneous test sets, existing solvers highly optimized
    - single digit improvements on MIPLIB (or internal client sets) are a celebrated success

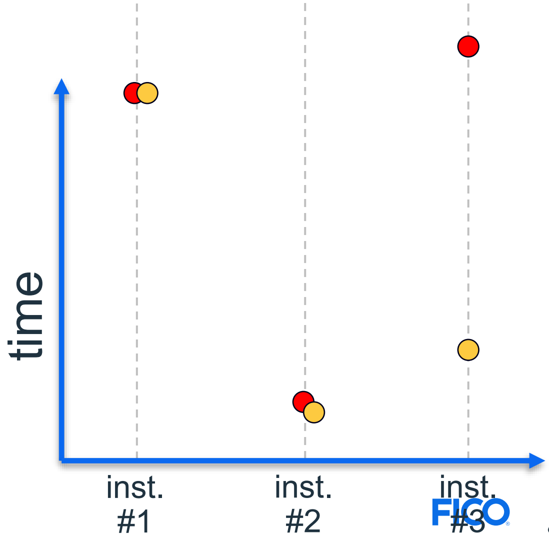
What do I mean by MIP solvers?

- The obvious:
  - A piece of software, commercial or academic, that can solve general-purpose MIPs to proven optimality
  - Does not require (or even allow) any application-specific input
- More specifically:
  - In this presentation: FICO Xpress and SCIP



What do I mean by Machine Learning?

- Most of the cases: Learning a (binary) decision on which action to take inside a MIP solver
  - Classification (but really: regression)
- Offline, supervised learning
- Trained on general, heterogeneous test sets, not individually trained for application
- Easy-to-evaluate, ideally interpretable model
- Use regression for „classification" w.r.t. a continuous measure
  - Labels are improvement factors, not „this one worked better"
  - Draws are a typical case
  - Misclassifications not too bad on instances where methods perform similar
  - But fatal when there is a huge difference



Learning To Scale

Proc. of AAAI 2021, joint work with Gregor Hendel

FICO

Huge Activity in Machine Learning for Mathematical Optimization

Ding et al.	Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction	2019
Bertsimas & Stellato	Online Mixed-Integer Optimization in Milliseconds	2019
Fischetti & Fraccaro	Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks	2018
Misra et al.	Learning for constrained optimization: Identifying optimal active constraint sets	2018
Bertsimas & Stellato	The Voice of Optimization	2018
Tang et al.	Reinforcement Learning for Integer Programming: Learning to Cut	2019
Baltean-Lugojan et al.	Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks	2018
Etheve et al.	Reinforcement Learning for Variable Selection in a Branch and Bound Algorithm	2020
Zarpellon et al.	Parameterizing Branch-and-Bound Search Trees to Learn Branching Policies	2020
Yang et al.	Learning Generalized Strong Branching for Set Covering, Set Packing, and 0-1 Knapsack Problems	2020
Song et al.	Learning to Search via Retrospective Imitation	2019
Gasse et al.	Exact Combinatorial Optimization with Graph Convolutional Neural Networks	2019
Lee et al.	Learning to Branch: Accelerating Resource Allocation in Wireless Networks	2019
Hansknecht et al.	Cuts, Primal Heuristics, and Learning to Branch for the Time-Dependent Traveling Salesman Problem	2018
Balcan et al.	Learning to branch	2018
Václavík et al.	Accelerating the branch-and-price algorithm using machine learning	2018
Hottung et al.	Deep Learning Assisted Heuristic Tree Search for the Container Pre-marshalling Problem	2017
Lodi & Zarpellon	On learning and branching: a survey	2017
Alvarez et al.	A Machine Learning-Based Approximation of Strong Branching	2017
Alvarez et al.	Online Learning for Strong Branching Approximation in Branch-and-Bound	2016
Khalil et al.	Learning to branch in mixed integer programming	2016
Khalil	Machine Learning for Integer Programming	2016
He et al.	Learning to Search in Branch and Bound Algorithms	2014
Alvarez et al.	A Supervised Machine Learning Approach to Variable Branching in Branch-And-Bound	2014
Di Liberto et al.	Dynamic Approach for Switching Heuristics	2013
Sabharwal et al.	Guiding Combinatorial Optimization with UCT	2012

Khalil et al.	Learning to Run Heuristics in Tree Search	2017
Hutter et al.	Algorithm Runtime Prediction: Methods & Evaluation	2012
Hutter et al.	Automated Configuration of Mixed Integer Programming Solvers	2010
Ferber et al.	MIPaaL: Mixed Integer Program as a Layer	2019
Wilder et al.	End to end learning and optimization on graphs	2019
Wang et al.	SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver	2019
Wilder et al.	Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization	2018
Elmachtoub & Grigas	Smart "Predict, then Optimize"	2017
Kool et al.	Attention, Learn to Solve Routing Problems!	2018
Li et al.	Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search	2018
Dai et al.	Learning Combinatorial Optimization Algorithms over Graphs	2017
Bello et al.	Neural Combinatorial Optimization with Reinforcement Learning	2016
Bowly et al.	Generation techniques for linear programming instances with controllable properties	2017
Bowly	Stress testing mixed integer programming solvers through new test instance generation methods	2019
François et al.	How to Evaluate Machine Learning Approaches for Combinatorial Optimization: Application to the Travelling Salesman Problem	2019
Fischetti et al.	Learning MILP Resolution Outcomes Before Reaching Time-Limit	2018
Kuhlmann	Learning to steer nonlinear interior-point methods	2019
Kruber et al.	Learning when to use a decomposition	2018
Bengio et al.	Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon	2018
Hendel	Adaptive Large Neighborhood Search for Mixed Integer Programming	2018
Bonami et al.	Learning a Classification of Mixed-Integer Quadratic Programming Problems	2017
Amos & Kolter	OptNet: Differentiable Optimization as a Layer in Neural Networks	2017
Schweidtmann & Mitsos	Global Deterministic Optimization with Artificial Neural Networks Embedded	2018
Sculley	Large Scale Learning To Rank	2020
Song et al.	A General Large Neighborhood Search Framework for Solving Integer Programs	2020

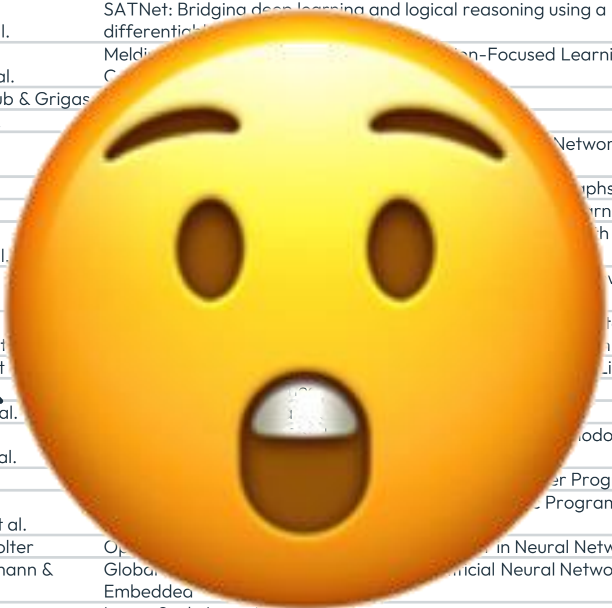
Huge Activity in Machine Learning for Mathematical Optimization

Ding et al.	Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction	2019
Bertsimas & Stellato	Online Mix	2019
Fischetti & Fraccaro	Mac	
Misra et al.		
Bertsimas & S		
Tang et al.		
Baltean-Lug		
Etheve et al.		
Zarpellon et al.		
Yang et al.	Lea and 0-1 Kn	0
Song et al.	Learning to Search via	
Gasse et al.	Exact Combinatorial Optimization with Graph Convolutional Neur Networks	
Lee et al.	Learning to Branch: Accelerating Resource Allocation in Wireless Networks	2019
Hansknecht et al.	Cuts, Primal Heuristics, and Learning to Branch for the Time-Dependent Traveling Salesman Problem	2018
Balcan et al.	Learning to branch	2018
Václavík et al.	Accelerating the branch-and-price algorithm using machine learning	2018
Hottung et al.	Deep Learning Assisted Heuristic Tree Search for the Container Pre-marshalling Problem	2017
Lodi & Zarpellon	On learning and branching: a survey	2017
Alvarez et al.	A Machine Learning-Based Approximation of Strong Branching	2017
Alvarez et al.	Online Learning for Strong Branching Approximation in Branch-and-Bound	2016
Khalil et al.	Learning to branch in mixed integer programming	2016
Khalil	Machine Learning for Integer Programming	2016
He et al.	Learning to Search in Branch and Bound Algorithms	2014
Alvarez et al.	A Supervised Machine Learning Approach to Variable Branching in Branch-And-Bound	2014
Di Liberto et al.	Dynamic Approach for Switching Heuristics	2013
Sabharwal et al.	Guiding Combinatorial Optimization with UCT	2012

Khalil et al.	Learning to Run Heuristics in Tree Search	2017
Hutter et al.	Algorithm Runtime Prediction: Methods & Evaluation	2012
Hutter et al.	Automated Configuration of Mixed Integer Programming Solvers	2010
Ferber et al.	MIPaaL: Mixed Integer Program as a Layer	2019
Wilder et al.	End to end learning and optimization on graphs	2019
ang et al.	SATNet: Bridging de learning and logical reasoning using a differenti	2019
r et al.	Meldi n-Focused Learning for C	2018
chtoub & Grigas		2017
et al.		2018
al.	Networks and	2018
t al.	phs	2017
llo et al.	rning	2016
Bowly et al.	h	2017
Bowly	w test	2019
çois et	torial	2019
ti et	imit	2018
Kuhlm		2019
Kruber et al.		2018
Bengio et al.	odological	2018
Hendel	er Programming	2018
Bonami et al.	Programming	2017
Amos & Kolter	O in Neural Networks	2017
Schweidtmann & Mitsos	Globa Embedded icial Neural Networks	2018
Sculley	Large Scale Learning To Rank	2020
Song et al.	A General Large Neighborhood Search Framework for Solving Integer Programs	2020

Only few of these implemented in general purpose MIP solvers!
(and activated by default)

(at the time)



FICO

Huge Activity in Machine Learning for Mathematical Optimization

Ding et al.	Accelerating Primal Solution Findings for Mixed Integer Programs Based	2019
Bertsimas & S		2019
Fisch	e	20
Misra e		201
Bertsim		201
Tang et al.	Learning to Cut	2019
Baltean-Lugojan et al.	Selecting cutting p atic semidefinite outer-approximation via trained neural networks	2018
Etheve et al.	Reinforcement Learning for Vari Selection in a Branch and Bound Algorithm	2020
Zarpellon et al.	Parameterizing Branch-and-Bound Search T Policies	
Yang et al.	Learning Generalized Strong Branch and 0-1 Knapsack Problems	
Song et al.	Learning to Search via Retrosp	
Gasse et al.	Exact Combinatorial Optimiz Networks	
Lee et al.	Learning to Branch: Accele Networks	
Hansknecht et al.	Cuts, Primal Heuristics, a Traveling Salesman Prob	
Balcan et al.	Learning to branch	
Václavík et al.	Accelerating the branch	
Hottung et al.	Deep Learning Assisted marshalling Problem	
Lodi & Zarpellon	On learning and branchin	
Alvarez et al.	A Machine Learning-Base	
Alvarez et al.	Online Learning for Strong Bound	
Khalil et al.	Learning to branch in mixed i	
Khalil	Machine Learning for Integer P	
He et al.	Learning to Search in Branch and	
Alvarez et al.	A Supervised Machine Learning App Branch-And-Bound	
Di Liberto et al.	Dynamic Approach for Switching Heuristics	013
Sabharwal et al.	Guiding Combinatorial Optimization with UCT	2012

Khalil et al.	Learning to Run Heuristics in Tree Search	2017
Hutter et	aluation	2012
H	mming Solvers	2010
		2019
		2019
	using a	2019
Wilde	sion-Focused Learning for	2018
Elmachtoub & Grigas	Smar Optimize"	2017
Kool et al.	Attention, Learn to Solve Routing Problems!	2018
Li et al.	Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search	2018
Dai et	Learning Combina	17
Bello et al.	Neural Com	
wly et al.	Generation fe controllable p	2017
	Stress testing mixed integ gh new test instance gene ion methods	2019
t al.	How to Evaluate Machine Learning Approaches for Combinatorial Optimization: Application to the Travelling Salesman Problem	2019
al.	Learning MILP Resolution Outcomes Before Reaching Time-Limit	2018
	Learning to steer nonlinear interior-point methods	2019
	Learning when to use a decomposition	2018
	Machine Learning for Combinatorial Optimization: a Methodological To d'Ho n	
	Adaptive ge	
	Learning a Cla Problem	
l.	OptNe	
ter	Global De	
ann &	Embedde	
	Large Scale Learn	020
g et al.	A General Large Neighbo ng Integer Programs	2020

Huge variability, no ground truth

Sophisticated rules already in place

Heterogenity…

We don't even know good features

Huge Activity in Machine Learning for Mathematical Optimization

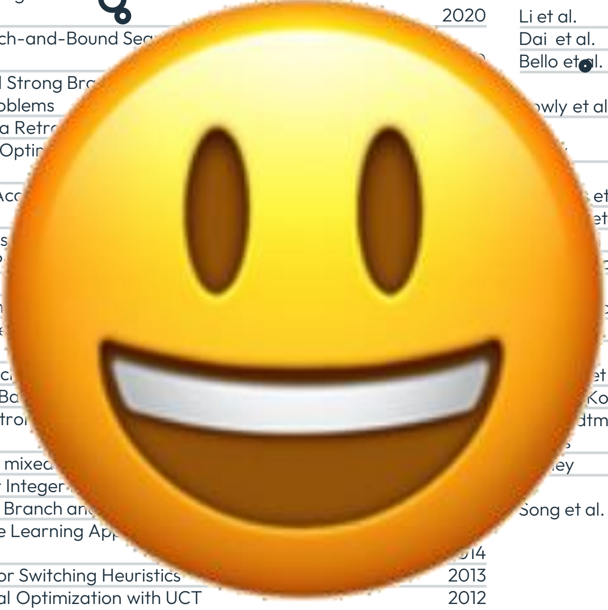
Ding et al.	Accelerating Primal Solution Findings for Mixed Integer Programs Based ...	2019
Bertsimas & S...		2019
Fisch...	...e	2018
Misra e...		2018
Bertsim...		2018
Tang et al.	...: Learning to Cut	2019
Baltean-Lugojan et al.	Selecting cutting p... ...atic semidefinite outer-approximation via trained neural networks	2018
Etheve et al.	Reinforcement Learning for Vari... Selection in a Branch and Bound Algorithm	2020
Zarpellon et al.	Parameterizing Branch-and-Bound Sea... Policies	
Yang et al.	Learning Generalized Strong Br... and 0-1 Knapsack Problems	
Song et al.	Learning to Search via Retr...	
Gasse et al.	Exact Combinatorial Optim... Networks	
Lee et al.	Learning to Branch: Acc... Networks	
Hansknecht et al.	Cuts, Primal Heuristics... Traveling Salesman P...	
Balcan et al.	Learning to branch	
Václavík et al.	Accelerating the bran...	
Hottung et al.	Deep Learning Assiste... marshalling Problem	
Lodi & Zarpellon	On learning and branc...	
Alvarez et al.	A Machine Learning-Ba...	
Alvarez et al.	Online Learning for Stro... Bound	
Khalil et al.	Learning to branch in mixe...	
Khalil	Machine Learning for Integer...	
He et al.	Learning to Search in Branch an...	
Alvarez et al.	A Supervised Machine Learning App... Branch-And-Bound	...14
Di Liberto et al.	Dynamic Approach for Switching Heuristics	2013
Sabharwal et al.	Guiding Combinatorial Optimization with UCT	2012

Khalil et al.	Learning to Run Heuristics in Tree Search	2017
Hutter et al.	Algorithm Runtime Prediction: Methods & Evaluation	2012
Hutter et al.	Automated Configuration of Mixed Integer Programming Solvers	2010
Ferber et al.	MIPaaL: Mixed Integer Program as a Layer	2019
Wilder et al.	...raphs	2019
	...ning using a	2019
	...sed Learning for	2018
Elmachto...		2017
Kool et al.	Attention, Learn to Solve Routing Problems!	2018
Li et al.	Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search	2018
Dai et al.	Learning Combinat... ...ms over Graphs	2017
Bello et al.	Neural C... ...cement Learning	2016
...wly et al.	Gen... con... ...nces with	2017
	...ough new test	2019
... et al.	...ombinatorial ...an Problem	2019
...et al.	L... ...eaching Time-Limit	2018
	...ethods	2019
...al.	...ning w... ...tion	2018
...al.	Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon	2018
	Adaptive Large Neighborhood Search for Mixed Integer Programming	2018
...et al.	Learning a Classification of Mixed-Integer Quadratic Programming Problems	2017
...Kolter	OptNet: Differentiable Opti...	2017
...dtmann &	G... Dete... Embedde...	2018
...ey	Lar...	2020
Song et al.	A ... Prog...	2020

Scaling!!!

(definition coming soon)

Either-or decision



We know meaningful features

No good rule in place

(at the time)

Numeric Stability (Somewhat Outdated Slide)

- Numeric stability of MIP solving is a crucial topic in many OR applications
  - 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
    - Numerics IV: Learning to pay attention
    - Numerics V: Integrality – When Being Close Enough is not Always Good Enough
- Real-life applications often complex and numerically challenging to handle:
  - More than half of client problems seen in the past year (2019) had some numeric issues
  - After performance, numeric failures are the most common support request
    - Unexpected solution status
    - Inconsistent results
    - Performance issues (e.g., simplex cycling)

See Ambros' lecture from Thursday

This number has dropped significantly due to the work presented here.

FICO

Information on numeric stability

- (some) MIP solvers provide numeric analysis tools
- A priori: spread of matrix, objective, rhs coefficients

Coefficient range	original	solved

```
Coefficients    [min,max] : [ 2.00e-06,  2.34e+02] / [ 1.25e-01,  1.67e+00]
RHS and bounds  [min,max] : [ 1.67e-01,  9.23e+03] / [ 1.67e-01,  8.21e+02]
Objective       [min,max] : [ 2.00e-06,  2.34e+02] / [ 2.00e-06,  2.34e+02]
```

compares effects of presolving AND scaling

- A posteriori: report on numeric issues 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)
```

numeric information raises awareness!

Adapt models or use non-default controls to prevent issues

We'll need this soon

Condition Number & Attention Level

- The condition number κ of a matrix A provides a bound on how much a small change in b can affect x, when $Ax = b$

#errorpropagation

- For a square, invertible matrix A

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

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

```
Nodes kappa stable      :       3757 (ratio: 0.0051)
Nodes kappa suspicious  :       8476 (ratio: 0.0115)
Nodes kappa unstable    :     723171 (ratio: 0.9831)
Nodes kappa ill-posed   :        193 (ratio: 0.0003)
Largest kappa seen      : 4.959805e+14
Kappa attention level   : 0.2953
```

- Summarized in a single attention level from 0.0 (all stable) to 1.0 (anything goes)
  - Non-default feature: Expensive
- One purpose of scaling is to reduce the condition numbers (and attention level)

What is Scaling?

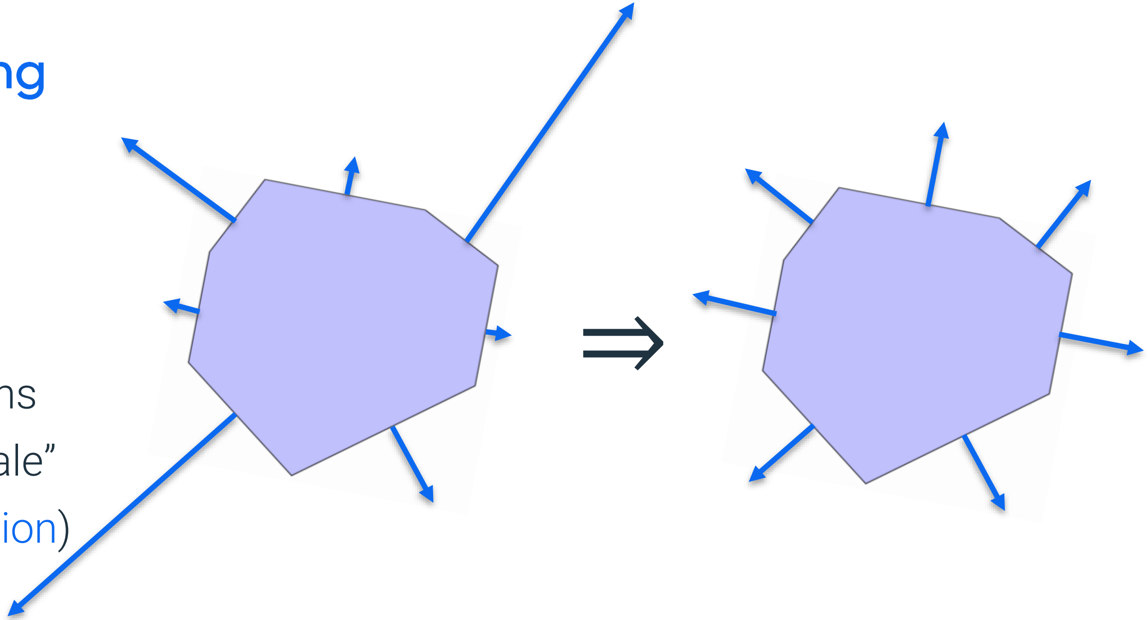
- LP Scaling refers to the (iterative) multiplication of rows and columns by scalars
  - to reduce the absolute magnitude of nonzero coefficients in matrix, rhs and objective
  - to reduce the relative difference of nonzero coefficients in matrix, rhs and objective
- Scaling is a widely used preconditioning technique, used by various kinds of algorithms
  - to improve the numeric behavior of the algorithms
  - to reduce the condition number of basis matrices
    - to reduce error propagation
  - to reduce the number of iterations required to solve the problem

Scaling in Linear Programming

- Basic Linear Program (LP):

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \le b \end{aligned}$$

- Scaling multiplies rows and columns
  - to bring coefficients “on one scale”
  - Typically, close to 1 (normalization)



$$\begin{aligned} \max \quad & (cC)^T (C^{-1}x) \\ \text{s.t.} \quad & (RAC)(C^{-1}x) \le Rb \end{aligned}$$

$$c' = cC,\ A' = RAC,\ b' = Rb$$
$$x' = C^{-1}x$$

$$\begin{aligned} \max \quad & c'^T x' \\ \text{s.t.} \quad & A'x' \le b' \end{aligned}$$

R: Square diagonal matrix of row scalars
C: Square diagonal matrix of column scalars

FICO

Two scaling methods

Equilibrium scaling (1975):

- Divide rows by largest coefficient
- Then divide columns by largest coefficient
- Potentially iterate

Curtis-Reid (1972):

- Minimize least-squares deviation from 1:
- $\min \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\log R_{ii} C_{jj} \, |A_{ij}|\right)^2$
- Positive semidefinite, unconstrained → conjugate gradient
- Binary logarithm, round scaling factors to powers of two

Example

- We want to set up a 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:

$$
\begin{array}{ll}
max & 5x^{box} + 10x^{chess} - 30b^{box} - 500b^{chess} \\
s.t. & x^{box} + x^{chess} \leq 100 \\
 & x^{box} \leq 100b^{box} \\
 & x^{chess} \leq 100b^{chess} \\
 & b^{box}, b^{chess} \in \{0,1\}
\end{array}
$$

- Coefficient matrix:

  … with a potential basis matrix

$$
\begin{bmatrix}
1 & 1 & & \\
1 & & -100 & \\
 & 1 & & -100
\end{bmatrix}
$$

Example

Basis inverse

- Unscaled:

$$\begin{array}{lcl} x^{box} + x^{chess} & \leq & 100 \\ x^{box} - 100b^{box} & \leq & 0 \\ x^{chess} - 100b^{chess} & \leq & 0 \end{array} \qquad \begin{bmatrix} -\frac{1}{100} & -\frac{1}{100} & \frac{1}{100} \\ & -1 & 1 \\ & 1 & \end{bmatrix} \qquad \kappa \approx 245$$

- Equilibrium scaling:

$$\begin{array}{lcl} x^{box} + x^{chess} & \leq & 100 \\ \frac{1}{100}x^{box} - b^{box} & \leq & 0 \\ \frac{1}{100}x^{chess} - b^{chess} & \leq & 0 \end{array} \qquad \begin{bmatrix} -1 & -1 & \frac{1}{100} \\ & -100 & 1 \\ & 100 & \end{bmatrix} \qquad \kappa \approx 245$$

Same!

- Curtis-Reid scaling;

$$\begin{array}{lcl} x^{box} + x^{chess} & \leq & 1 \\ x^{box} - b^{box} & \leq & 0 \\ x^{chess} - b^{chess} & \leq & 0 \end{array} \qquad \begin{bmatrix} -1 & -1 & 1 \\ & -1 & 1 \\ & 1 & \end{bmatrix} \qquad \kappa \approx 25$$

Best!

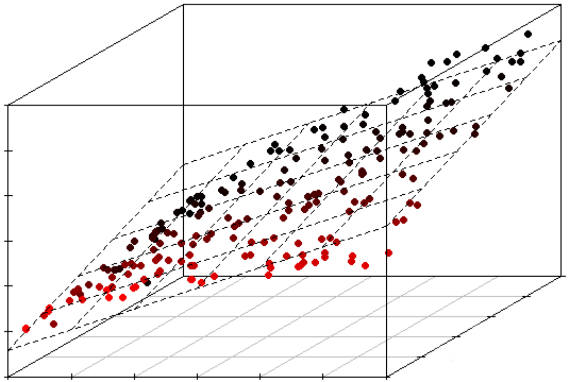
Learning to Scale

- One fixed method not always best
- New approach: Learn to Scale
  - Try each scaling method: Equilibrium and Curtis-Reid
  - Try to predict which method will result in the smaller attention level
  - Or rather: The factor by which the attention level differs (label)
- Features drawn from coefficient distributions
  - Coefficient spread: $\gamma \coloneqq \log \frac{\max_{i,j} |A_{ij}|}{\min_{i,j} |A_{ij}|}$
  - Use $\gamma_{Equi} - \gamma_{Curtis}$ as a feature
  - Same procedure to get features for objective spread and right-hand side spread

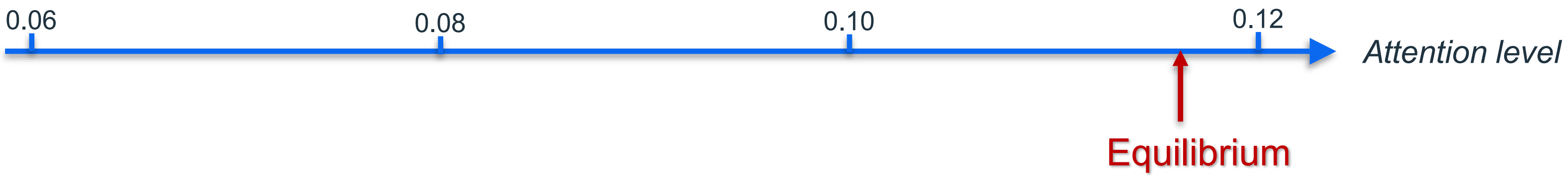
FICO

Learning to Scale

- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances



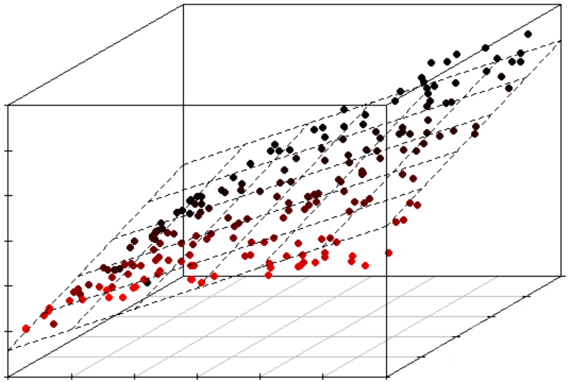
- Validation outcome:



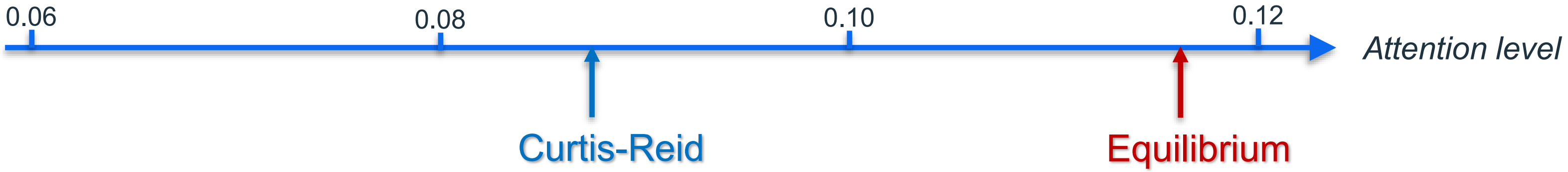
FICO

Learning to Scale

- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances



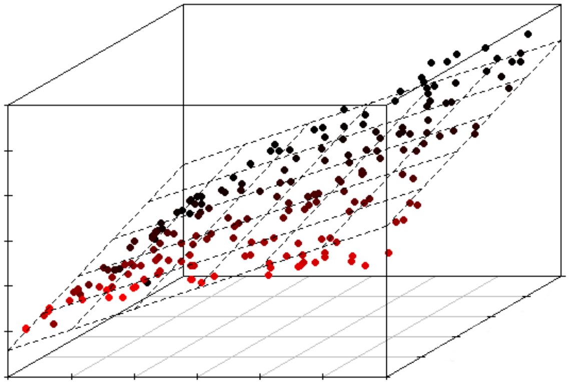
- Validation outcome:



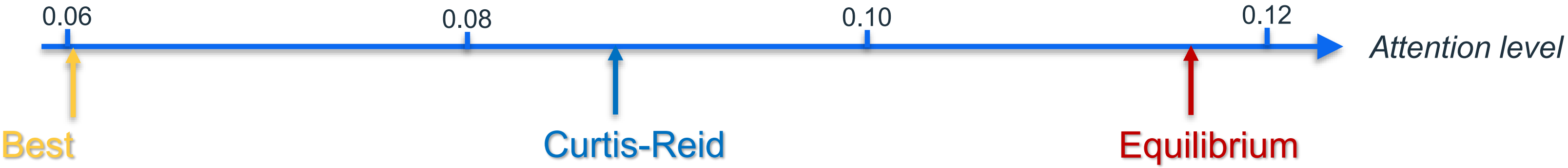
FICO

Learning to Scale

- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances

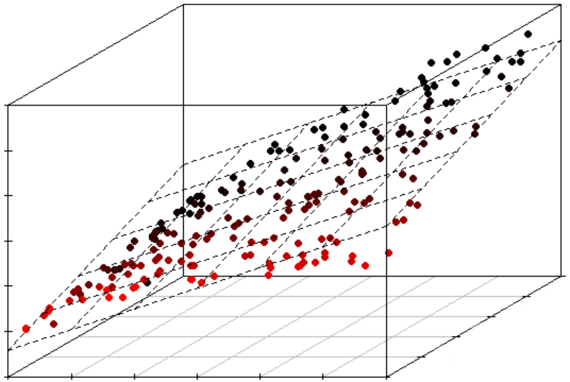


- Validation outcome:

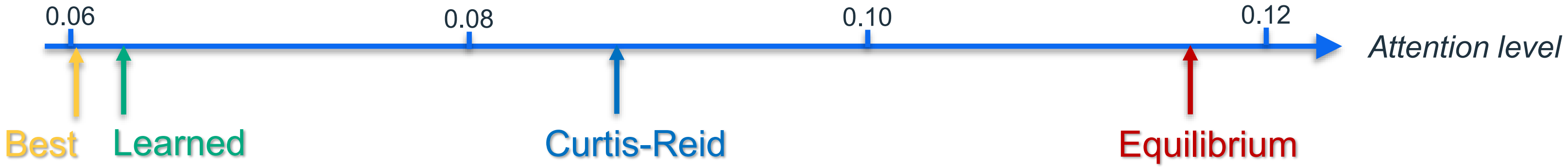


Learning to Scale

- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances



- Validation outcome:



Computational Results

- On our set of numerically Challenging instances:
  - Tremendous improvements in all stability criteria

Dual Fails	-64%	Primal Fails	-67%	Singular Inverts	-48%
Infeasibilities	-26%	Inconsistencies	-35%	Violated Sols	-12%
Kappa Stable	+148%	Max. Condition	-979%	**Attn. Level**	**-88%**

- ≈10% performance improvements on Xpress simplex test sets

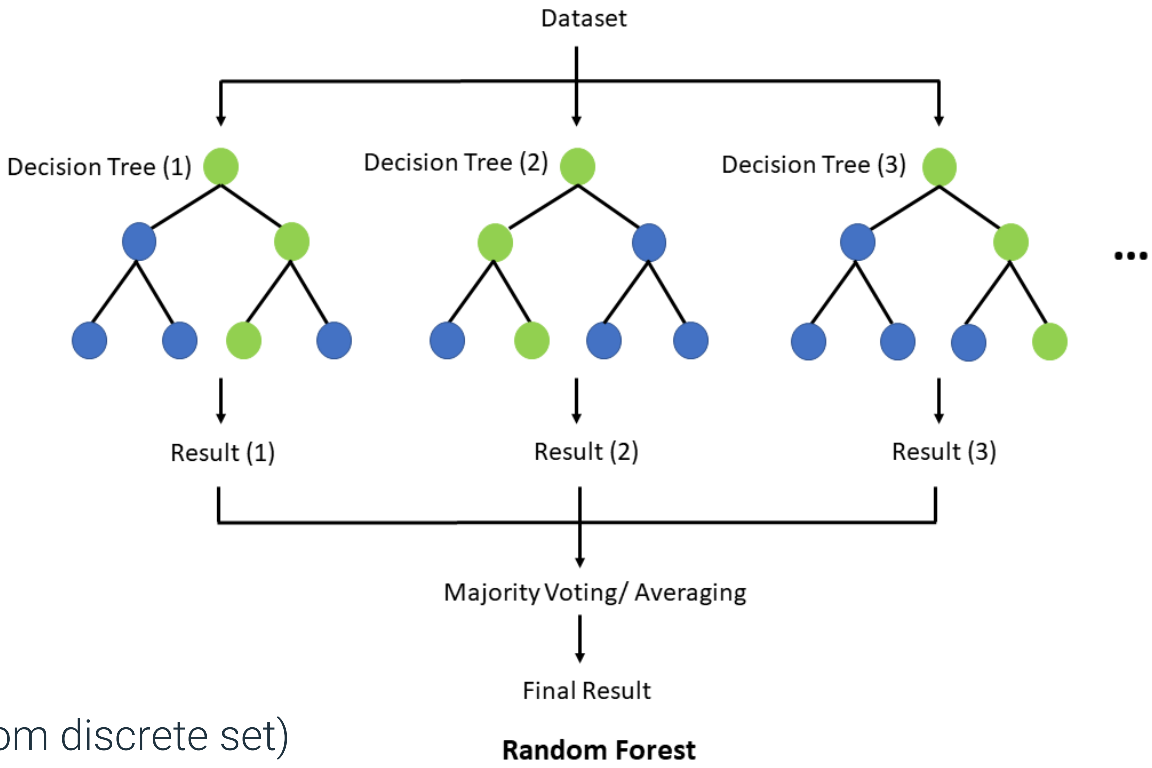
Learning The Attention Level

Motivation

- Attention level: indicator of numeric sensitivity of a problem:
  - 0.0 (very stable) to 1.0 (anything goes)
  - Computed a posteriori ☹
  - uses condition numbers of LP bases (expensive!) ☹
    - Most comprehensive and comprehensible numerics analysis tool deactivated by default
- ML-based prediction: Might the current solve lead to a high attention level?
  - Print a warning for the user
- Learning the attention level:
  - A priori ☺: After the initial LP relaxation
  - Cheap ☺: Similar features as in “Learning to scale”
    - Additionally use conditioning of matrix w.r.t. right-hand side

FICO

Regression forests



- Supervised learning
- Ensemble learning
- No interaction between trees
- Each tree predicts a value (from discrete set)
- Overall prediction: average

FICO

Results: Confusion matrix

- Our prediction uses a regression forest

Everything okay

This is critical, warning missed!

	Is low	Is high
Predicted low	1393	2
Predicted high	35	128

This is not great, unneccesary warning

This is why we do this: warn user!

- Accuracy ~ 98%
- False negative rate ~ 1.5%

Learning the Attention Level

```
Coefficient range                     original                 solved
  Coefficients   [min,max] : [ 2.82e-04,  6.23e+03] / [ 1.56e-02,  3.26e+01]
  RHS and bounds [min,max] : [ 7.98e-01,  1.04e+05] / [ 2.73e-01,  1.43e+05]
  Objective      [min,max] : [ 1.00e+00,  3.12e+06] / [ 3.12e-02,  4.00e+06]
Autoscaling applied Curtis-Reid scaling
…
Final LP objective                     : 1.074622443761501e+08
  Max primal violation      (abs/rel) : 8.292e-13 / 8.292e-13
  Max dual violation        (abs/rel) : 4.849e-08 / 1.825e-08
  Max complementarity viol. (abs/rel) :       0.0 /       0.0
High attention level predicted from matrix features
```

- Might want to abort solution process early to fix numerics
- Might want to do a run with enabled attention level computation
  - Pay more attention to other numeric statistics, unusual solver behavior, ...
- Precise prediction can be queried as attribute: PREDICTEDATTLEVEL

FICO

Learning To Use Local Cuts

2nd round of review for Mathematical Programming Comp.,
joint work with Matteo Francobaldi & Gregor Hendel

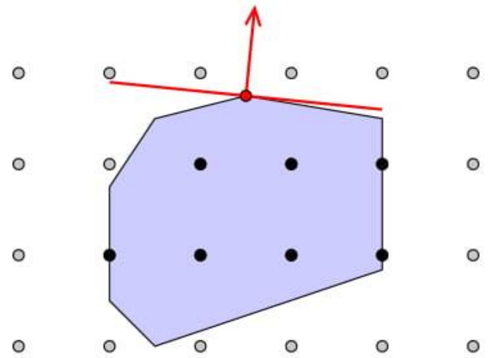
FICO

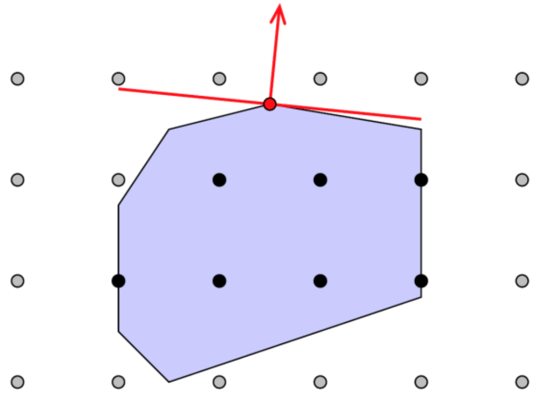
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$
$$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 & & \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound





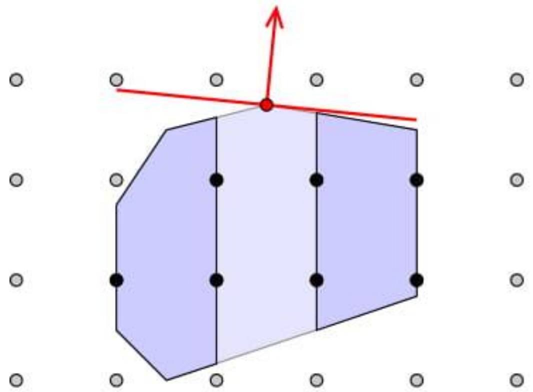
Solving Integer Optimization Problems

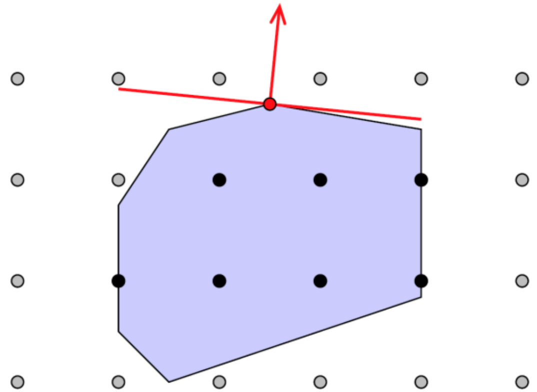
- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ \ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ \ x + 3y &\leq 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound





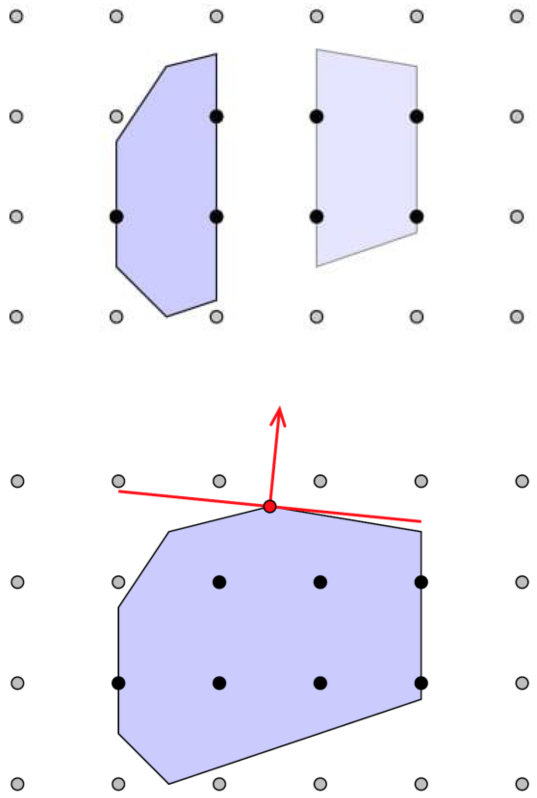
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

$$\begin{array}{rlrl} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 & & \end{array}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound

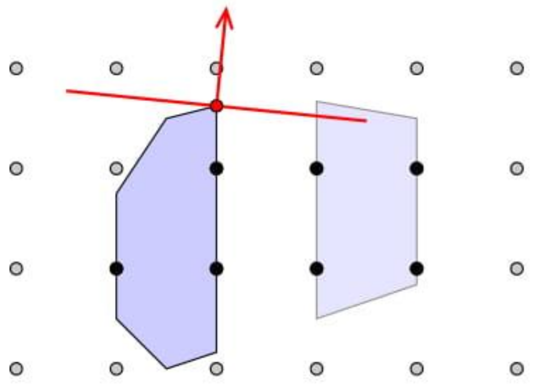


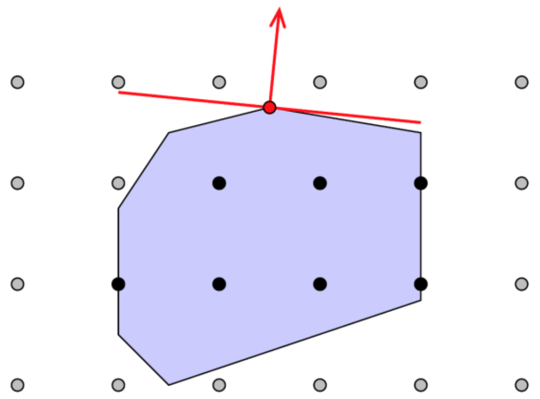
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$
$$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound



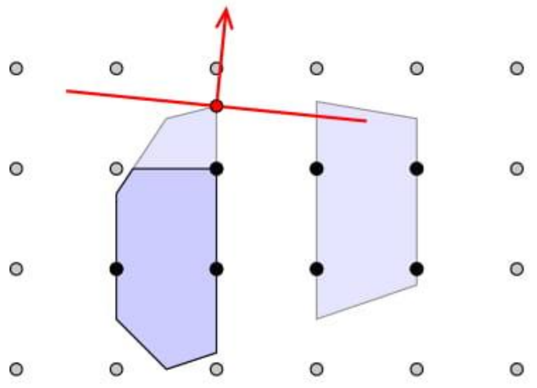


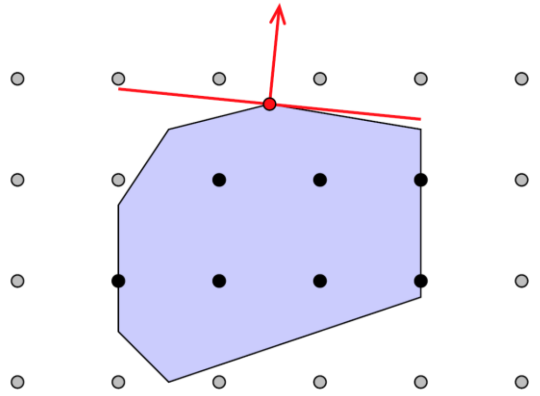
Solving Integer Optimization Problems

- Example MIP:

$$
\begin{array}{ll}
\max x + 5y & \\
\quad x - 3y \leq 3.6 & -x - 3y \leq -7.2 \\
-\quad x - 3y \leq 0.5 & 0 \leq x \leq 3 \\
-2x + 3y \leq 0.8 & x, y \in \mathbb{Z} \\
-\quad x + 3y \leq 4.2 &
\end{array}
$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound





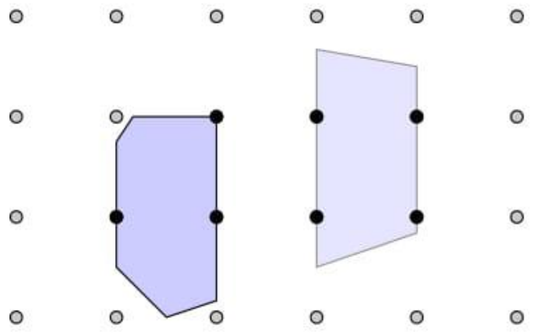
Solving Integer Optimization Problems

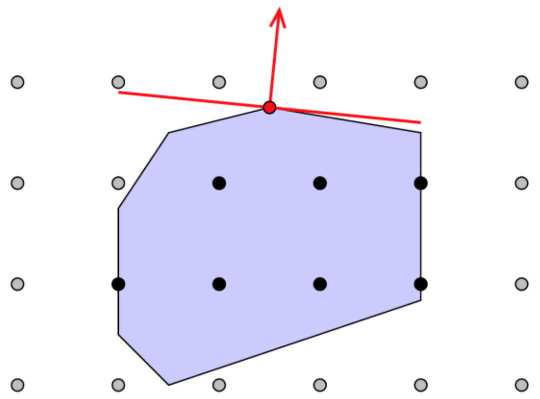
- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\le 3.6 & -x - 3y &\le -7.2 \\ -\ x - 3y &\le 0.5 & 0 \le x &\le 3 \\ -2x + 3y &\le 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\le 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound



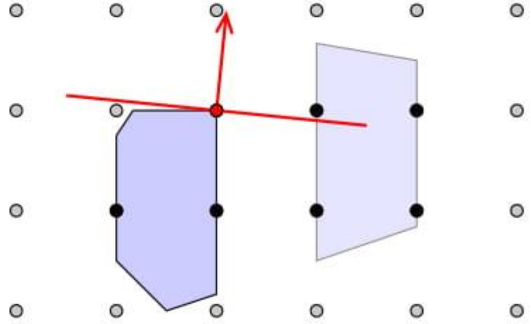


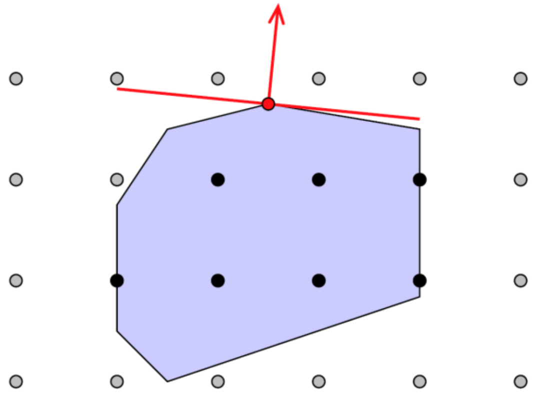
Solving Integer Optimization Problems

- Example MIP:

$$
\begin{array}{ll}
\max x + 5y & \\
\quad x - 3y \leq 3.6 & -x - 3y \leq -7.2 \\
- \quad x - 3y \leq 0.5 & 0 \leq x \leq 3 \\
-2x + 3y \leq 0.8 & x, y \in \mathbb{Z} \\
- \quad x + 3y \leq 4.2 &
\end{array}
$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound





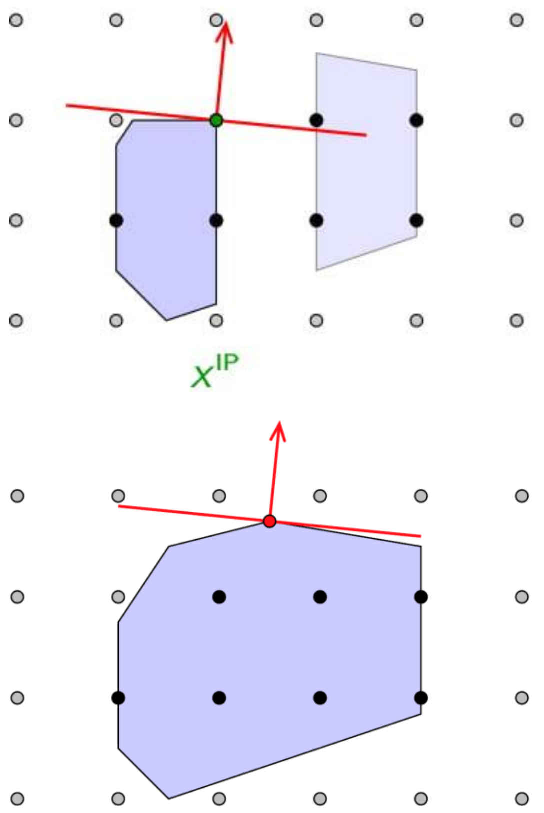
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound

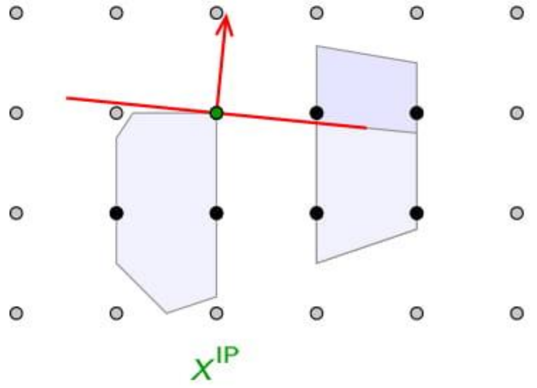


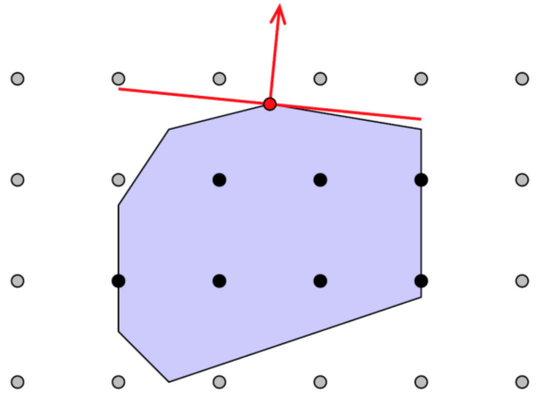
Solving Integer Optimization Problems

- Example MIP:

$$
\begin{array}{llll}
\max x + 5y & & & \\
& x - 3y \le 3.6 & & -x - 3y \le -7.2 \\
- & x - 3y \le 0.5 & & 0 \le x \le 3 \\
-2x & + 3y \le 0.8 & & x, y \in \mathbb{Z} \\
- & x + 3y \le 4.2 & &
\end{array}
$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound





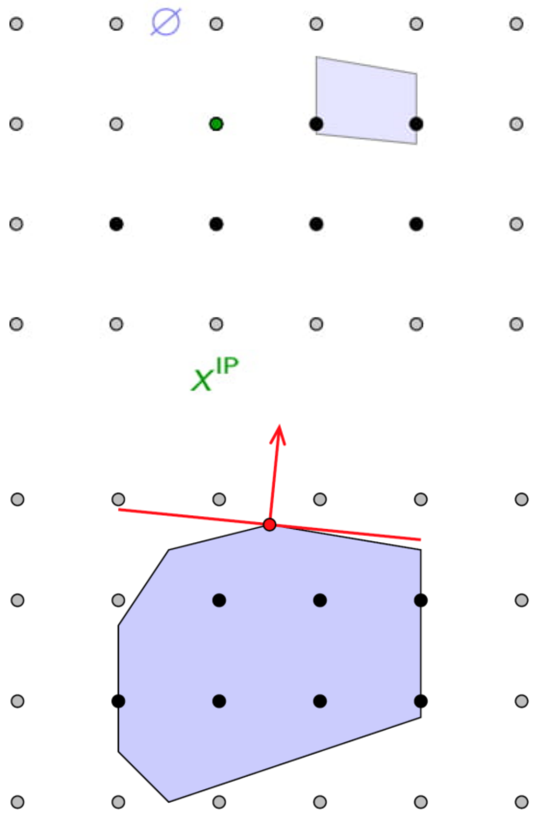
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 &\leq x \leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound



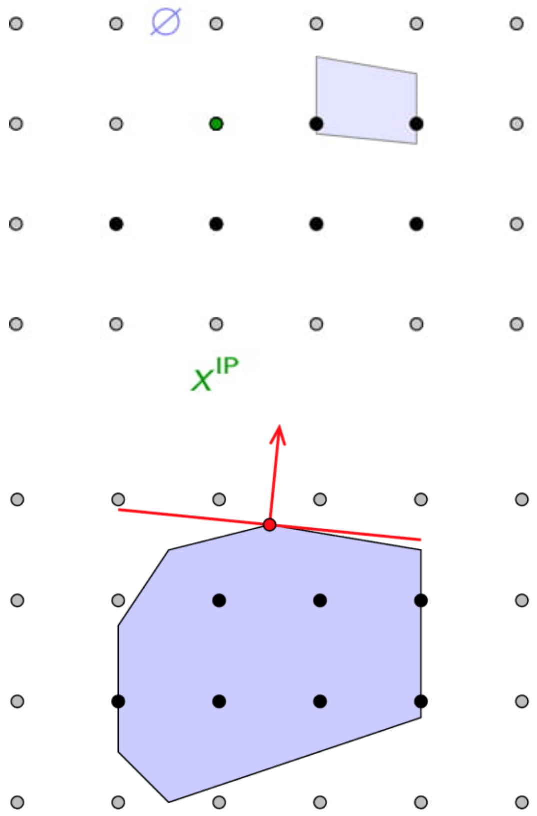
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\le 3.6 & -x - 3y &\le -7.2 \\ -\ x - 3y &\le 0.5 & 0 \le x &\le 3 \\ -2x + 3y &\le 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\le 4.2 & & \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method



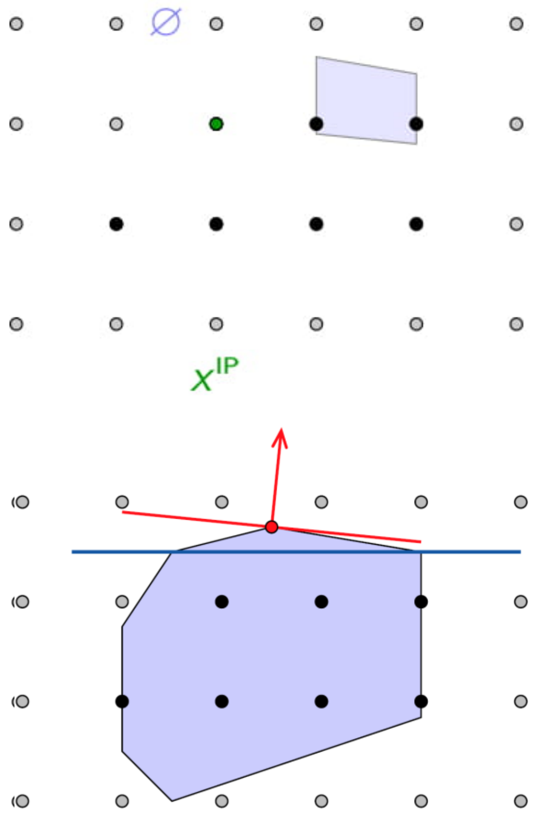
Solving Integer Optimization Problems

- Example MIP:

  $$\max x + 5y$$

  $$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 & & \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method



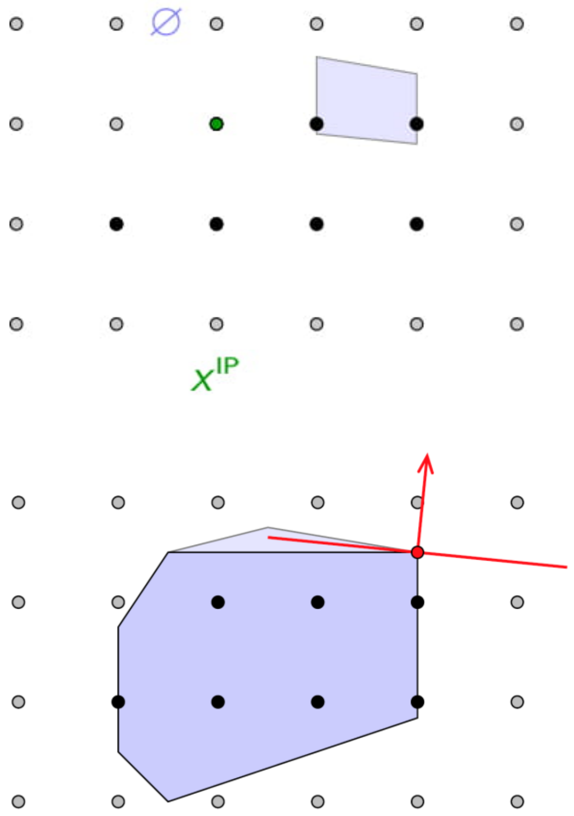
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\le 3.6 & -x - 3y &\le -7.2 \\ -\ x - 3y &\le 0.5 & 0 \le x &\le 3 \\ -2x + 3y &\le 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\le 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method

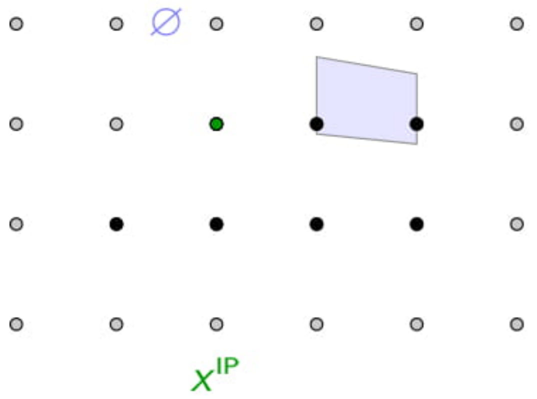


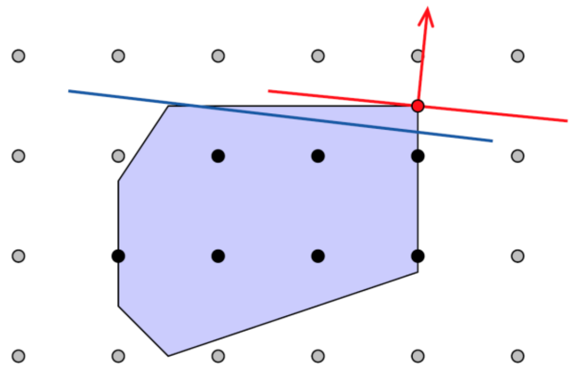
Solving Integer Optimization Problems

- Example MIP:

$$
\begin{array}{llll}
\max x + 5y & & & \\
x - 3y \leq 3.6 & & -x - 3y \leq -7.2 & \\
-\ x - 3y \leq 0.5 & & 0 \leq x \leq 3 & \\
-2x + 3y \leq 0.8 & & x, y \in \mathbb{Z} & \\
-\ x + 3y \leq 4.2 & & &
\end{array}
$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method



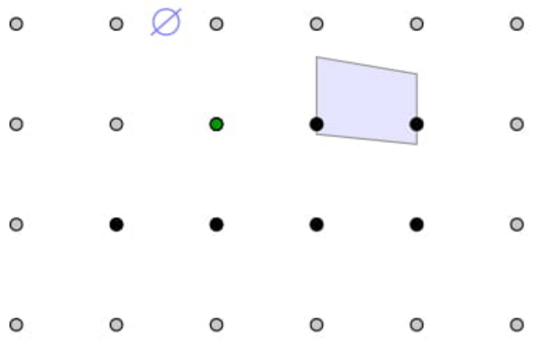


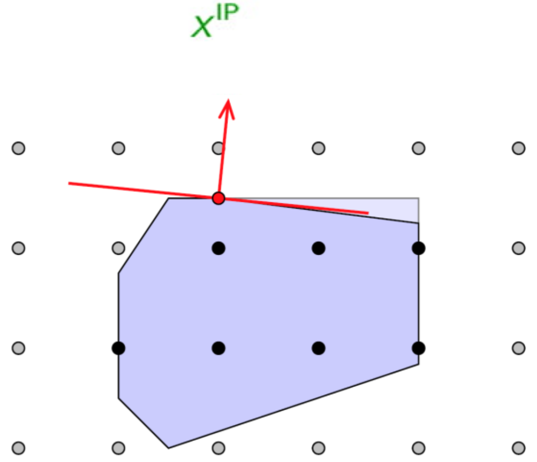
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$
$$\begin{array}{rl} x - 3y \leq 3.6 & -x - 3y \leq -7.2 \\ -\ \ x - 3y \leq 0.5 & 0 \leq x \leq 3 \\ -2x + 3y \leq 0.8 & x, y \in \mathbb{Z} \\ -\ \ x + 3y \leq 4.2 & \end{array}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method





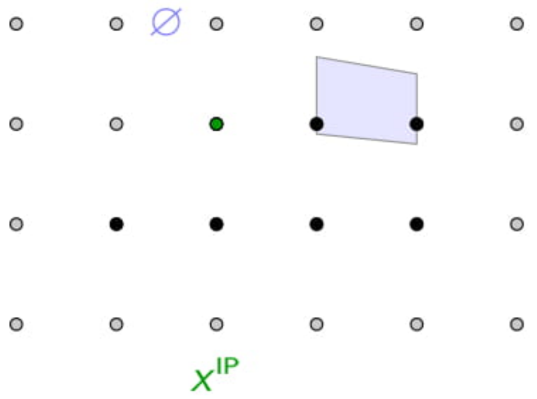
Solving Integer Optimization Problems

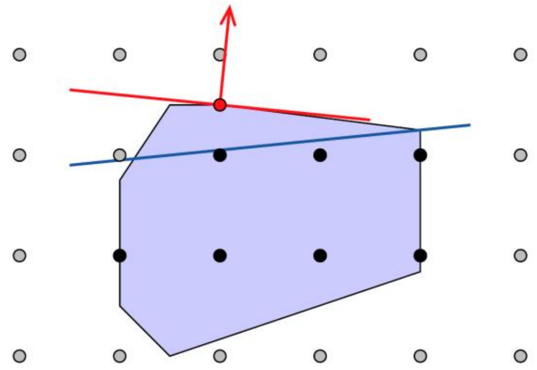
- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\ -\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\ -2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\leq 4.2 \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method





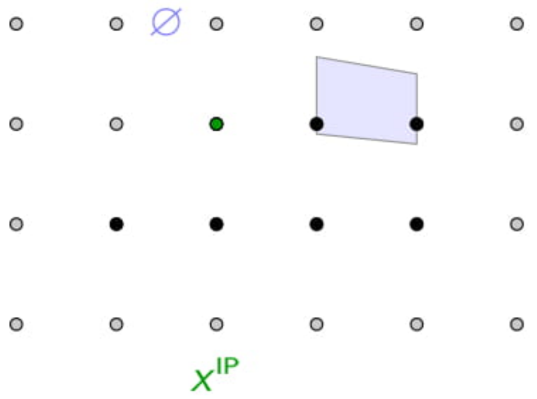
Solving Integer Optimization Problems

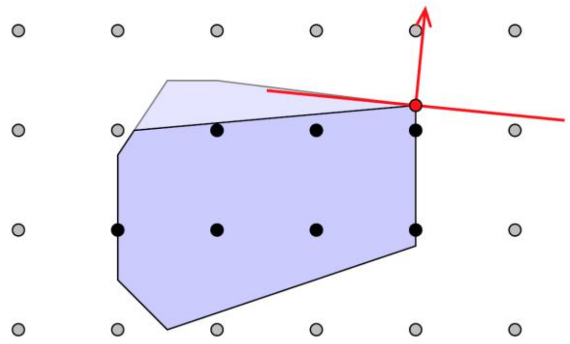
- Example MIP:

$$\max x + 5y$$

$$\begin{array}{rl}
x - 3y \le 3.6 & -x - 3y \le -7.2 \\
-\ \ x - 3y \le 0.5 & 0 \le x \le 3 \\
-2x + 3y \le 0.8 & x, y \in \mathbb{Z} \\
-\ \ x + 3y \le 4.2 &
\end{array}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method





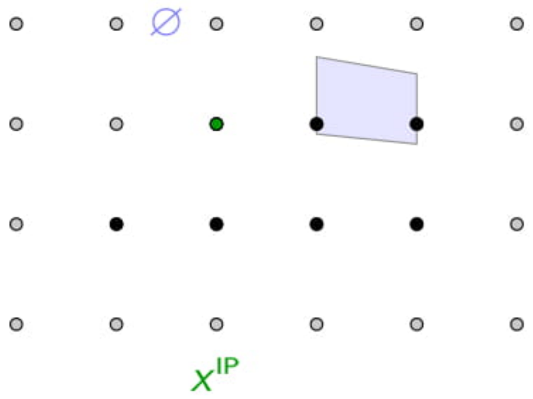
Solving Integer Optimization Problems

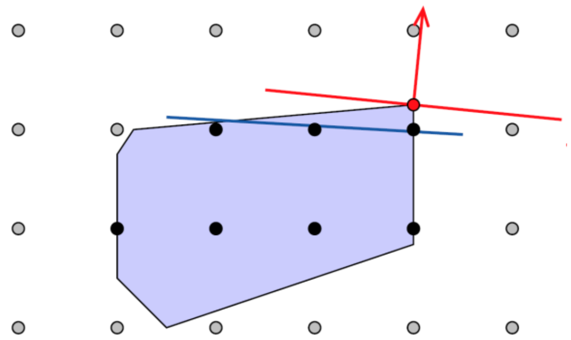
- Example MIP:

$$\max x + 5y$$

$$\begin{array}{rlrl}
x - 3y &\leq 3.6 & -x - 3y &\leq -7.2 \\
-\ x - 3y &\leq 0.5 & 0 \leq x &\leq 3 \\
-2x + 3y &\leq 0.8 & x, y &\in \mathbb{Z} \\
-\ x + 3y &\leq 4.2 & &
\end{array}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method





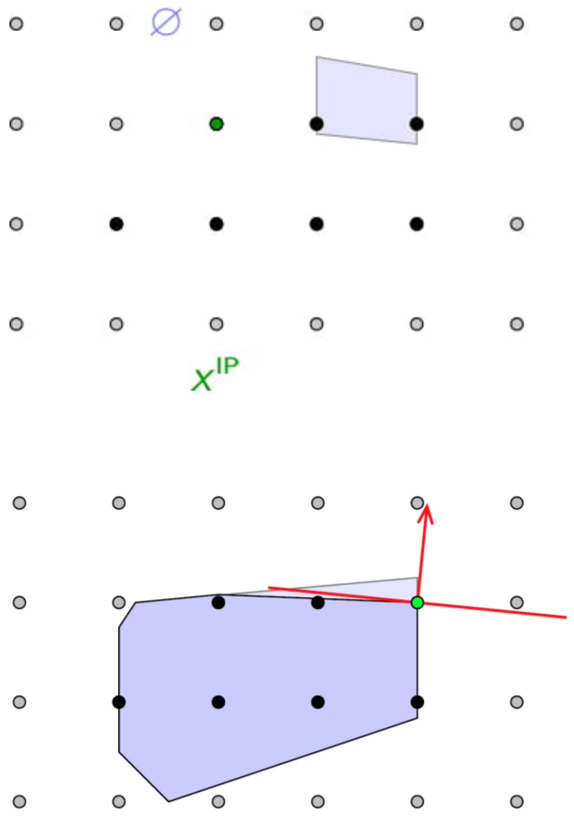
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\le 3.6 & -x - 3y &\le -7.2 \\ -\ \ x - 3y &\le 0.5 & 0 \le x &\le 3 \\ -2x + 3y &\le 0.8 & x, y &\in \mathbb{Z} \\ -\ \ x + 3y &\le 4.2 & & \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method



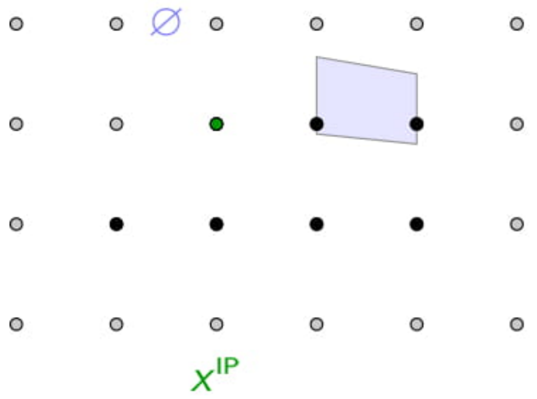
Solving Integer Optimization Problems

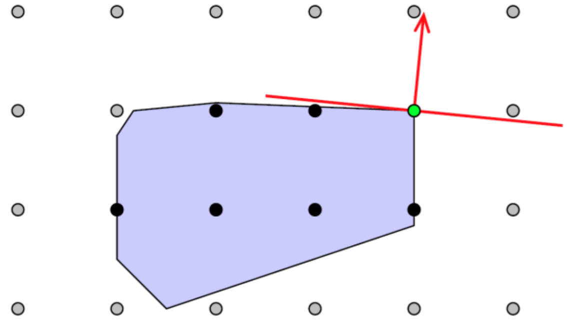
- Example MIP:

$$\max x + 5y$$

$$\begin{aligned} x - 3y &\le 3.6 & -x - 3y &\le -7.2 \\ -\ x - 3y &\le 0.5 & 0 \le x &\le 3 \\ -2x + 3y &\le 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\le 4.2 & & \end{aligned}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method





Solving Integer Optimization Problems

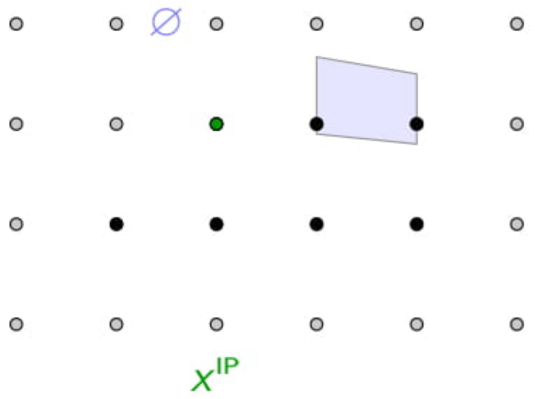
- Example MIP:

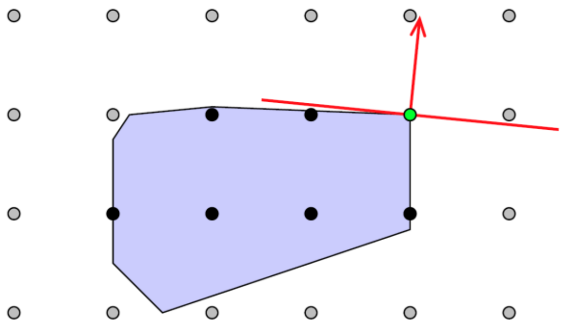
$$\max x + 5y$$

$$\begin{array}{rlrl} x - 3y &\le 3.6 & -x - 3y &\le -7.2 \\ -\ x - 3y &\le 0.5 & 0 \le x &\le 3 \\ -2x + 3y &\le 0.8 & x, y &\in \mathbb{Z} \\ -\ x + 3y &\le 4.2 & & \end{array}$$

- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method
- Two meaningful combinations of those algorithms:
  - Cut & Branch
  - Branch & Cut

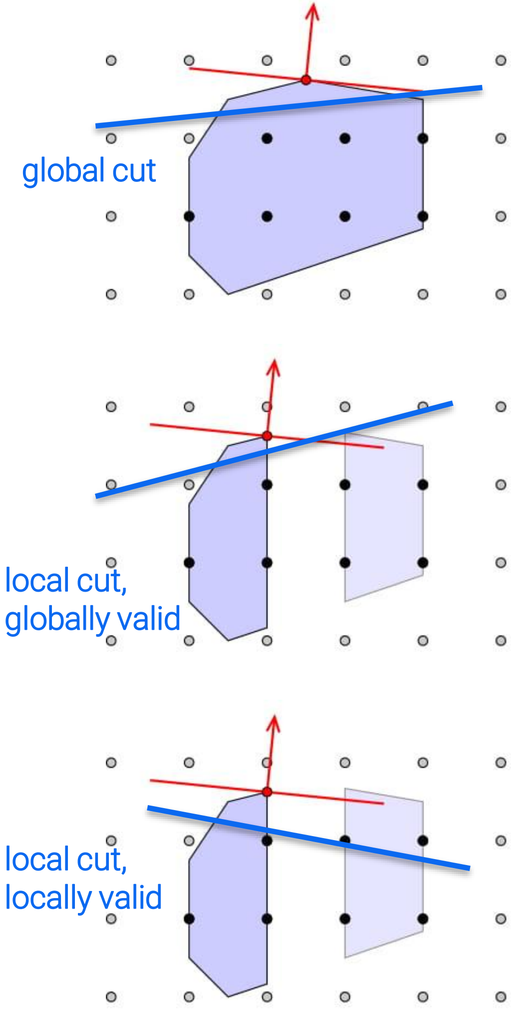
Which one is better?





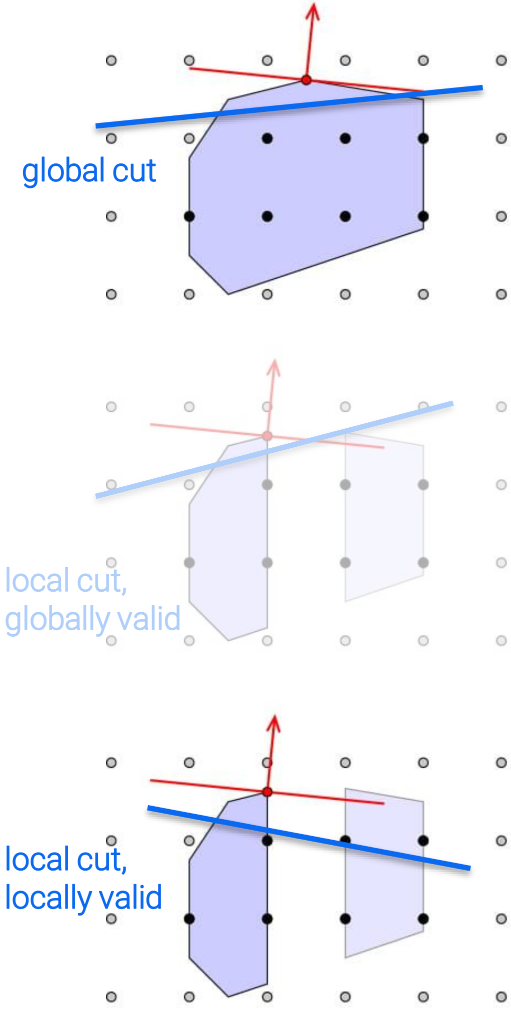
Local Cuts

- Global cuts
  - Generated at the root node
  - Hence globally valid by construction
- Local cuts
  - Generated at internal nodes
  - Either globally valid
    - When only using global information (e.g. bounds)
    - Can be re-used in other parts of the tree
  - Or locally valid
    - When using local bounds
    - Potentially stronger



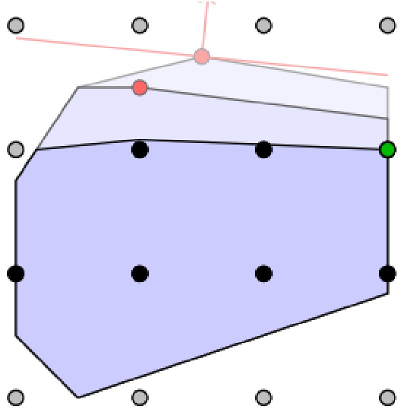
Local Cuts

- Global cuts
  - Generated at the root node
  - Hence globally valid by construction
- Local cuts
  - Generated at internal nodes
  - Either globally valid
    - When only using global information (e.g. bounds)
    - Can be re-used in other parts of the tree
  - Or locally valid
    - When using local bounds
    - Potentially stronger

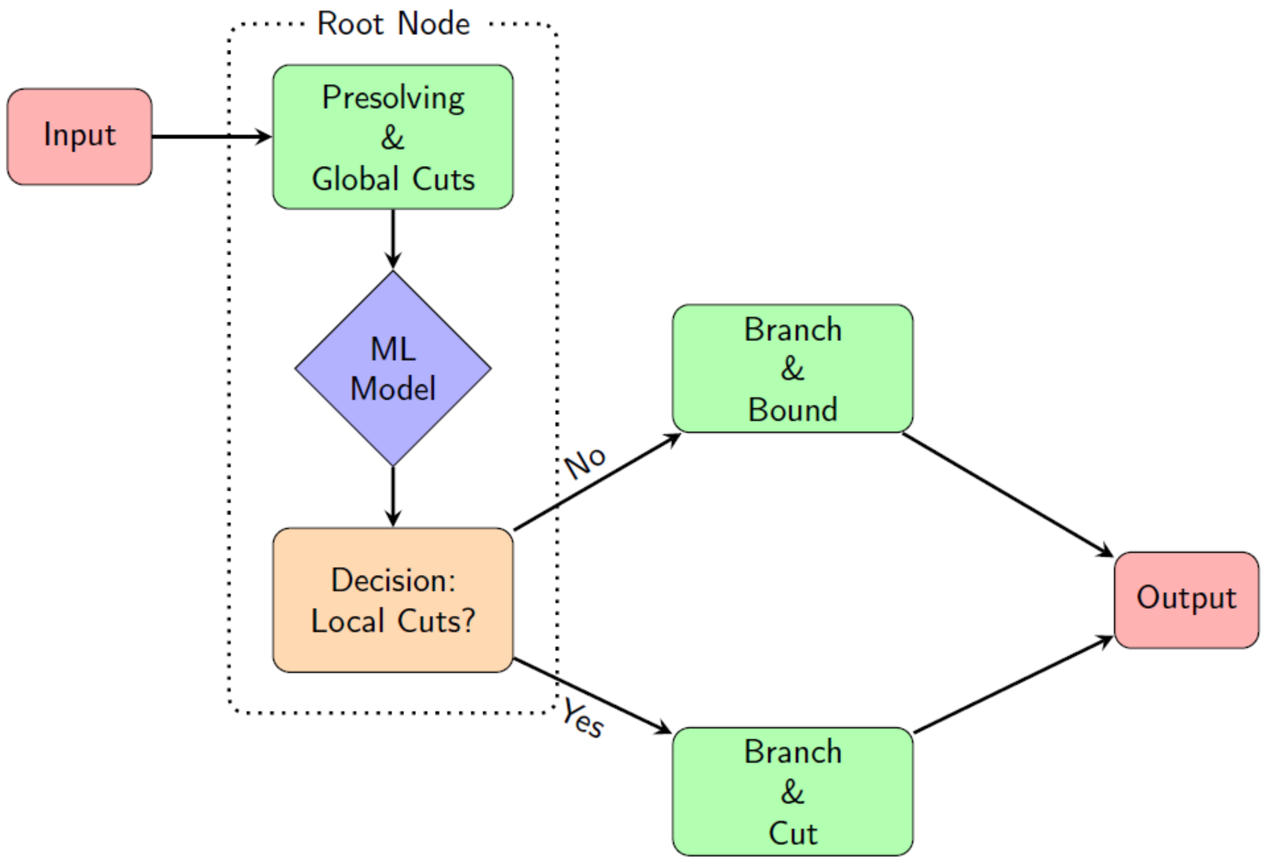


To Cut or Not to Cut?

- Crucial question: Should we generate cutting planes at tree nodes or only at the root?
  - Cuts are an essential part of branch-and-cut algorithms, many problems unsolvable without them
  - Local cuts complicate some other solver features (e.g., conflict analysis)
  - Cutting plane generation costs time and makes LPs slower to solve
    - When cuts do not make a difference, they slow down the solve
- How good are local cuts?
  - 45% of instances significantly benefit from local cuts
  - 23% of instances suffer from local cuts
- Idea: Try to predict at the beginning of tree search whether local cuts will work



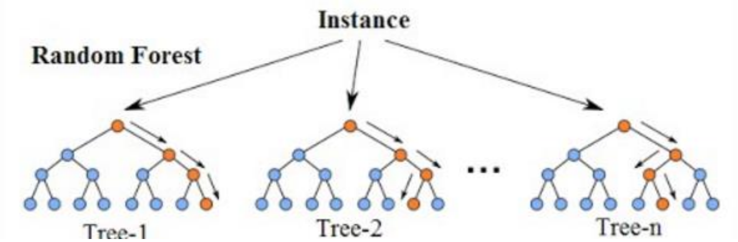
Flowchart



Root Node
Input
Presolving
&
Global Cuts
ML
Model
Decision:
Local Cuts?
No
Yes
Branch
&
Bound
Branch
&
Cut
Output

Learning to Use Local Cuts

- Train a random regression forest to predict speed-up factor from deactivating cuts
  - On more than 3000 customer instances
  - Final decision by an average and a majority vote

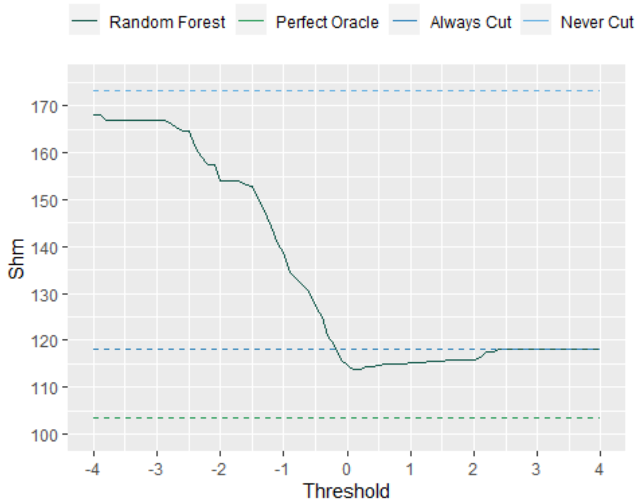


- Static Features: Row types, percentage binaries
  - Indicate whether this is a combinatorial problem
- Semi-static features: Density, numeric conditioning
  - Indicate whether cuts might lead to expensive LPs
- Dynamic feature: Gap closed by root cuts
  - Indicates the potential of cuts to raise the best bound

Computational results

Benchmark	#Instances	ΔSolved	Time	Nodes	PDI
MIP – Benchmark	5331	+9	-2%	+2%	-1%
→ > 100 sec	2226	+3	-2%	0	-1%
→ affected	480	+9	-15%	+19%	-10%

- Opposite effect on Time and Nodes (to be expected)
- Conservative setting, failed predictions are rare
- Under review for Math. Programming Computation



Learning To Select Cuts

To appear in Proc. of CPAIOR 2023,
joint work with Matthieu Besançon & Mark Turner

FICO

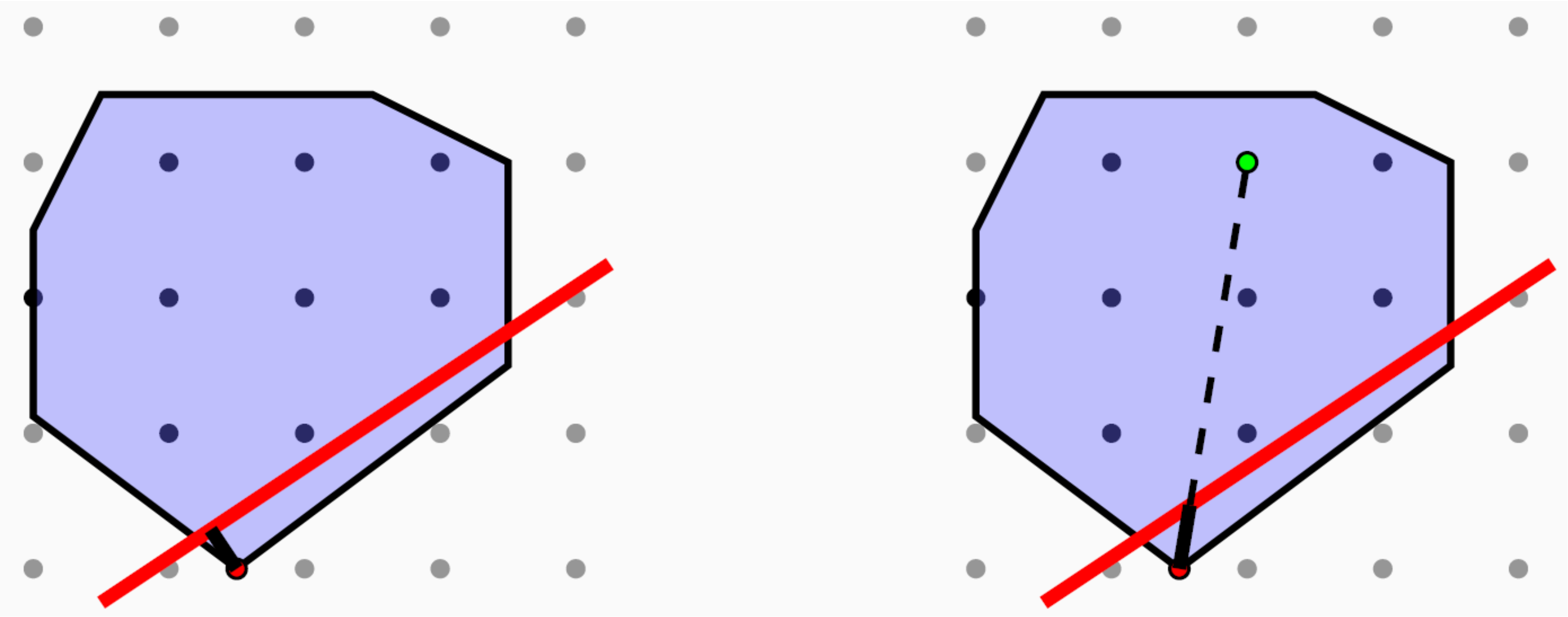
Cut Selection

- Cuts generated in rounds: separate, solve LP, rinse and repeat
- Most separation routines cheap (much cheaper than LP solve)
    - ⇒ Generate more cuts than needed, select the best ones

- Trade-off:
    - Adding more cuts?
        - Expensive LPs, numerically unstable
    - Adding less cut?
        - More nodes needed to solve
- ⇒ Sweet spot in between ⇒ How do we select promising cuts?

Cut Selection: state-of-the-art

- Efficacy – Intuition: Chop as much volume of the polyhedron as possible
- Directed cutoff distance (dcd) – Intuition: cut as close to the incumbent as possible

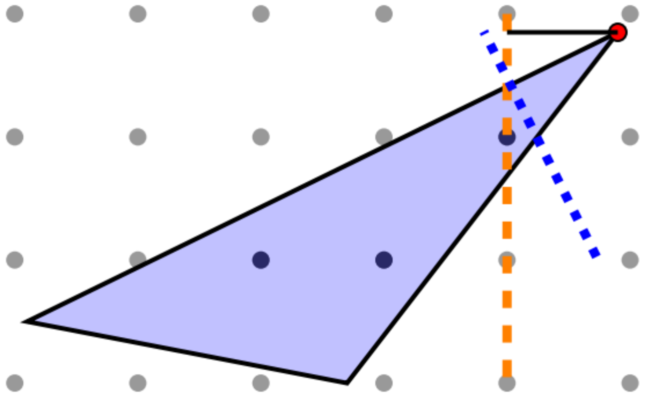


What is wrong with efficacy (and dcd)?

- Distance-based measures: How intuitive and robust are they really?

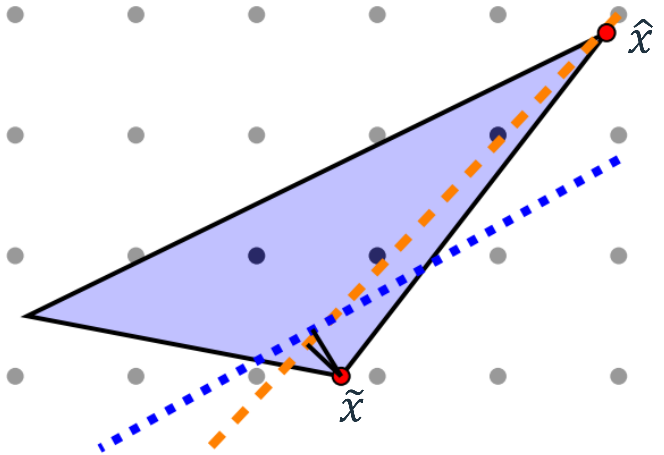
Distances to an infeasible projection

- Blue cut "better"
- But orange cut only "bad" outside polyhedron



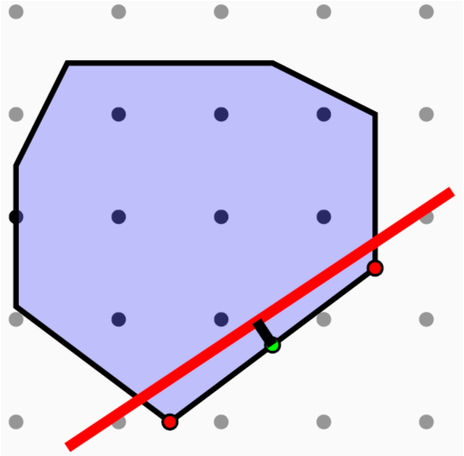
Dual degeneracy (optimal face)

- Blue cut slightly "better" for solution $\tilde{x}$
- But does not cut off solution $\hat{x}$
- Arbitrary solution (similar for dcd – incumbent)



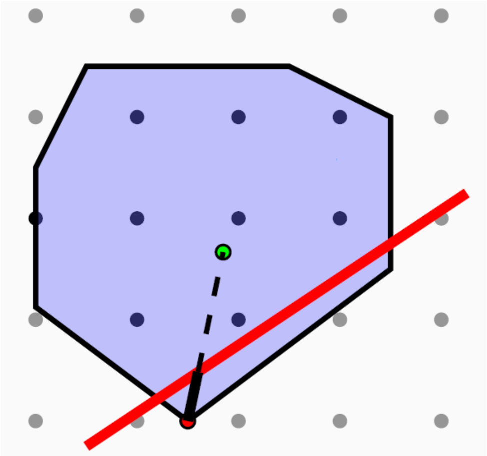
How might we do better?

- Use analytic centers (AC) ≈ central point
- Can be computed as LP, side-product of barrier algo



AC of the optimal face (AC-efficacy)

- Counters degeneracy
- Reduces risk to „project outside"



AC of polyhedron (AC-dcd)

- Projection always inside polyhedron
    - Expected to be central
- Constant reference point

Instance features

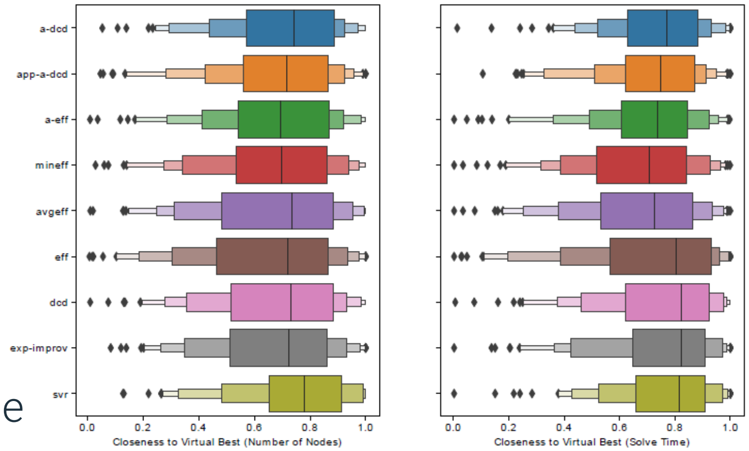
- Goal: Determine best selection criterion (out of eight):
  Given instance features, which score will produce the minimum tree size/runtime?
- Features of the transformed problem:
  - Dual degeneracy: % non-basic variables with zero-reduced cost
  - Primal degeneracy: % basic variables at bounds
  - Solution fractionality at root node
  - Thinness: % equality constraints
  - Density of the whole constraint matrix
- Test set: MIPLIB2017 Collection, solver: SCIP 8.0

Multiregression model

- Not a binary decision (but 8-fold): Classification?
- No single best for many instances:
  - no cut selected, certain cuts always selected, etc...
  - ties allowed?
- Multi-output regression: $\#\mathrm{nodes}_{\mathrm{best}} / \#\mathrm{nodes}$
- Support vector regression with cubic kernel
  - Several attempted models: regression trees, support vectors, random forests

Results

- Boxenplots: More right = better, olive: learned model
  - Nodes: Median and all percentiles are better
    - In particular at the lower end
  - Time: On par with Analytic DCD
  - Sh. geom. mean: Better for nodes, worse for time
  - Reduced performance variability

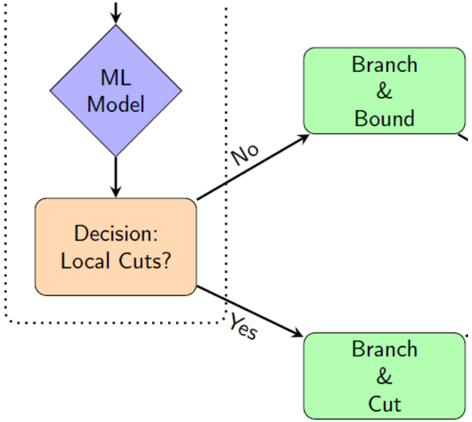


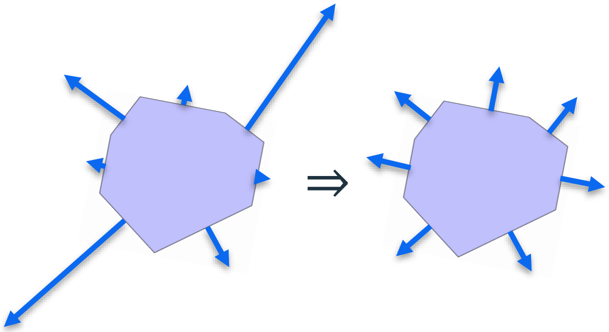
Conclusion

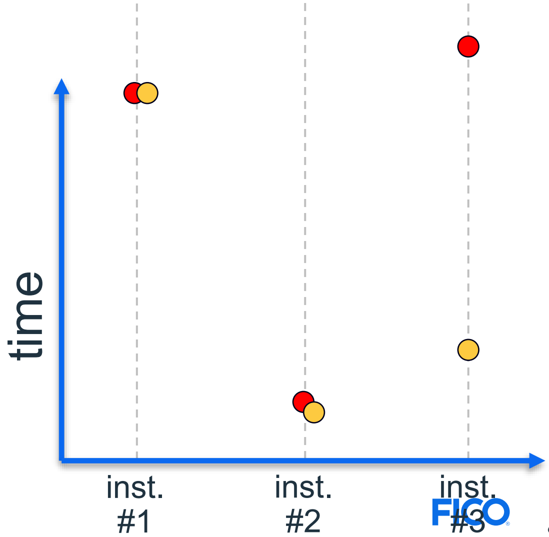
FICO

Key take-aways

- Modern MIP solvers use (fast, interpretable) ML models for decision making
- Faster is not the only definition of better
  - Improving numeric stability or reducing performance variability valuable by itself
- Use regression for „classification“ w.r.t. a continuous measure







Thank You!

Timo Berthold
TU Berlin, FICO, MODAL

image source: DALL-E