Machine learning in MI(NL)P solvers

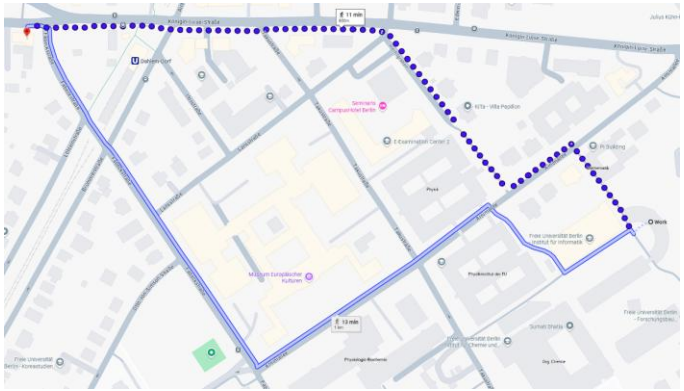
PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO

Exam

- Exam dates
  - September 11, September 25, October 5
  - 10-17, every 45min. Target length: 30min
- Contents
  - All lectures, and algorithmic/mathematical material from exercises, basic concepts
  - You will not need to do C++ coding (or discuss code)
  - Describe algorithms, intuitions behind them, the mathematical concepts, apply them to concrete examples, provide or explain visualizations
  - Ask yourself: How would you set up an exam for this course?
- Room: ZIB, 3104 (first floor Langbau)

Organization

- Q&A session 17.00-~17.40
- Drinks and dinner:
  - Alter Krug, 18.00



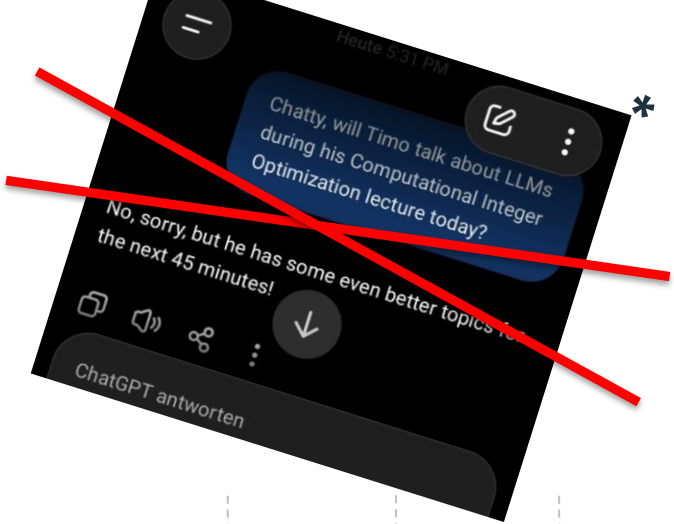
Agenda

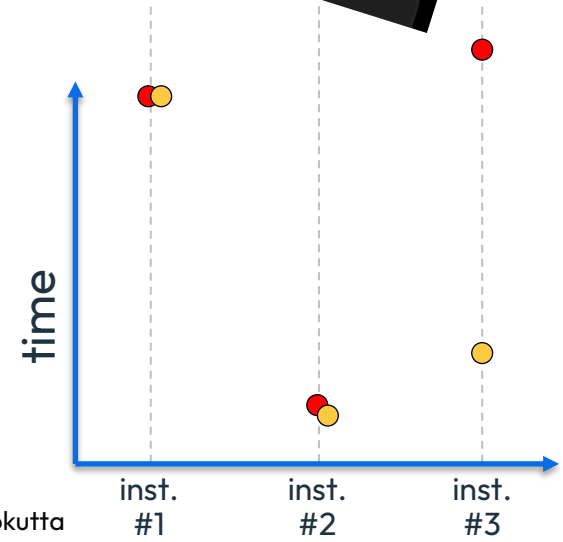
1. What do I mean by ML?
2. Learning to Scale
3. Learning to Use Local Cuts
4. Learning to Select Branching Rules

What do I mean by ML?

What do I mean by Machine Learning?

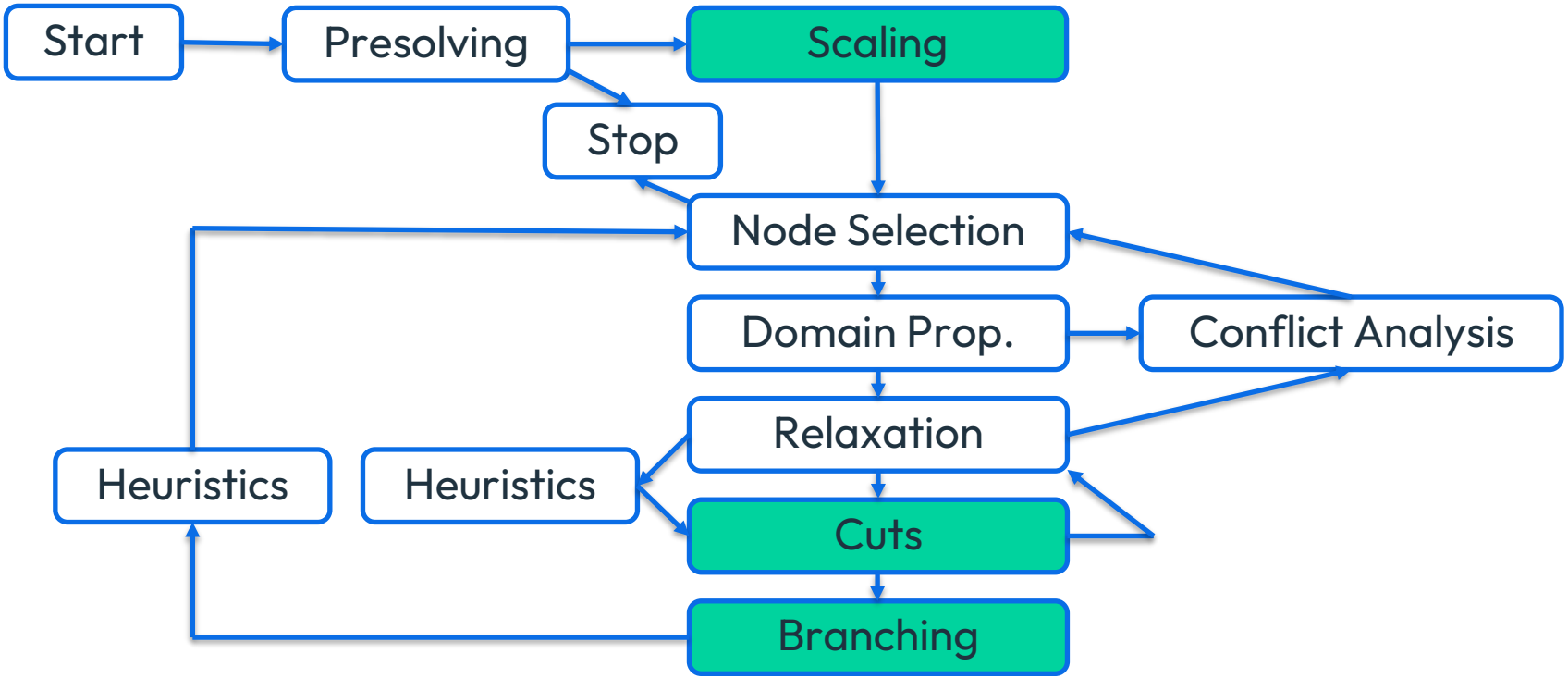
- Most of the cases: Learning a (binary) decision on which action to take inside a MIP solver
- „Classic“ offline, supervised learning
- Trained on general, heterogeneous test sets, not individually trained for application
- Easy-to-evaluate, ideally interpretable model
- **Use regression for „classification“ 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





*Side-track: “Global Optimization for Combinatorial Geometry Problems Revisited in the Era of LLMs”, https://arxiv.org/html/2601.05943v1, joint work with Dominik Kamp, Gioni Mexi, Imre Polik & Sebastian Pokutta

MIP Solver Flowchart



Learning to Scale

Proc. of AAAI 2021, joint work with Gregor Hendel

Huge Activity in Machine Learning for Mathematical Optimization

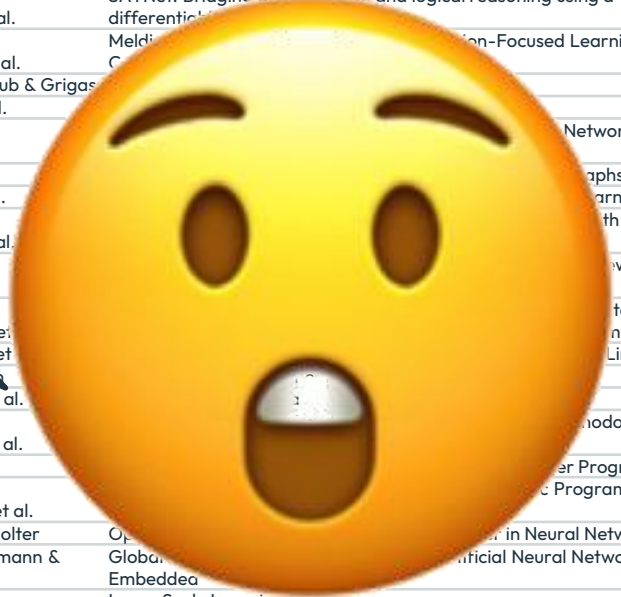
Ding et al.	Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction	2019
Bertsimas & Stellato	Online Mixed-Integer Optimization in Milliseconds	2019
Fischetti & Fraccaro	Machine learning meets mathematical optimization to predict the optimal production of offshore 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	Mach…	
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 Neural 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

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

(at the time)



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 deep learning and logical reasoning using a differenti…	2019
…er et al.	Meldi… …on-Focused Learning for C…	2018
…chtoub & Grigas		2017
…et al.		2018
…al.	…Networks and	2018
…t al.	…aphs	2017
…llo et al.	…arning	2016
Bowly et al.	…th	2017
Bowly	…w test	2019
…cois et…	…torial …n	2019
…tti et…	…Limit	2018
Kuhlmann		2019
Kruber et al.		2018
Bengio et al.	…thodological	2018
Hendel	…er Programming	2018
Bonami et al.	…c Programming	2017
Amos & Kolter	Op… …in Neural Networks	2017
Schweidtmann & Mitsos	Global… …ificial 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	2019
Bertsimas & S		2019
Fisch		2
Misra e		201
Bertsim		201
Tang et al.	ng: 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 Var e 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 Pro	
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 branchi	
Alvarez et al.	A Machine Learning-Base	
Alvarez et al.	Online Learning for Strong Bound	
Khalil et al.	Learning to branch in mixed	
Khalil	Machine Learning for Integer	
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	2013
Sabharwal et al.	Guiding Combinatorial Optimization with UCT	2012
Khalil et al.	Learning to Run Heuristics in Tree Search	2017
Hutter et al.	valuation	2012
H	mming Solvers	2010
		2019
		2019
	using a	2019
Wilde	sion-Focused Learning for	2018
Elmachtoub & Grigas	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 Combin	17
Bello et al.	Neural Com	
	Generation te	
vly et al.	controllable p	2017
	Stress testing mixed integ gh new test instance gene on 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 Tou d'Horizon	
	Adaptive Large N	
	Learning a Clo	
l.	Problem	
lter	OptNet	
nann &	Global De	
	Embedde	
	Large Scale Learni	020
g et al.	A General Large Neighborhood Search H ing Integer Programs	2020

Huge variability, no ground truth

Sophisticated rules already in place

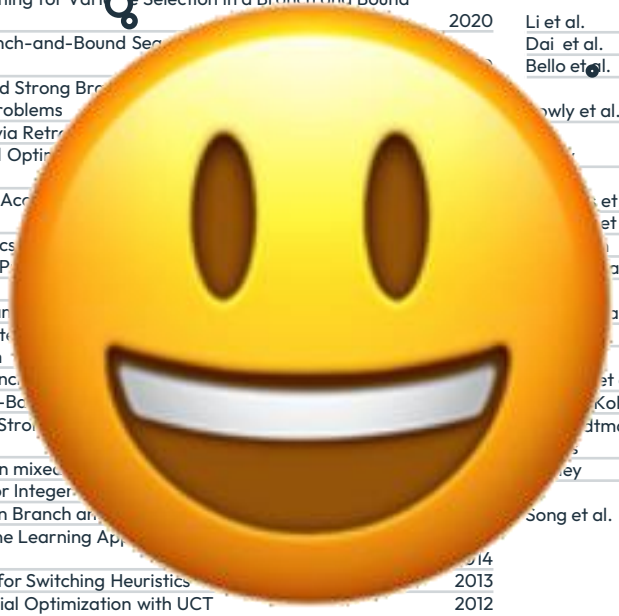
Heterogenity...

We don't even know good features

Huge Activity in Machine Learning for Mathematical Optimization

	Accelerating Primal Solution Findings for Mixed Integer Programs Based	
Ding et al.		2019
Bertsimas & S		2019
Fisch		2018
Misra e		2018
Bertsim		2018
Tang et al.	ng: Learning to Cut	2019
	Selecting cutting p atic semidefinite outer-approximation	
Baltean-Lugojan et al.	via trained neural networks	2018
	Reinforcement Learning for Var e Selection in a Branch and Bound	
Etheve et al.	Algorithm	2020
	Parameterizing Branch-and-Bound Se	
Zarpellon et al.	Policies	
	Learning Generalized Strong Bra	
Yang et al.	and 0-1 Knapsack Problems	
Song et al.	Learning to Search via Retr	
	Exact Combinatorial Optim	
Gasse et al.	Networks	
	Learning to Branch: Ac	
Lee et al.	Networks	
	Cuts, Primal Heuristics	
Hansknecht et al.	Traveling Salesman P	
Balcan et al.	Learning to branch	
Václavík et al.	Accelerating the bran	
	Deep Learning Assist	
Hottung et al.	marshalling Problem	
Lodi & Zarpellon	On learning and branc	
Alvarez et al.	A Machine Learning-Ba	
	Online Learning for Stro	
Alvarez et al.	Bound	
Khalil et al.	Learning to branch in mixed	
Khalil	Machine Learning for Integer	
He et al.	Learning to Search in Branch an	
	A Supervised Machine Learning Ap	
Alvarez et al.	Branch-And-Bound	014
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
	used Learning for	2018
Elmachtoub		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 Opti hms over Graphs	2017
Bello et al.	Neural C rcement Learning	2016
	Gene nces with	
	cont owly et al.	2017
	ough new test	
		2019
	Combinatorial	
	et al. an Problem	2019
	et al. eaching Time-Limit	2018
	ethods	2019
	al. tion	2018
	Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon	2018
	Adaptive Large Neighborhood Search for Mixed Integer Programming	2018
	Learning a Classification of Mixed-Integer Quadratic Programming Problems	2017
	et al.	
	Kolten OptNet: Differentiable Opti ks	2017
	dtmann &	
	Embedde	2018
	ey	2020
	A	
	Song et al. 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:
    - 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:
  - In the past (2019), more than half of client problems 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)

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

Recall

- Spread of nonzero matrix coefficient, rhs and objective coefficients

```
Coefficient range                    original                    solved
  Coefficients   [min,max] : [ 9.17e-05,  4.20e+02] / [ 1.17e-02,  1.31e+01]
  RHS and bounds [min,max] : [ 3.00e-02,  1.00e+04] / [ 5.00e-01,  4.35e+02]
  Objective      [min,max] : [ 1.00e+00,  7.16e+00] / [ 1.56e-02,  1.86e+02]
```

- Difference in the exponents (aka orders of magnitude) more relevant than the actual numbers
- Attention level gives a $[0,1]$-measure of the expected numerical difficulty
  - 0.0 = no major issues expected
  - The larger, the worse
  - 1.0 = Good luck…

Two scaling methods

Equilibrium scaling (Tomlin 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

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

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

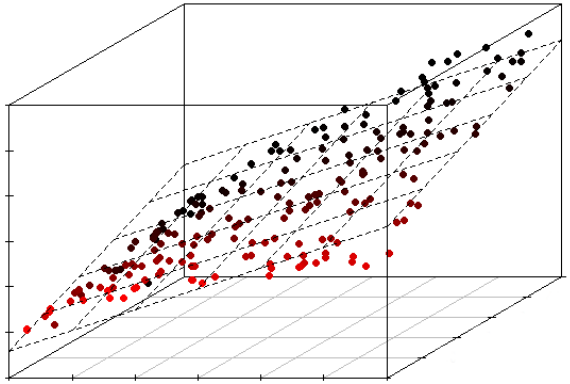
$$\begin{aligned} \max \quad & c'^T x' \\ \text{s.t.} \quad & A'x' \leq b' \end{aligned}$$

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: $\gamma := \log \frac{\max_{i,j} |A_{ij}|}{\min_{i,j} |A_{ij}|}$
  - Use $\gamma_{Equi} - \gamma_{Curtis}$ as a feature
  - Same procedure to get features for objective spread and right-hand side spread

Learning to Scale

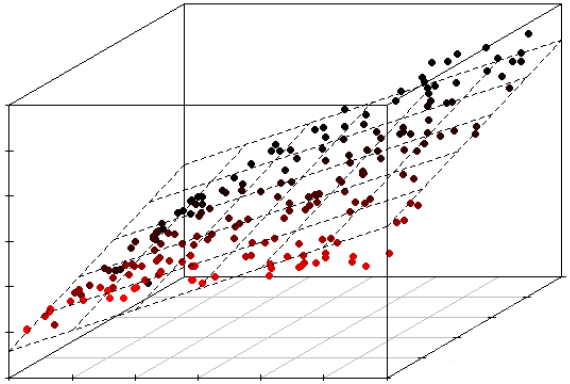
- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances



- Validation outcome:

Learning to Scale

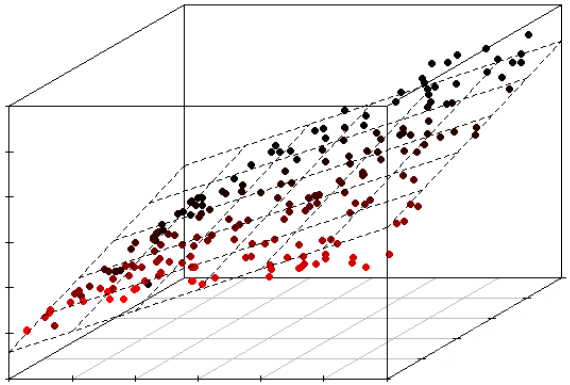
- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances



- Validation outcome:

Learning to Scale

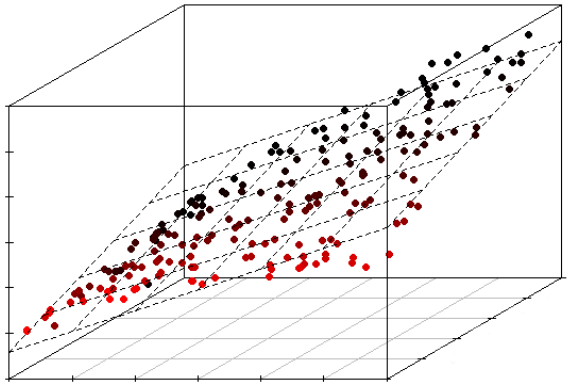
- New approach: Learn to Scale
  - Use linear regression model to predict which method will result in the smaller attention level
  - Tried random forests, neural nets, …
  - Use matrix spread, objective spread, rhs spread as features
- Trained on thousands of customer MIP instances



- 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 to Use Local Cuts

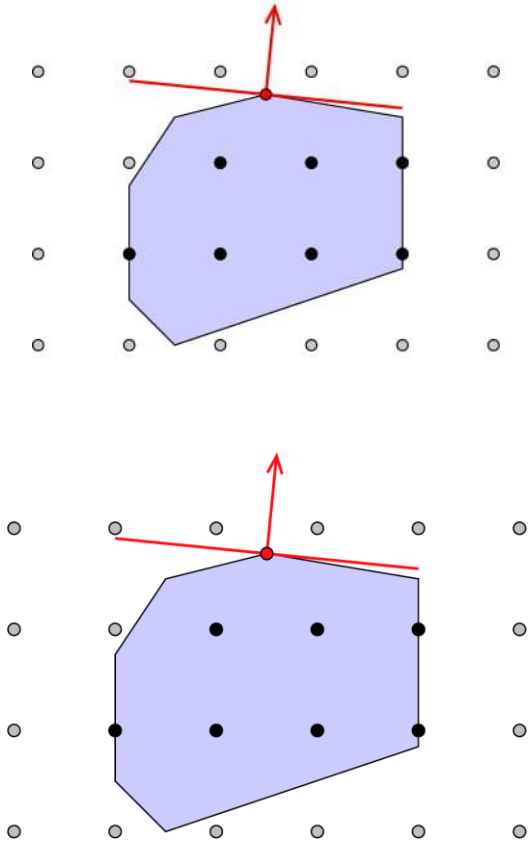
Mathematical Programming Computation, 2025
joint work with Matteo Francobaldi & Gregor Hendel

Solving Integer Optimization Problems

- Example MIP:

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

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



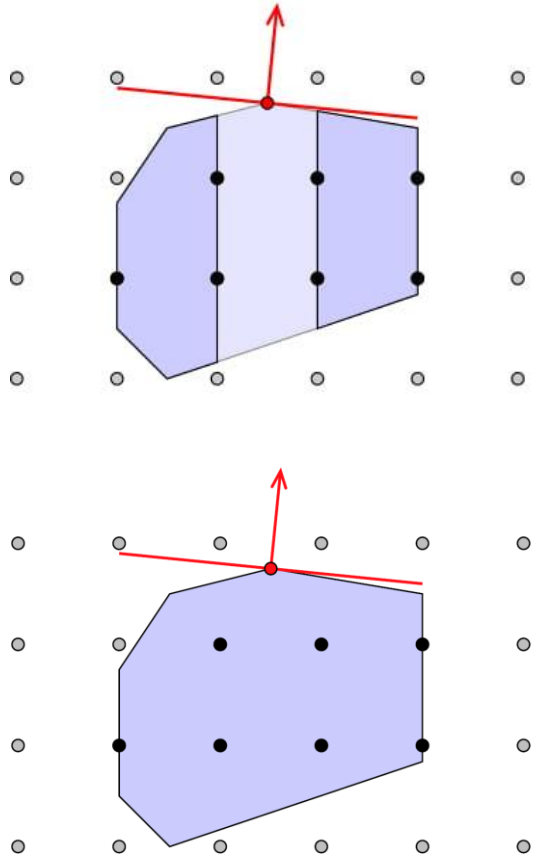
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



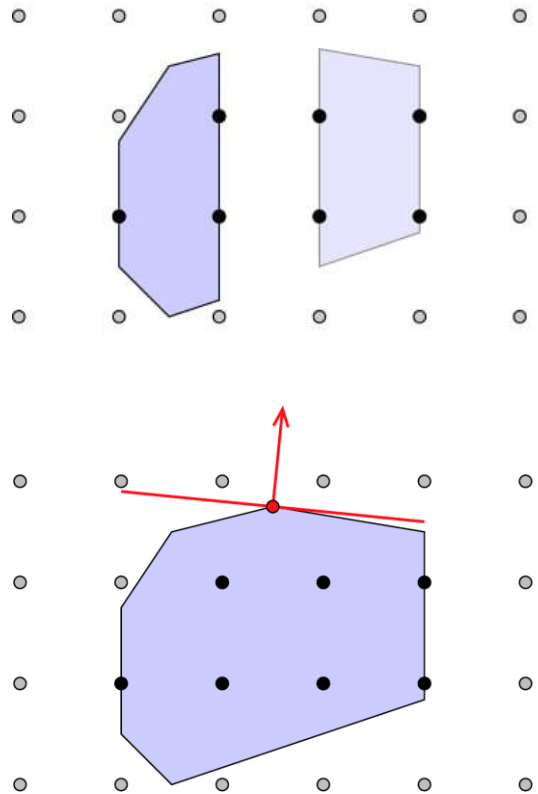
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



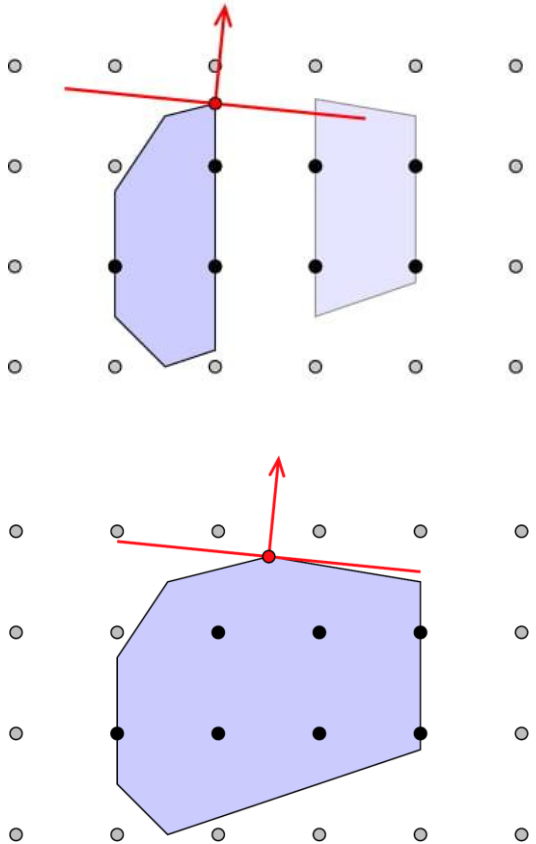
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



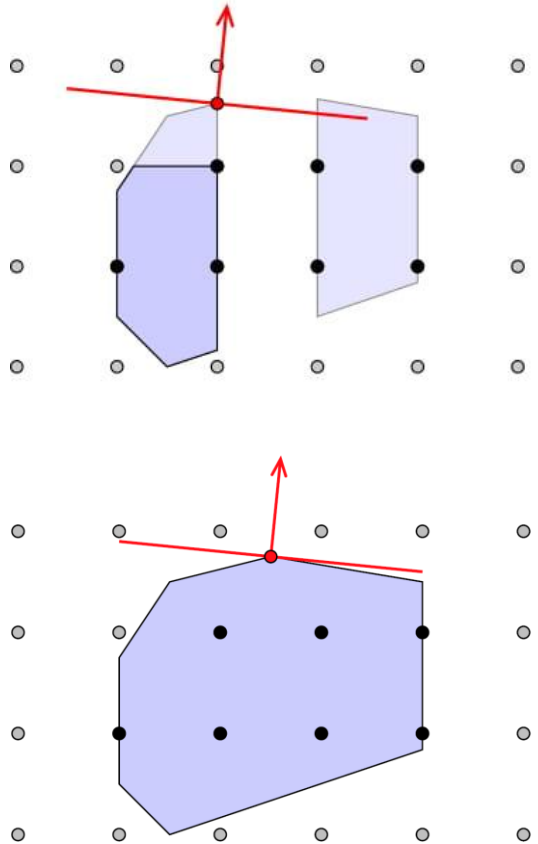
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



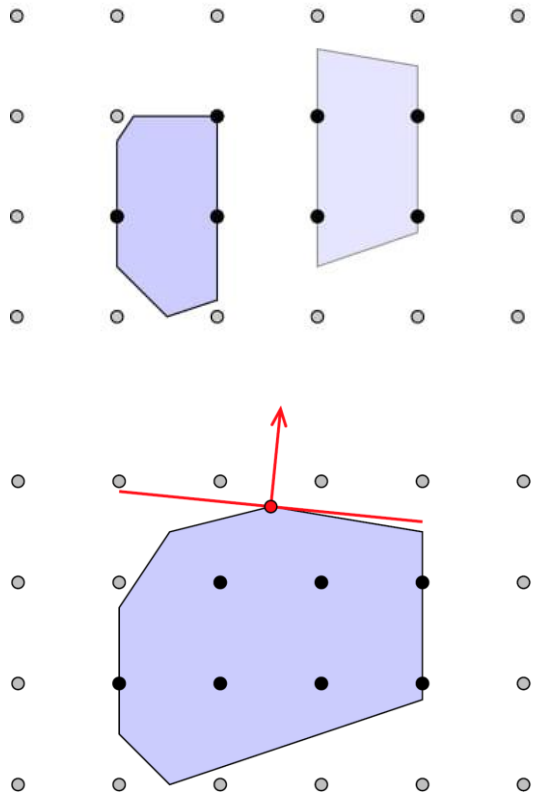
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



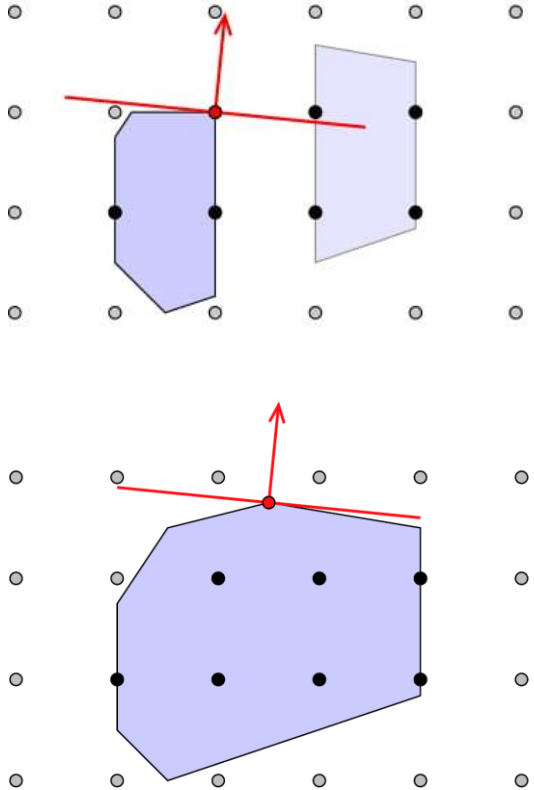
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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

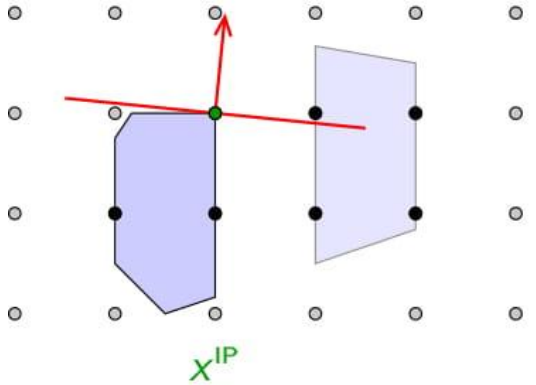


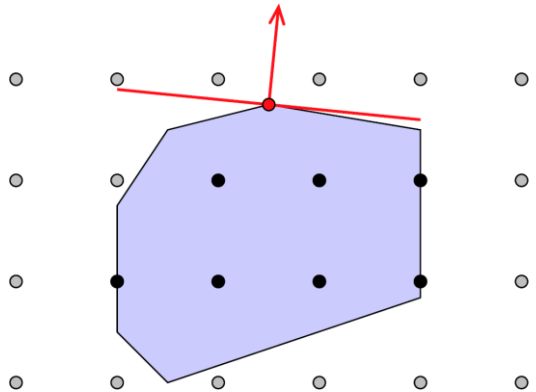
Solving Integer Optimization Problems

- Example MIP:

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

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



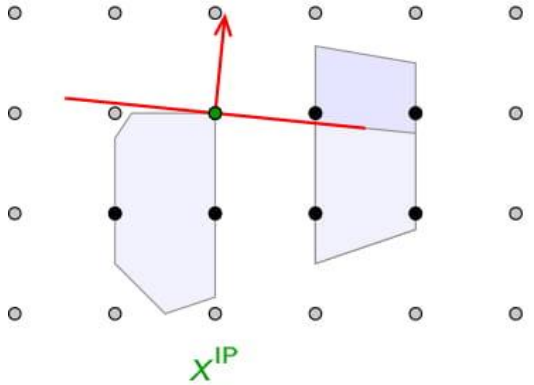


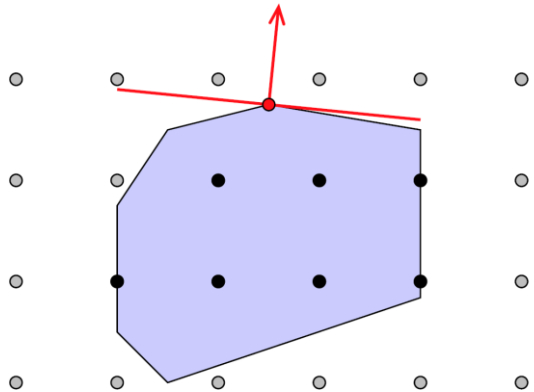
Solving Integer Optimization Problems

- Example MIP:

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

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





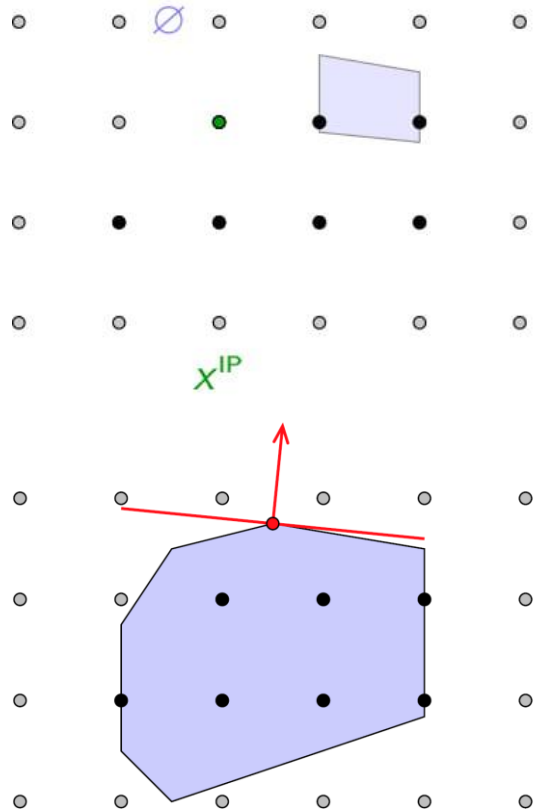
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



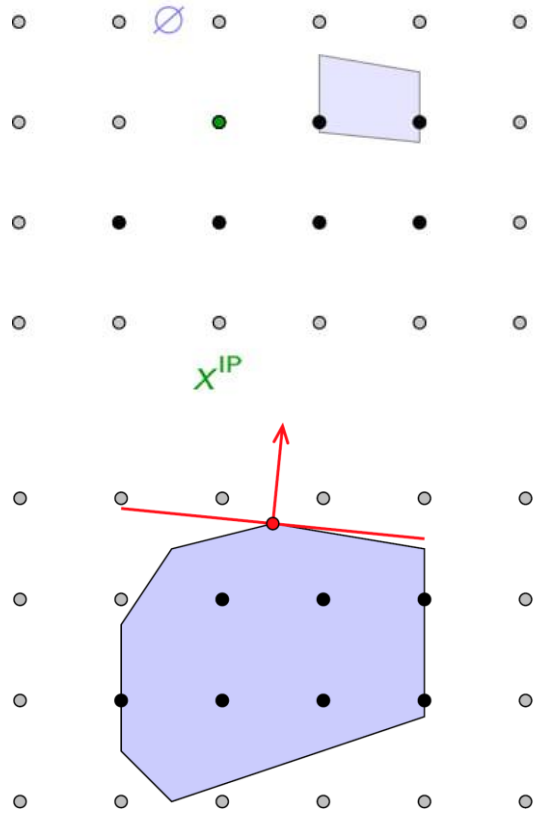
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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

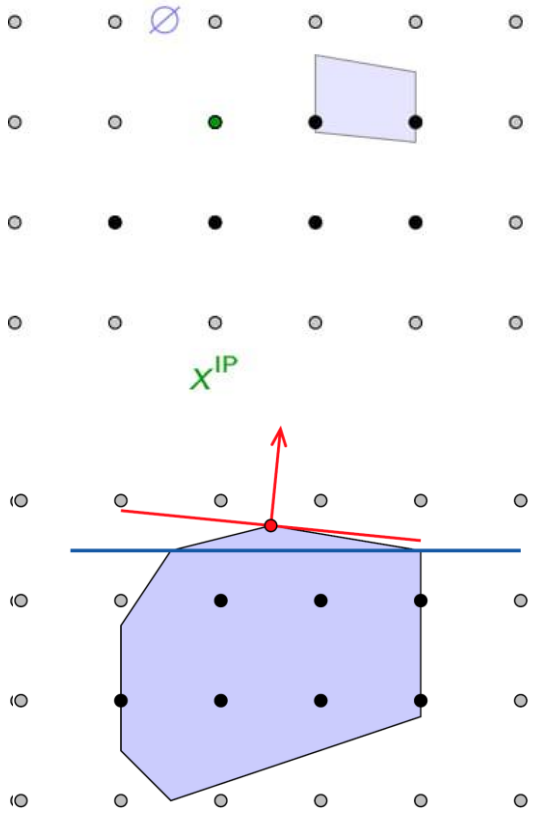


Solving Integer Optimization Problems

- Example MIP:

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

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



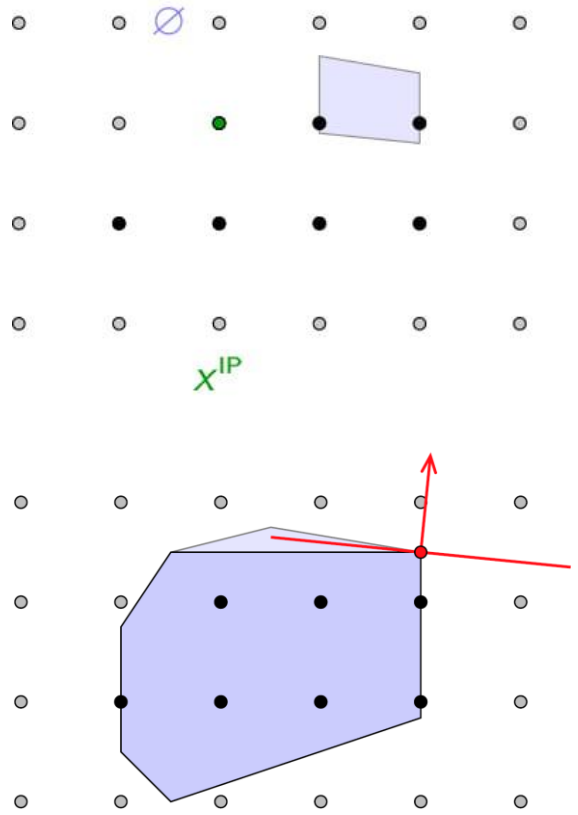
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



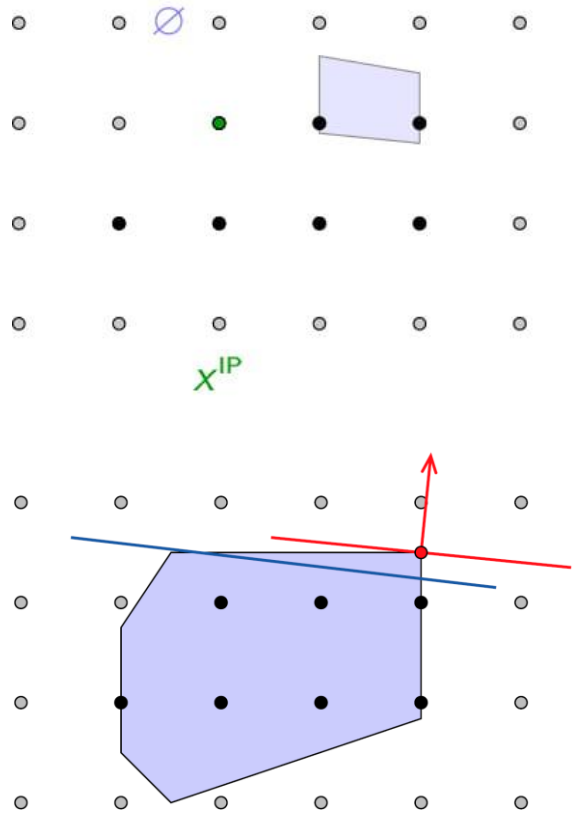
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



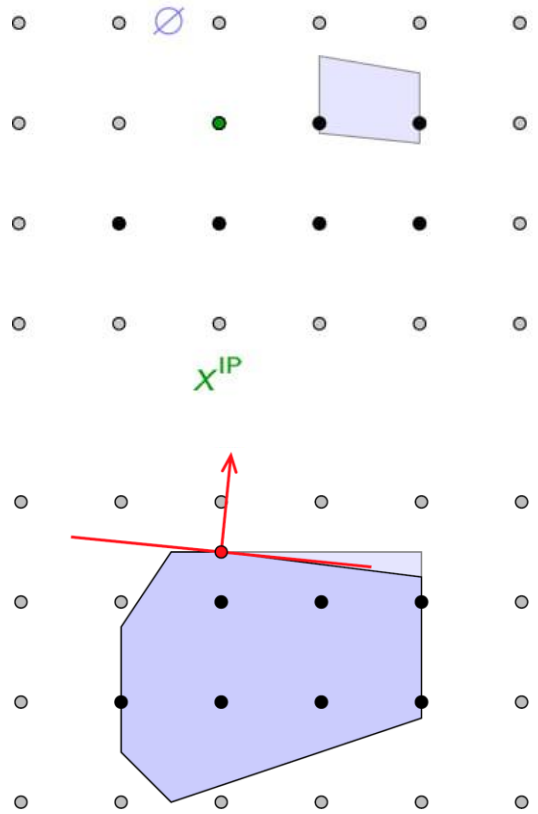
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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

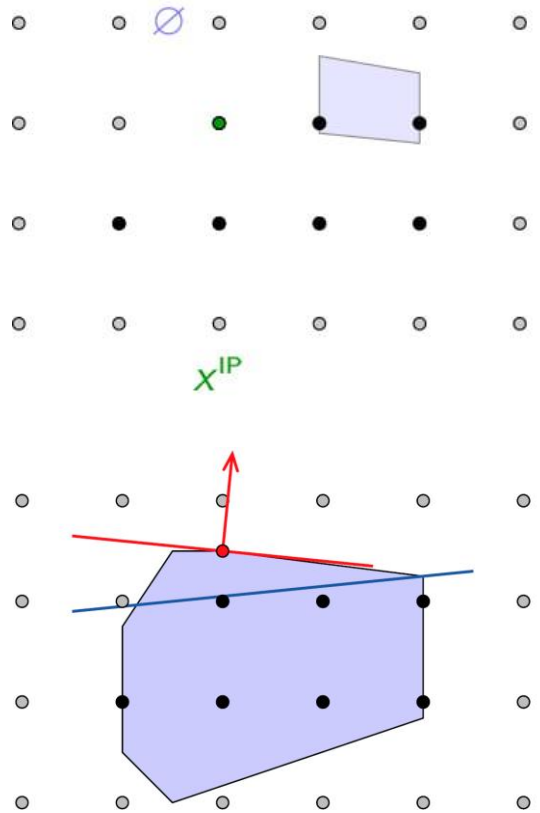


Solving Integer Optimization Problems

- Example MIP:

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

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



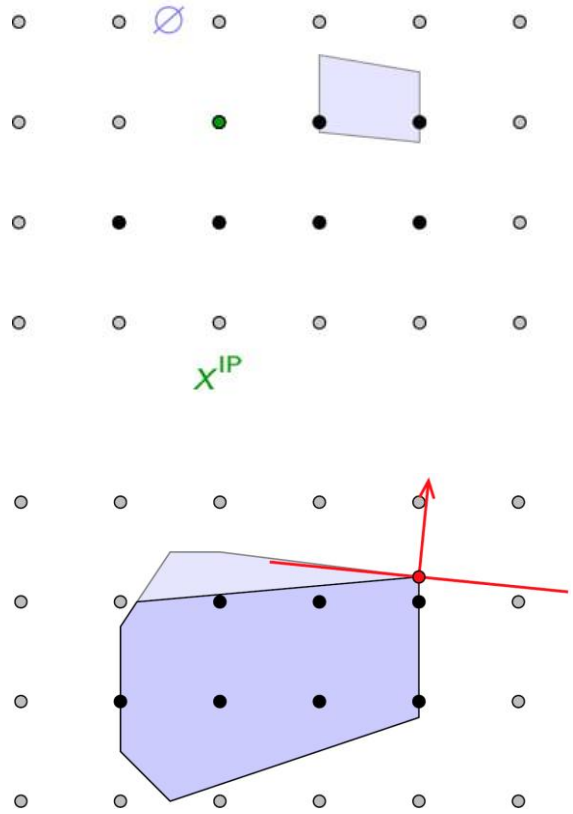
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



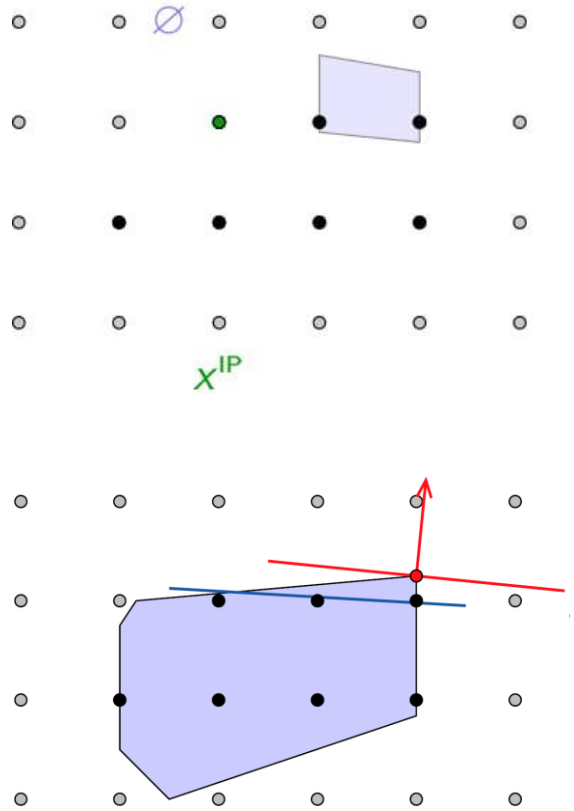
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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

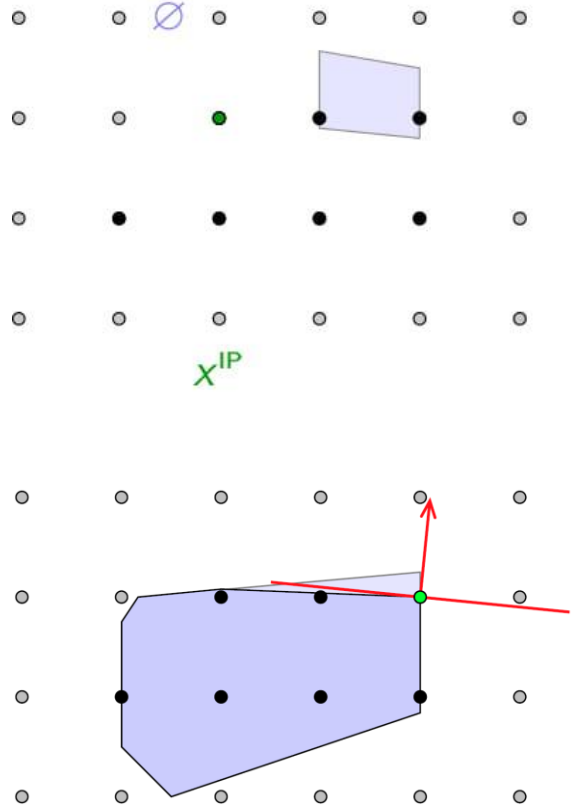


Solving Integer Optimization Problems

- Example MIP:

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

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



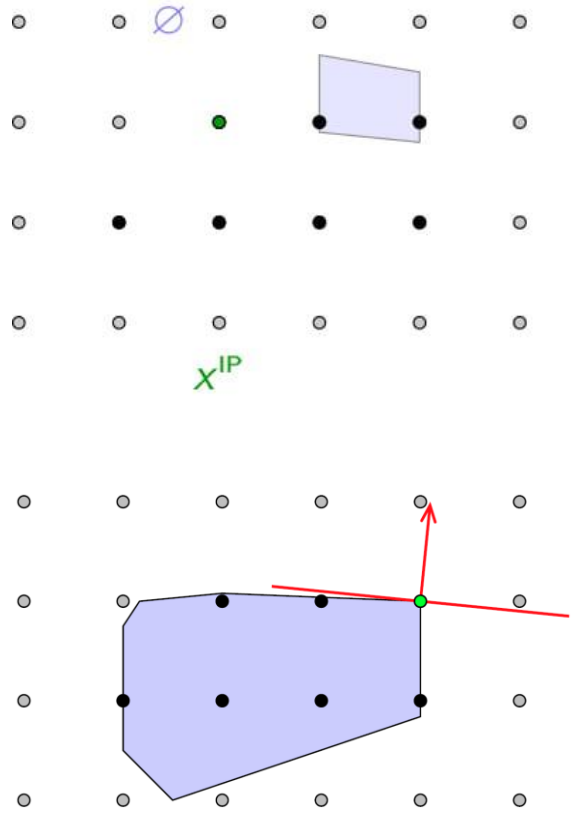
Solving Integer Optimization Problems

- Example MIP:

$$\max x + 5y$$

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

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



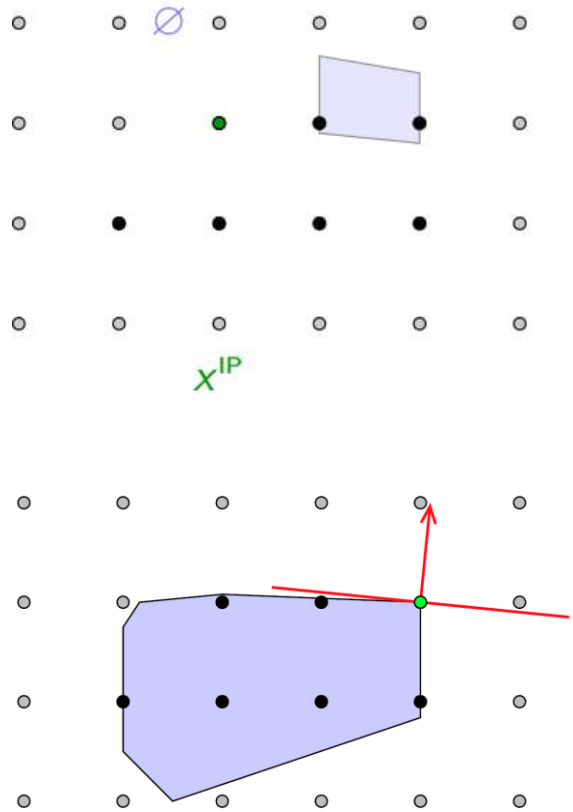
Solving Integer Optimization Problems

- Example MIP:

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

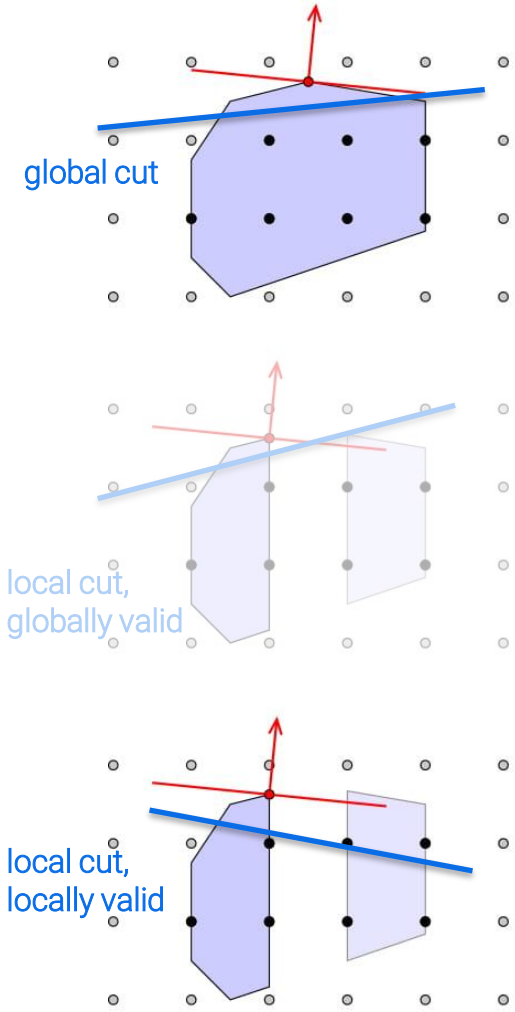
- Two principal algorithms to solve such problems:
  - Branch-and-Bound
  - Cutting Plane Method
- Two meaningful combinations of those algorithms:
  - Cut & Branch
  - Branch & Cut

Which one is better?



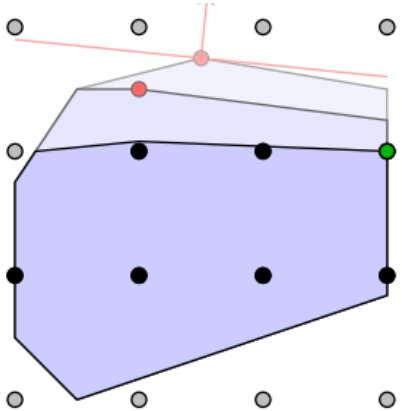
Local Cuts

- 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



To Cut or Not to Cut?

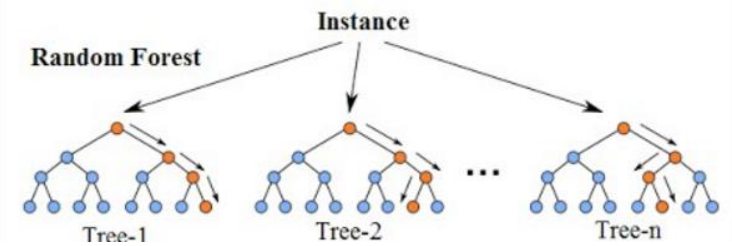
- Crucial question: Should we generate cutting planes at tree nodes or only at the root?
  - Cuts are an essential part of branch-and-cut algorithms, many problems unsolvable without them
  - Local cuts complicate some other solver features (e.g., conflict analysis)
  - Cutting plane generation costs 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



46

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



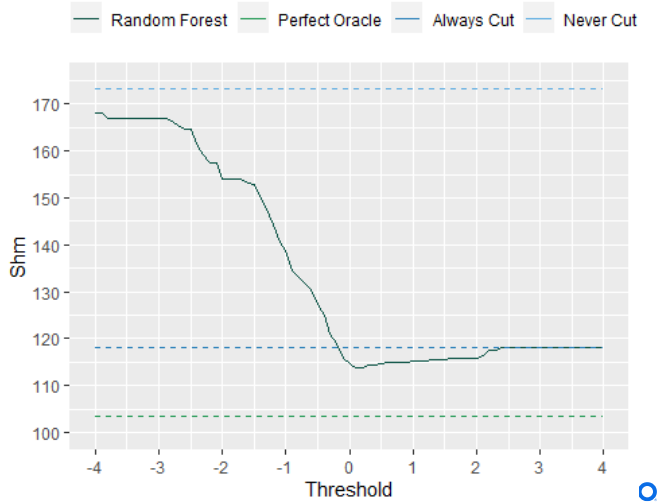
- Static Features: Row types, percentage binaries
  - Indicate whether this is a combinatorial problem
- Semi-static features: Density, numeric conditioning
  - Indicate whether cuts might lead to expensive LPs
- Dynamic feature: Gap closed by root cuts
  - Indicates the potential of cuts to raise the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

48

Computational results

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

- Opposite effect on Time and Nodes (to be expected)
- Conservative setting, failed predictions are rare



Learning to Select Branching Rules for MINLP

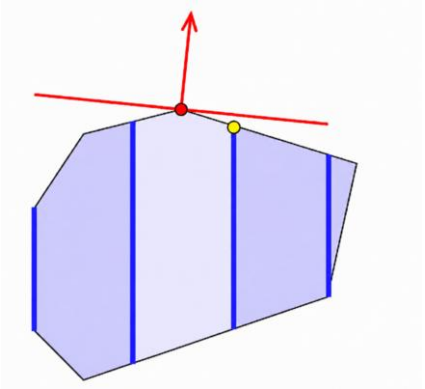
Proceedings of CPAIOR 2026,
joint work with Fritz Geis

FICO

Learning to Select Branching Rules for MINLP

Proceedings of CPAIOR 2026,
joint work with Fritz Geis

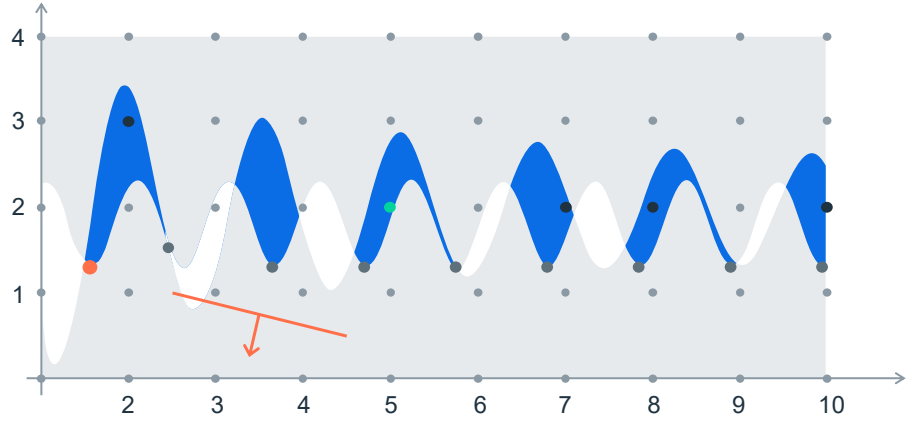
From MIP to MINLP



- General MIP:
  - 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
  - $\max c^T x$
    $s.t.\ Ax \leq b$
    $x \in \mathbb{Z}^I \times \mathbb{R}^{N \setminus I}$
    $\ell \leq x \leq u$

- Solution algorithm: (supercharged) LP-based branch-and-bound (on integer variables)
  - Once the integer variables take integral value in the LP solution, the solution is feasible and optimal for the branching subproblem

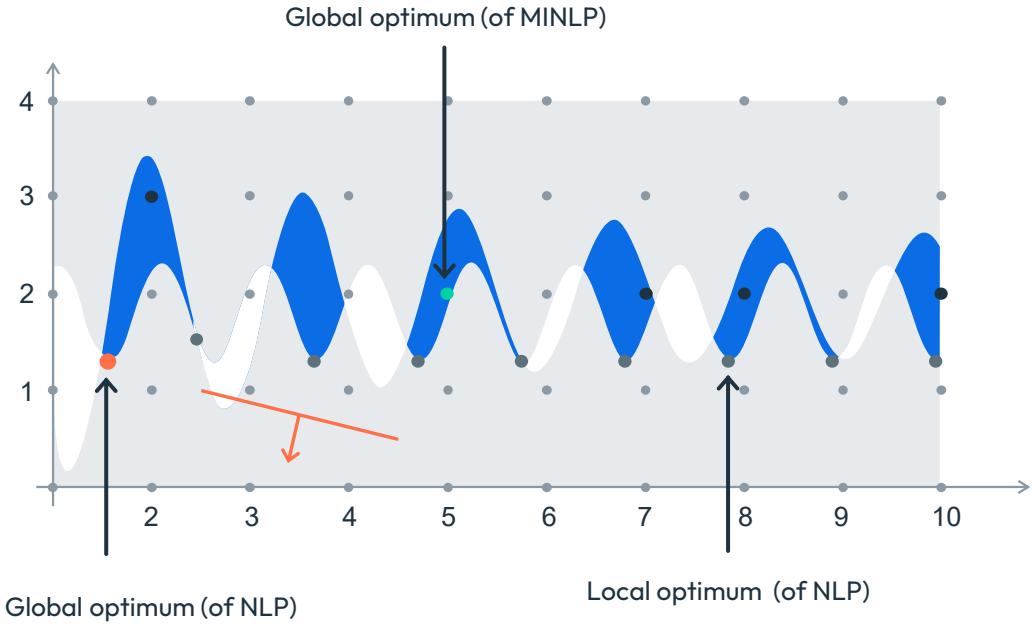
From MIP to MINLP



- General MINLP:
  - maximizing a **non**linear objective function over a set of **non**linear constraints
  - Some or all variables have to take integral values
  - $\min f(x)$
    $\text{s.t. } g_j(x) \leq 0 \text{ for } k = 1, \dots, m,$
    $x \in \mathbb{Z}^I \times \mathbb{R}^{N \setminus I}$
    $\ell \leq x \leq u$
    where $f, g_j \colon \mathbb{R}^n \supseteq [l, u] \to \mathbb{R}$
- A couple of state-of-the-art MIP solvers have recently been extended to MINLP
- Solution algorithm: (supercharged) LP-based branch-and-bound (on integer and continuous variables)
  - Convex MINLP: Once the integer variables take integral value in the LP solution, cutting planes can be used to get to ε-feasibility
  - Nonconvex MINLP: Might need to branch on continuous variables to reach constraint feasibility

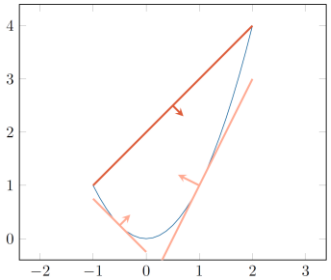
Nonconvex MINLP

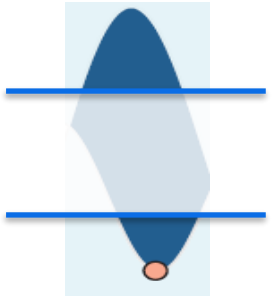
- Example: Two nonconvex inequalities
  - six disconnected regions
  - nine local optima
- Rich literature on local optimization
- This talk: Global Optimization
  - of nonconvex MINLPs
  - using FICO Xpress Global

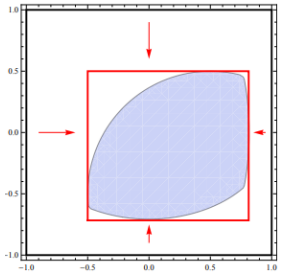


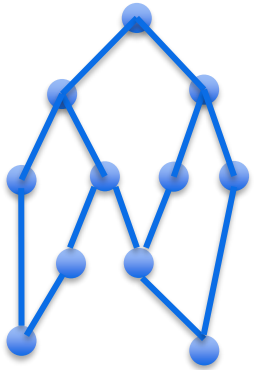
Global Optimizations Techniques

- Nothing to be afraid of ☺
- Cutting Planes
- Branching
- Presolving / bound tightening
- And a (maybe peculiar, but super-useful) representation of the problem as a graph



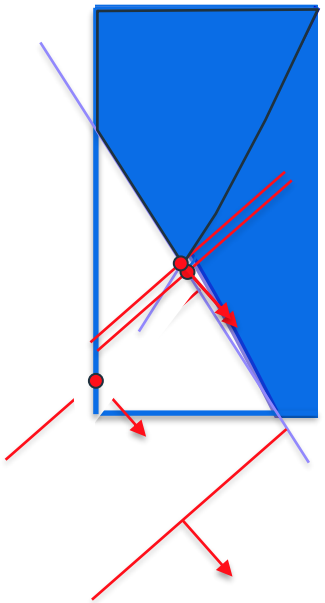






Spatial Branching: Why? (example)

- $\min y - x$
  $s.t. \; y \geq x^2$
  $y \geq 8 - x^2$
- Feasible region: “curved, purple W-shape”
- First constraint: Easy to relax linearly
  - Second constraint: No linear relaxation
- Solve LP
  - Violates the second constraint
  - Resolve via spatial branching
  - Now, add linear relaxation of the second constraint
- Solve LP
  - Add cut
  - Solve again
- Are we good?
  - Still on secant, need to branch again



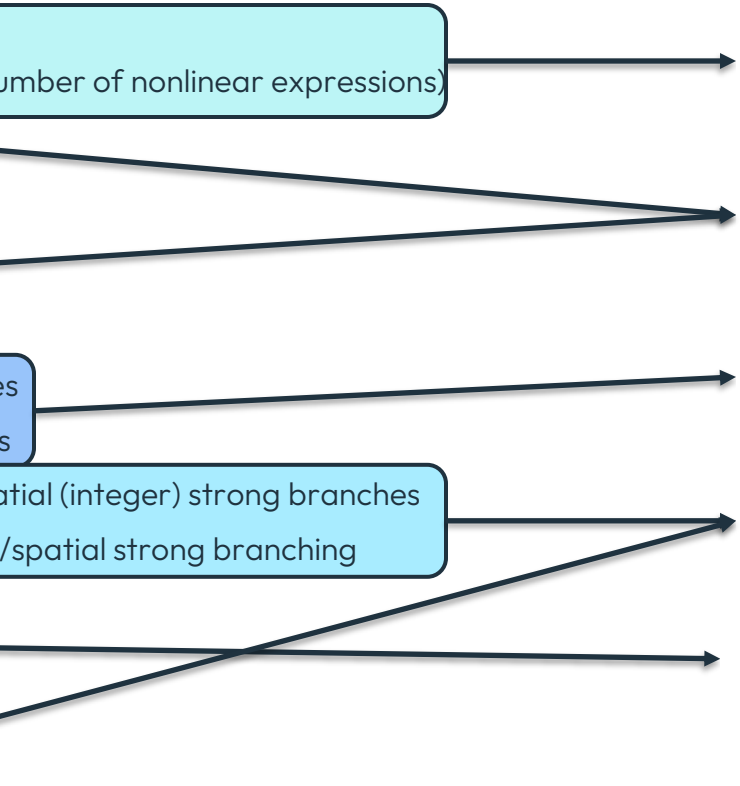
On Which Variable Do We Branch?

- We generally know good rules for branching on integers (in MIP)
  - Strong branching & pseudocost branching
  - Reliability branching
  - Hybrid branching
- We know good rules for branching on spatial branching candidates
  - "What closes the convexification gap the most?"
  - Plus some history information
- But…how do we combine those?
  - Integer violations first, then nonconvexities?
  - Or mix?
  - Different solvers have different answers to that question

⇒ Machine Learning opportunity

Let‘s look for good features

- Constraints
  - Percentage nonlinear constraints
  - Linear nonzeros vs size of the DAG (number of nonlinear expressions)
  - Percentage equality constraints
- Variables
  - Percentage integer variables
- Branching „situation“ at the root node
  - Average number of integer candidates
  - Average number of spatial candidates
  - Average relative bound change of spatial (integer) strong branches
  - Percentage variables fixed by integer/spatial strong branching
- Cost of strong branching
  - Average work
- Numerics of convexification cuts



Nonlinearity of the problem

Combinatorics of the problem

Where is the infeasibility?

Effect of branching

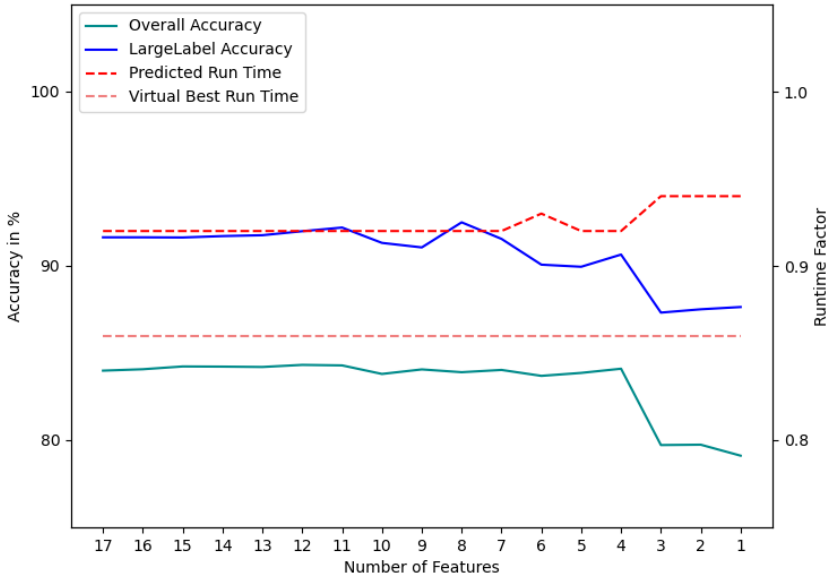
Cost/benefit consideration

Computational Experiments

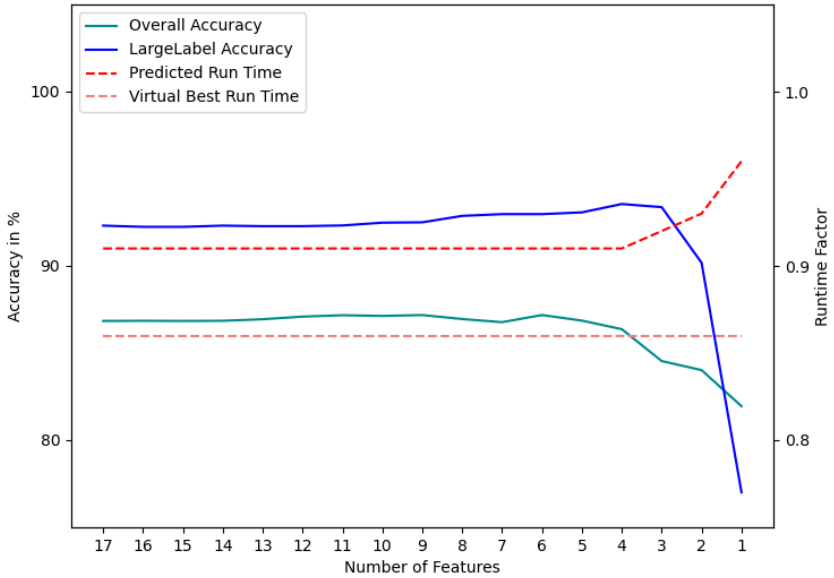
- All features get scaled
- Our models map features to speedup factor: y: $\mathbb{R}^{17} \to \mathbb{R}$
- Label $fac(t_{INT}, t_{MIX}) := \begin{cases} {}^{t_{INT}}/_{t_{MIX}} - 1 \text{ if } t_{INT} \geq t_{MIX} \\ -{}^{t_{MIX}}/_{t_{INT}} + 1 \text{ else} \end{cases}$

  for t_{INT} the runtime with integer-first branching, t_{MIX} the runtime with mixed branching
- Performance Measures:
  - Accuracy: $\text{sign}\big(y(f)\big) = \text{sign}(fac(t_{INT}, t_{MIX}))$, if $|fac(t_{INT}, t_{MIX})| \geq 1.01$
    - LargeLabel Accuracy: for instances with $|fac(t_{INT}, t_{MIX})| \geq 4$
  - RunTime (shifted geometric mean)
    - (virtual best runtime for comparison)
- 683 instances from MINLPLib and FICO customers
- Xpress 9.6, ML models trained with Scikit-learn

Feature Reduction



Linear regression



Random forest regression

- Good: High Accuracy, low runtime
- Performance ~constant when reducing from 17→4 features
- Drops with fewer than four features

Final Performance

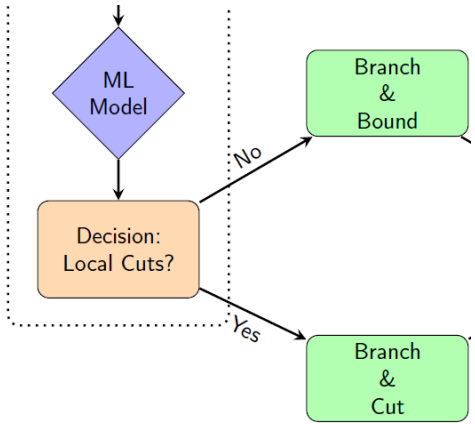
	Linear		Forest	
Accuracy	Train	Test	Train	Test
Overall	84.2%	84.1%	89.1%	86.4%
LargeLabel	91.9%	90.7%	99.7%	93.6%
Time Factor				
Predicted	0.914	0.919	0.884	0.910
Virtual Best	0.862	0.863	0.862	0.863

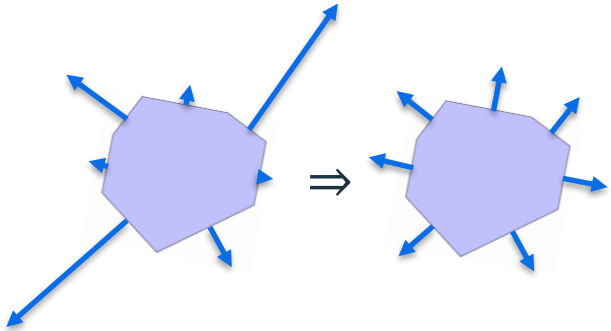
- Both models achieve significant improvements
- Forest shows clear signs of overfitting, linear model not at all
- Linear model closes more than half of the gap to virtual best
- Results transfer to other solver versions
    - 9.6 linear model as good on 9.8 data as model retrained from scratch

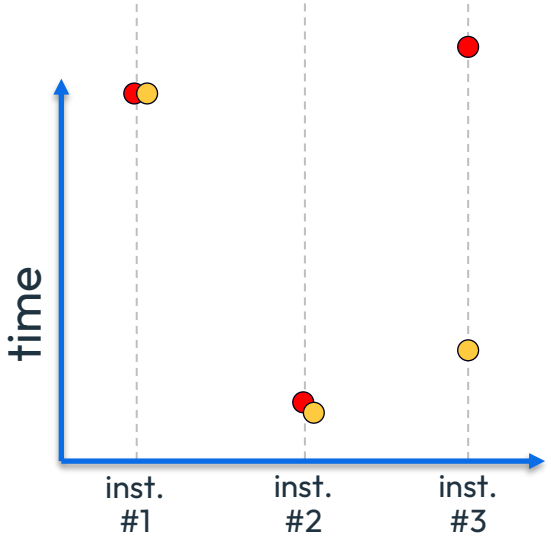
Conclusion

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!

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO

Learning To Select Cuts

Proceedings of CPAIOR 2023,
joint work with Mathieu Besançon, Mark Turner, Thorsten Koch

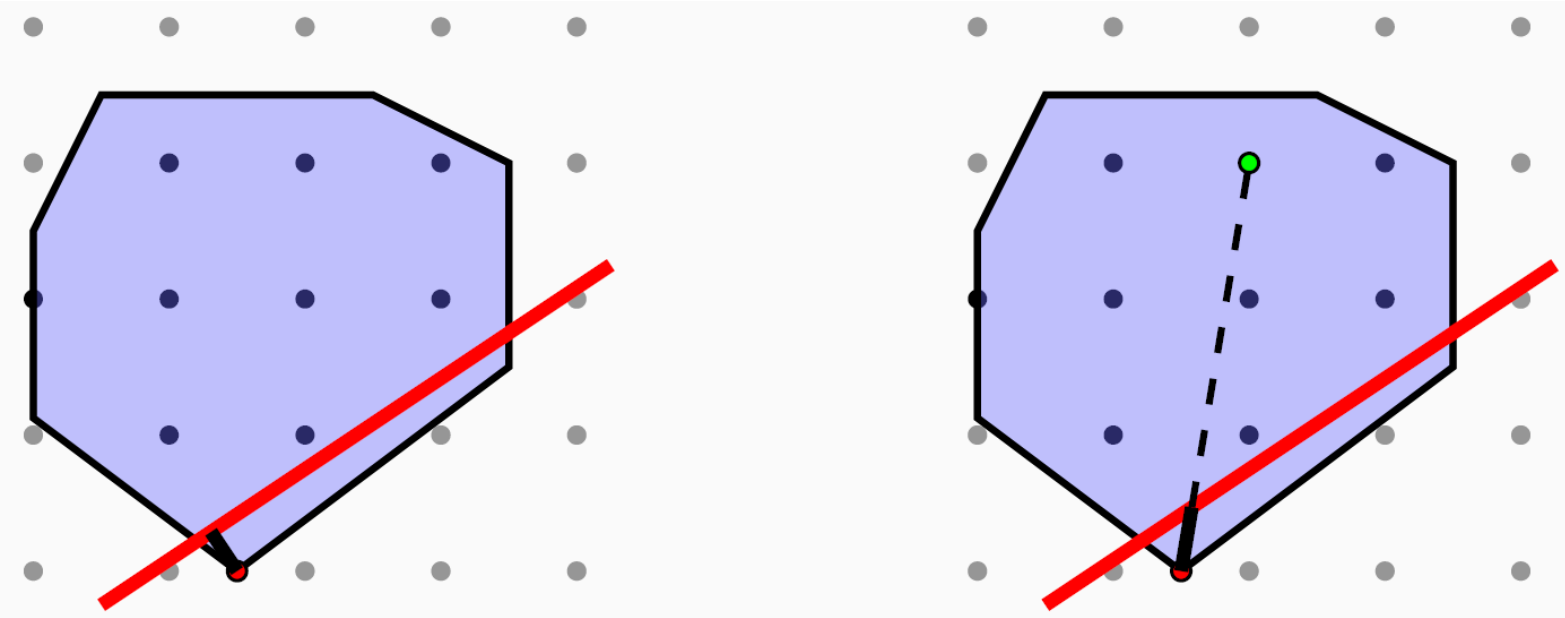
Cut Selection

- 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

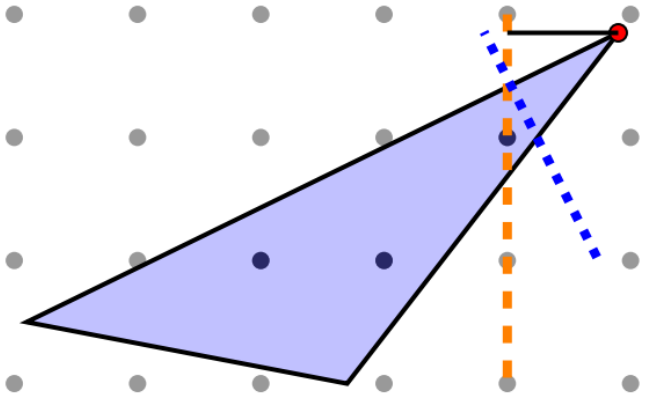


What is wrong with efficacy (and dcd)?

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

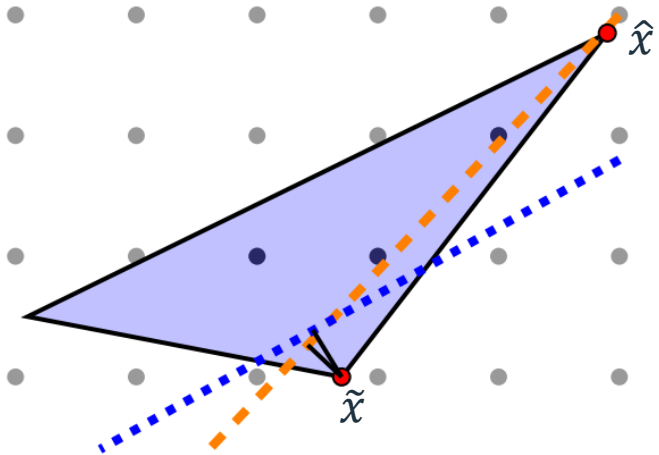
Distances to an infeasible projection

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



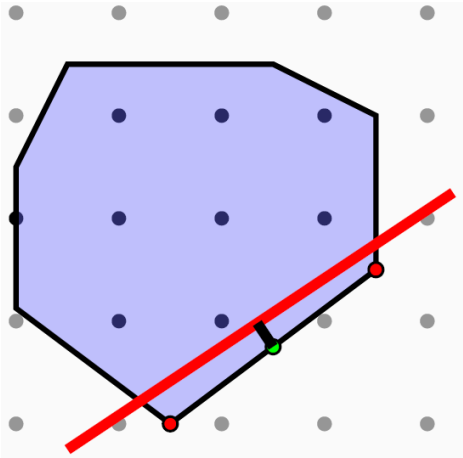
Dual degeneracy (optimal face)

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



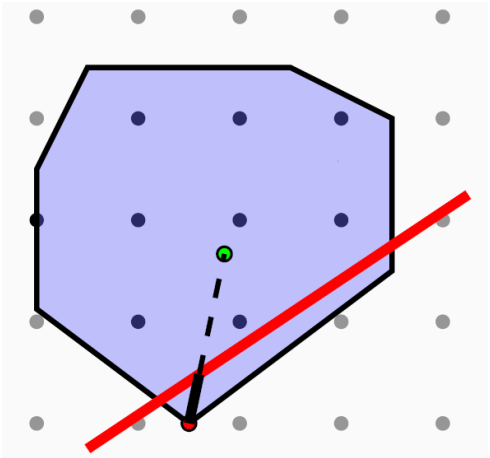
How might we do better?

- 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

FICO
FICO
70

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: $^{\#\text{nodes}_{\text{best}}}/_{\#\text{nodes}}$
- 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