INFORMS WORKSHOP

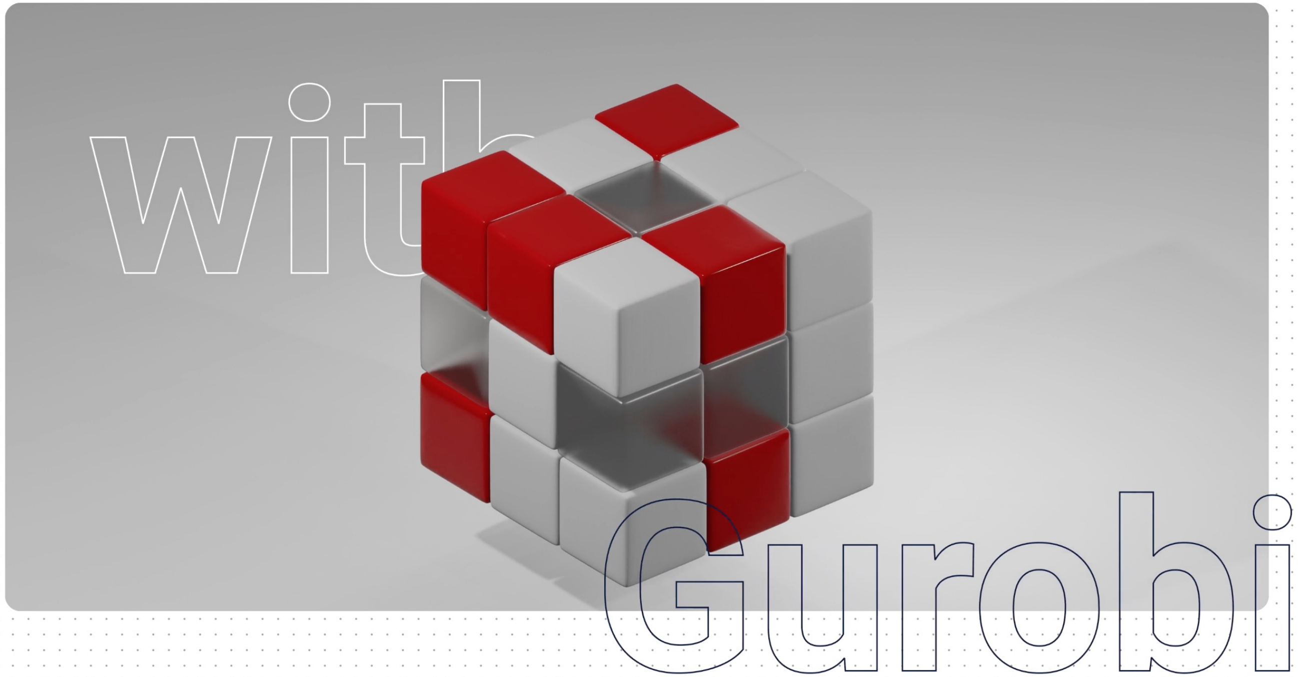
GUROBI 10.0

April 2023

GUROBI
OPTIMIZATION

Agenda

- **Introduction**
  Xavier Nodet
- **10.0 Overview**
  Dan Jeffrey
- **Open Source: Pandas**
  Zed Dean
- **Open Source: Gurobi Machine Learning**
  Xavier Nodet
- **Jupyter Notebook Modeling Examples**
  Jerry Yurchisin
- **Q&A**
  Xavier Nodet
- **Wrapping up: Games, Competitions, and Giveaways**
  Alison Cozad, Lindsay Montanari, Colum Devine



Explainer Video

Telling the Gurobi Story

- Written with a non-technical audience in mind
- Covers the basics of how Gurobi works, in just 60 seconds
- Available on Gurobi YouTube Channel: https://youtu.be/DqjE-P0VyoQ

Optimization

with Gurobi

GUROBI
OPTIMIZATION

Agenda

- **Introduction**
  Xavier Nodet
- **10.0 Overview <--**
  Dan Jeffrey
- **Open Source: Gurobi Machine Learning**
  Xavier Nodet
- **Open Source: Pandas**
  Zed Dean
- **Jupyter Notebook Modeling Examples**
  Jerry Yurchisin
- **Q&A**
  Xavier Nodet
- **Wrapping up: Games, Competitions, and Giveaways**
  Alison Cozad, Lindsay Montanari, Colum Devine

Gurobi 10 Overview

GUROBI
OPTIMIZATION

Performance and API Improvements

Major advances in the underlying algorithmic framework

Enterprise Development & Deployment Experience

New tools for model deployment, monitoring and advanced diagnosis

Innovative Open-Source Projects

Pandas, Machine Learning, and more.

Note: Version & Support Policy Has Changed

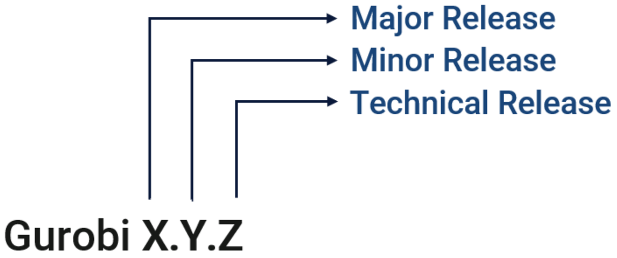
Recent Major/Minor Releases:

10.0: 2022-November

9.5: 2021-November

9.1: 2020-October

9.0: 2019-November



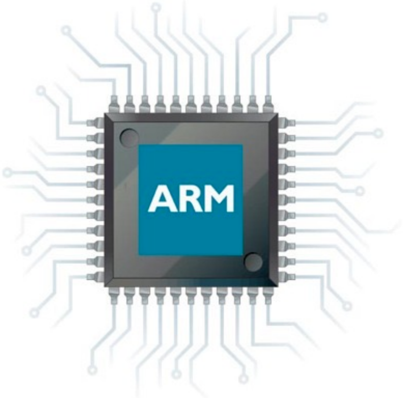
- Previously
  - Annual major or minor release
  - Two major versions supported
    - Along with related minor ones
    - Latest technical release only
- Going Forward
  - Aim for annual major release
  - Quarterly technical release
  - 3 years support for major version
    - Along with related minor ones
    - Latest technical release only
  - *More predictable for customers*

https://support.gurobi.com/hc/en-us/articles/360048138771-Gurobi-release-and-support-history

GUROBI OPTIMIZATION

Linux on ARM Now Supported

- Red Hat® Enterprise Linux 7, 8, 9
- CentOS 7
- SUSE® Enterprise Linux 12, 15
- Ubuntu® 20.04, 22.04
- Amazon Linux 2



Other Supported OS'es

- Linux® x86-64 64-bit
- Windows 64-bit
- MacOS 64-bit Universal
  - Native support for both M- and -x64 chips
- AIX® 64-bit

Performance Improvements In Gurobi 10

- Concurrent solver
- MIP
- Mixed Integer Quadratics
- Non-Convex MIQCP

Poll: Gurobi Versions

The Value of Performance

- What is the immediate business advantage of faster solve times in your solution?
- What would happen if you were able to consider more optimal outcomes in the same time frame?
- How could your solution benefit from increased model complexity or accuracy?

Performance Summary

- New network simplex algorithm
- Concurrent LP improvements:
- Concurrent only on the final presolved model
- Threads used to duplicate work
- Crossover improvements
- New and improved presolve reductions

Compare to Gurobi 9.5

Algorithm	Overall speed-up	On >100sec models
LP – default	10%	25%
LP – primal simplex	3%	10%
LP – dual simplex	3%	10%
MILP	13%	24%
Convex MIQP	57%	2.4x*
Convex MIQCP	28%	88%*
Non-convex MIQCP	51%	2.6x

* MIQP and MIQCP hard model test sets too small to give reliable benchmark results

GUROBI OPTIMIZATION

MIP Performance

- Strong branching
- Symmetry improvements
- Aggressive solving of sub-MIPs
- Presolve reductions
- Optimization-based bound tightening
- Models from machine learning
  - Will be discussed in a later presentation

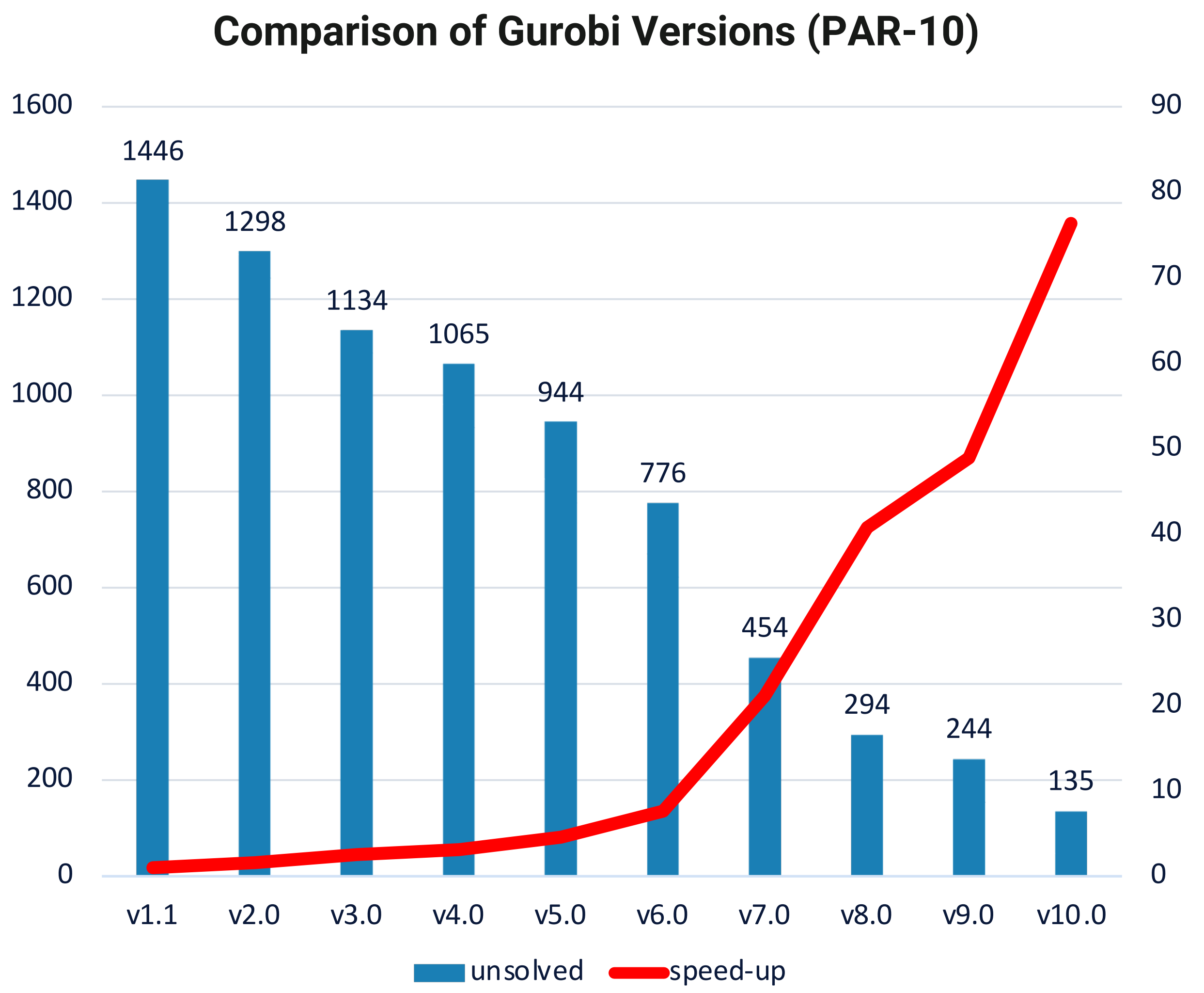
Compare to Gurobi 9.5

Algorithm	Overall speed-up	On >100sec models
LP – default	10%	25%
LP – primal simplex	3%	10%
LP – dual simplex	3%	10%
MILP	13%	24%
Convex MIQP	57%	2.4x*
Convex MIQCP	28%	88%*
Non-convex MIQCP	51%	2.6x

* MIQP and MIQCP hard model test sets too small to give reliable benchmark results

GUROBI
OPTIMIZATION

MIP Performance



GUROBI OPTIMIZATION

MIQ Performance

- New QUBO heuristic
- Perspective strengthening
- Move Q objective terms to constraints
- Work limit adjustment for QC fixing heuristics
- Fix binary in certain order for heuristics
- Solve set covering problem to select linearization
- Remove common variables
- Optimization-based bound tightening
- Many MIP improvements also apply

Compare to Gurobi 9.5

Algorithm	Overall speed-up	On >100sec models
LP – default	10%	25%
LP – primal simplex	3%	10%
LP – dual simplex	3%	10%
MILP	13%	24%
Convex MIQP	57%	2.4x*
Convex MIQCP	28%	88%*
Non-convex MIQCP	51%	2.6x

* MIQP and MIQCP hard model test sets too small to give reliable benchmark results

Non-Convex MIQCP

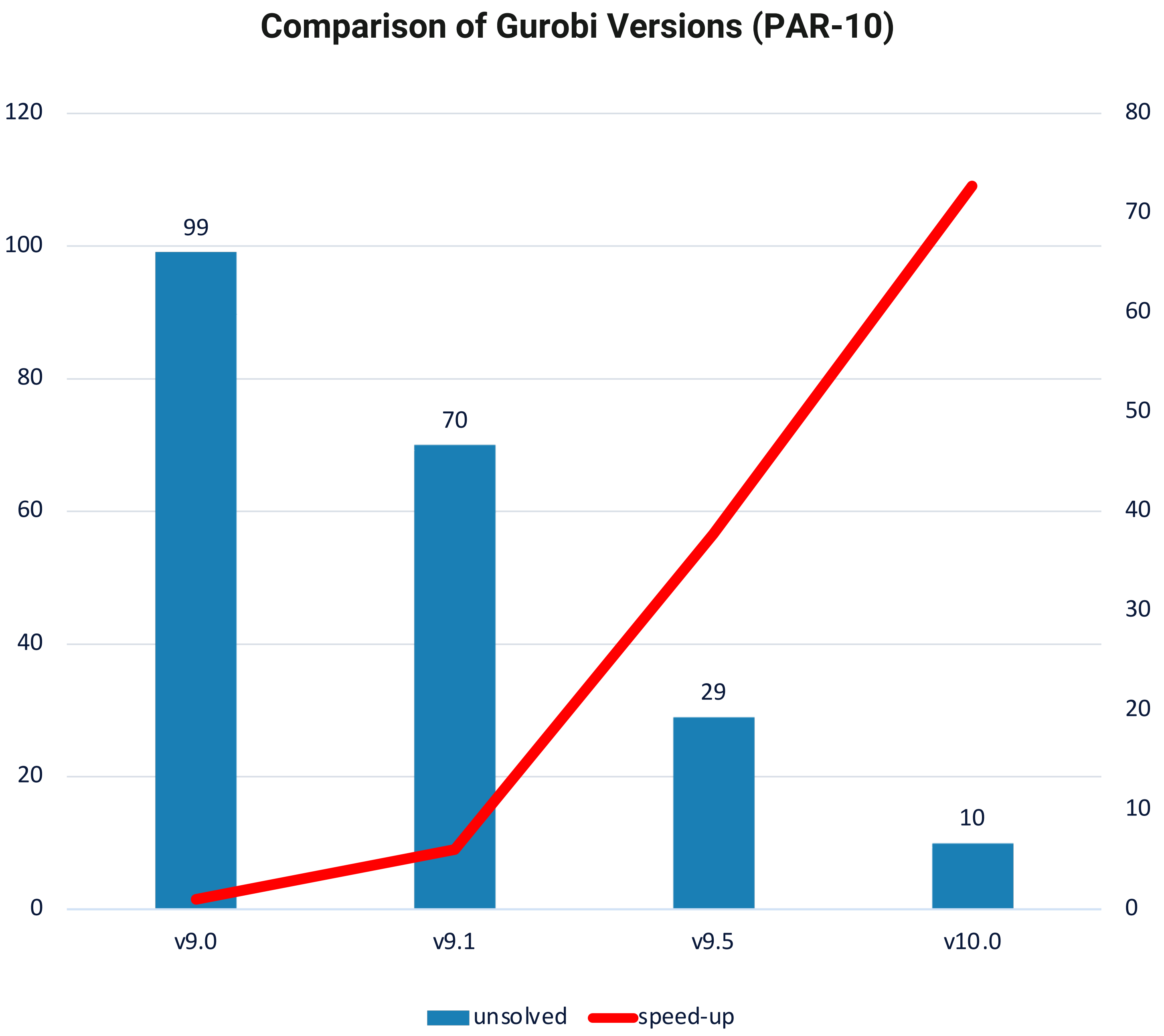
- Optimization-based bound tightening
- Dealing explicitly with bipartite graphs in the product term covering
- Improvement on NLP heuristic termination
- NLP heuristic multi-start
- Many MIP and convex MIQCP improvements also apply

Algorithm	Overall speed-up	On >100sec models
LP – default	10%	25%
LP – primal simplex	3%	10%
LP – dual simplex	3%	10%
MILP	13%	24%
Convex MIQP	57%	2.4x*
Convex MIQCP	28%	88%*
Non-convex MIQCP	51%	2.6x

* MIQP and MIQCP hard model test sets too small to give reliable benchmark results

Non-Convex MIQCP

Time limit: 10,000 sec.
Intel Xeon CPU E3-1240 v5 @ 3.50GHz
4 cores, 8 hyper-threads
32 GB RAM
Test set has 874 models:
- 38 discarded due to inconsistent answers
- 308 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 205 models




Can you use these performance increases in new ways?

API and Engine Improvements In Gurobi 10

- New logistic general constraint
- New Memory limit parameter
- Python matrix API improvements
- NuGet package for .NET
  - Automate Gurobi downloads using NuGet

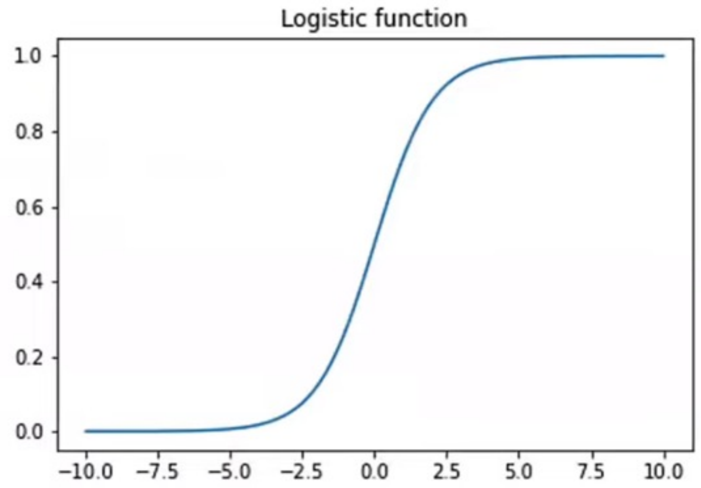
Logistic Constraint

- Background: Function Constraints in Gurobi
  - $y = f(x)$
  - Gurobi generates piecewise-linear approximation
  - Examples: *sine, power, exponential*

- Logistic function added in v10:
  - Applications:
    - ecology, statistics, medicine, chemistry, …
  - Python example:

```
y = 1 / (1 + exp(-x))
gc = model.addGenConstrLogistic(x, y, params)
```

$$p(x) = \frac{1}{1 + e^{-x}}$$



Memory Limit Parameter

- **`SoftMemLimit`**
- Memory limit parameter allows graceful exit
  - Set a memory limit in GB
  - Get best solution
  - Resume optimization
- Contrast with existing MemLimit
  - Memory limit is “soft”
    - Not as strict as MemLimit
    - Might overshoot
  - Both are non-deterministic

```
model.SoftMemLimit=28
```


Matrix API Improvements In Gurobi 10

- Improve usability
- Works well with numpy.ndarray’s
- Name construction
- NumPy functions
- Broadcasting
- Modeling Performance

List-Based vs Matrix-Based

$$\min c^T \cdot x$$
$$s.t.\ A \cdot x = b$$

List-Based API:

```
x1 = [1.0,2.0,3.0]
x2 = [1.0,2.0,0.0]
x3 = [3.0,2.0,0.0]
b = np.array([4,1,2])
c = np.array([1,2,3])
model2 = gp.Model("Pythonic model")
x = model2.addVars(len(b))
model2.setObjective(c[0]*x[0]
                + c[1]*x[1] + c[2]*x[2])
for m in range(b.size):
    model2.addConstr(x1[m]*x[0]
                   + x2[m]*x[1]
                   + x3[m]*x[2] >= b[m])
model.optimize()
for var in model.getVars():
    print(var.X)
```

Gurobi 10 Matrix API:

```
A = np.array([[1.0,2.0,3.0],
              [1.0,2.0,0.0],
              [3.0,2.0,0.0]])
b = np.array([4,1,2])
c = np.array([1,2,3])

model = gp.Model()
x = model.addMVar(3, name="x")
model.setObjective(c @ x)
model.addConstr(A @ x <= b)

model.optimize()

print(f"Result values: {x.X}")
```


Multi-Dimensional Variables, Expressions

- MVar, MLinExpr, MQuadExpr support arbitrary dimensions
  - 9.5 had limited multi-dimensional modeling support

```python
A = np.array([[1.0,2.0,3.0],
              [1.0,2.0,0.0],
              [3.0,2.0,0.0]])
print(A)
```

→

```
[[1. 2. 3.]
 [1. 2. 0.]
 [3. 2. 0.]]
```

```python
x = model.addMVar((3,1))
print(x)
```

→

```
<MVar (3, 1)>
array([[<gurobi.Var C9>],
       [<gurobi.Var C10>],
       [<gurobi.Var C11>]])
```

```python
mLinExpr = A @ x
print(mLinExpr)
```

→

```
<MLinExpr (3, 1)  >
array([[ C9 + 2.0 C10 + 3.0 C11],
       [ C9 + 2.0 C10],
       [ 3.0 C9 + 2.0 C10]])
```


Multi-Dimensional Modeling

Adding constraints → multidimensional MConstr *or MQConstr*

```
A = np.array([[0.3, 0.6, 0.3],
              [0.3, 0.3, 0.2],
              [0.8, 0.8, 0.8],
              [0.6, 0.3, 0.3]])
B = np.array( [[0.2, 0.4, 0.5, 0.4],
                [0.9, 0.3, 0.8, 0.6],
                [0.8, 0.8, 0.1, 0.2],
                [0.3, 0.0, 0.1, 0.6],
                [0.2, 0.4, 0.3, 0.0],
                [0.2, 0.4, 0.7, 0.3]])
X = model.addMVar(3,4)
model.setObjective(X.sum())
mc = model.addConstr(A @ X <= B)
```

→ `<MConstr (6, 4)>`

NumPy Functions for Gurobi Matrix Objects

- gurobipy v10 implements the most-used ones
  - “sum”, “diagonal”, “reshape”
- Extract a diagonal from an MVar X:
  - X.diagonal(offset).
- Convert a list of Var objects to an MVar:
  - x = MVar.fromlist(varlist)
- Sum along an axis of an MVar X:
  - X.sum(axis=...)
- Elementwise squaring of an Mvar X:
  - pow(X, 2), X**2

Name Construction

- Names now show n-dimensional index values

```
z = model.addMVar((3,1), name="Long Name")
model.update()
print(z.VarName)
```

→

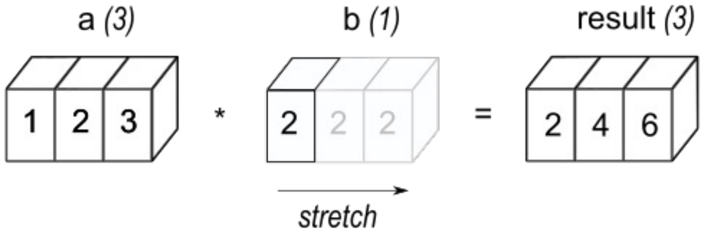
```
[['Long Var Name[0,0]’]
 ['Long Var Name[1,0]’]
 ['Long Var Name[2,0]']]
```


Broadcasting 1

GUROBI OPTIMIZATION

- Operands of different shape:
  - What should the result shape be?
- NumPy Broadcasting Convention

```
model = gp.Model()
a = model.addMVar(3, name = "a" )
b = model.addMVar(1, name ="b")
model.update()
result = a * b
print(result)
```

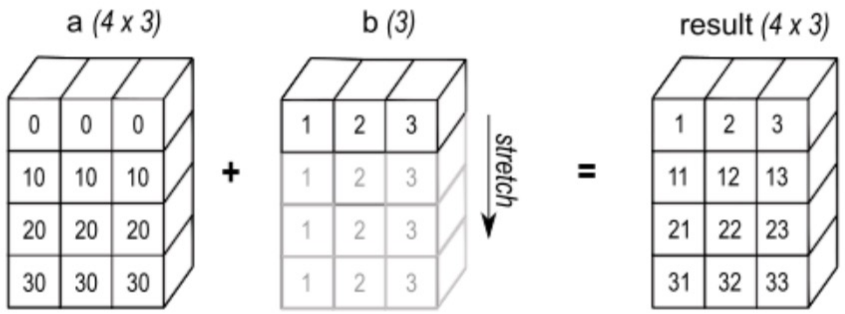



```
<MQuadExpr (3,)>
array([ 0.0 + [ a[0] * b[0] ], 0.0 + [ a[1] * b[0] ], 0.0 + [ a[2] * b[0] ]])
```


Broadcasting 2

- When a binary operation is applied to operands of different shape, what should the result shape be?
- Using NumPy Broadcasting Convention

```
model = gp.Model()
X = model.addMVar((4,3),name="X")
y = model.addMVar((3,),name="y")
model.update()
qe = X + y
print(qe)
```




```
<MLinExpr (4, 3)> array([[ X[0,0] + y[0], X[0,1] + y[1], X[0,2] + y[2]],
                         [ X[1,0] + y[0], X[1,1] + y[1], X[1,2] + y[2]],
                         [ X[2,0] + y[0], X[2,1] + y[1], X[2,2] + y[2]],[
X[3,0] + y[0], X[3,1] + y[1], X[3,2] + y[2]]])
```


Matrix API Performance Comparison

```python
m = 1024; n = 8192; d = 0.2
A = sp.random(m, n, d, format='csr')
b = np.random.rand(m)
# Using MVar
x = model.addMVar(n)
model.addConstr(A@x == b)
# Iterate One List
x = model.addVars(n).values()
for i in range(m):
    (_, colidx, colcoef) = sp.find(A[i, :])
    le = gp.LinExpr(colcoef, [x[j] for j in colidx])
    model.addConstr(le == b[i])
# Iterate Two Lists
(rowidx, colidx, coef) = sp.find(A)
x = model.addVars(n).values()
le = [gp.LinExpr() for i in range(m)]
for k in range(len(coef)):
    le[rowidx[k]] += coef[k] * x[colidx[k]]
for i in range(m):
    model.addConstr(le[i] == b[i])
```

Results

Using MVar: 0.083sec.

One List: 1.729sec.

Two Lists: 53.77sec.

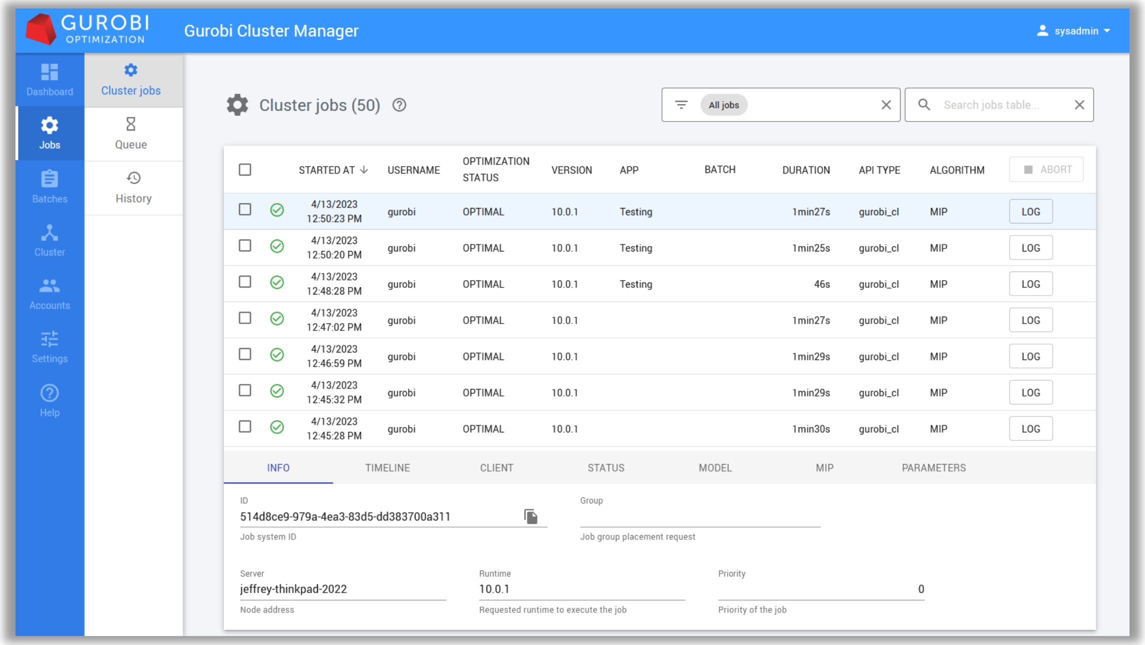
Enterprise Development & Deployment Experience

- Compute Server Improvements
- New Licensing Options

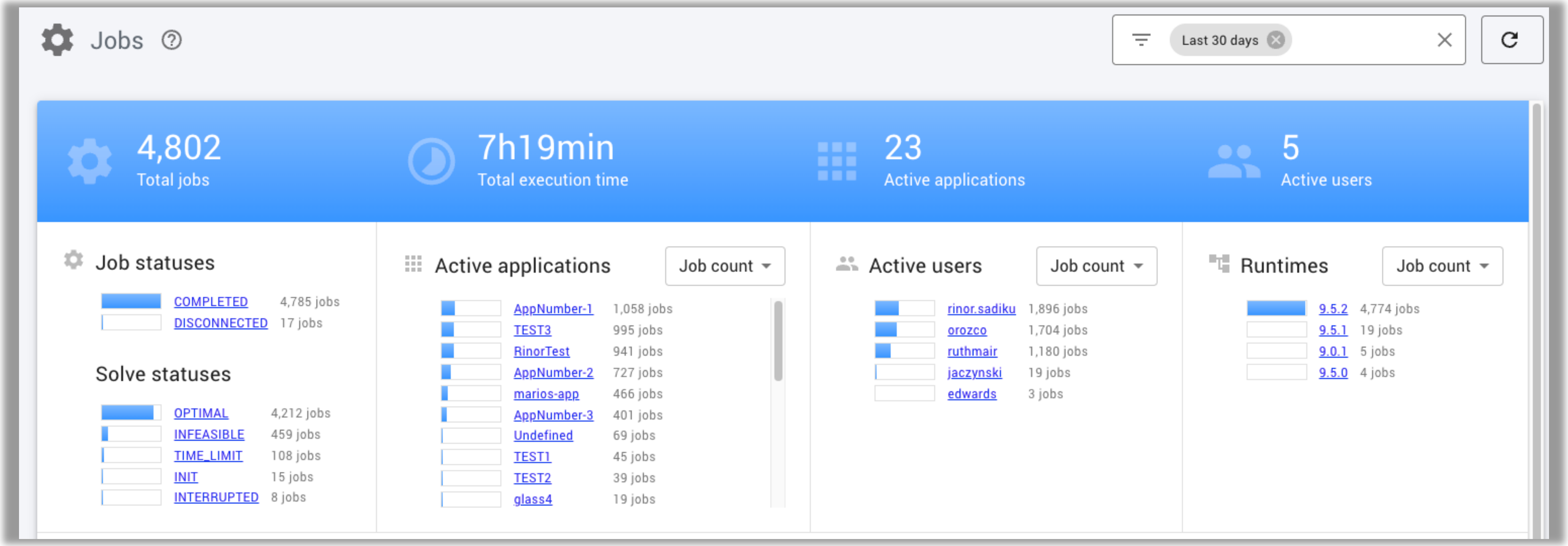
New Platform Features

- Compute Server
    - Background
        - Gurobi in a service architecture
        - Web UI, security, queuing, more
    - Cluster Manager Improvements
        - Job Dashboard
        - Node Dashboard

- Web License Service
    - Now available on all environments



Job Dashboard 1



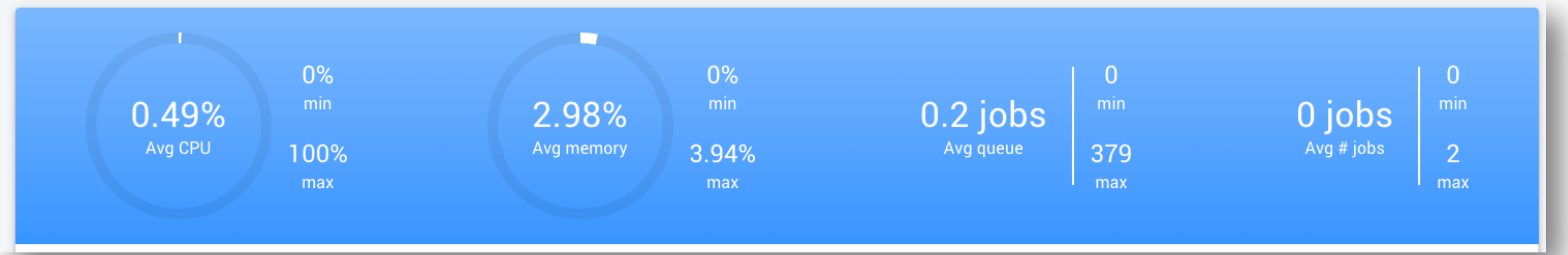
- Global Metrics
  - number of jobs, execution time, active application, active users
- Predefined filters for last 24h, 7 days or 30 days
  - More filtering available
- Distributions
  - Shown in several dimensions
  - Job and solve statuses, applications, users, runtimes
  - Drill down to job list

Job Dashboard 2

- Timeline by several dimensions
  - Applications, Job/Solve statuses, Users, Runtime and solve times
  - Zoom and pan over time
  - Legend and colors to differentiate values
- Drilldown
  - Go to the job list of the selected period
  - Filter the dashboard with the selected period

Node Dashboard

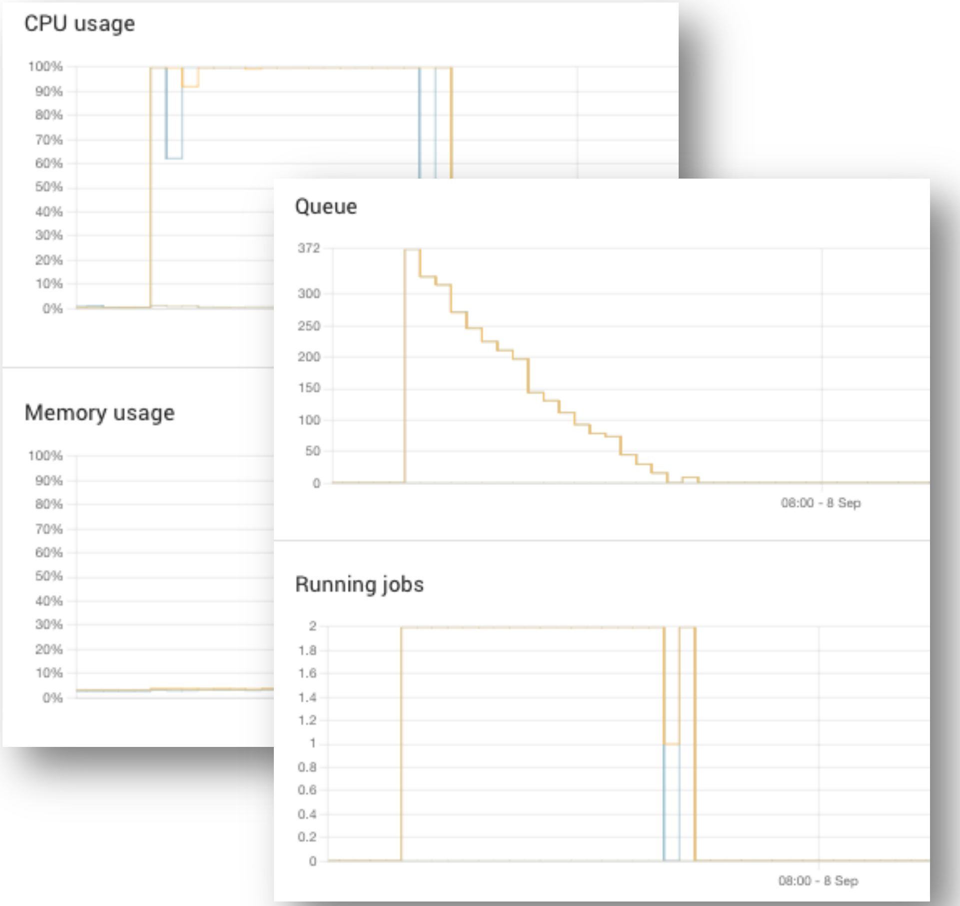
GUROBI OPTIMIZATION



- Global Overview
  - CPU, Memory,
  - Job in queue and running
- Predefined filters for last 24h, 7 days or 30 days

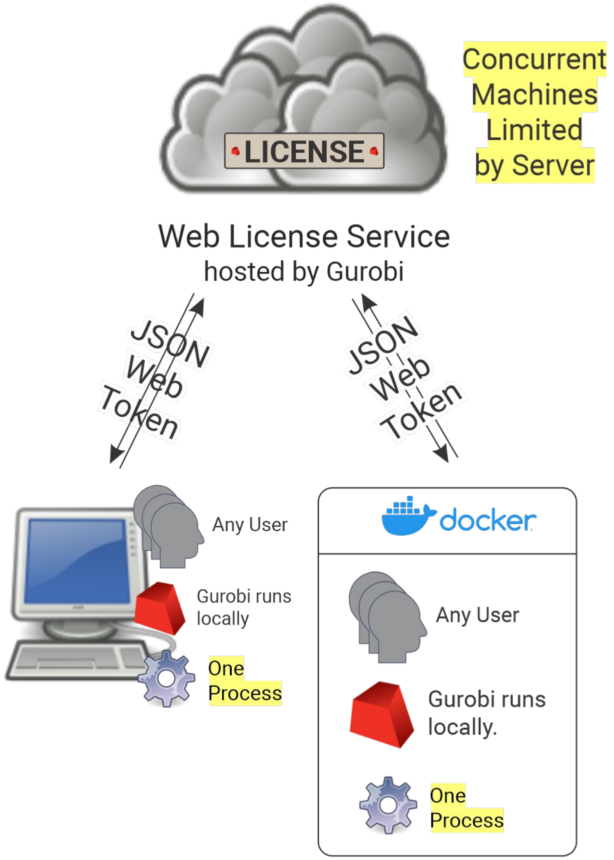
Node Dashboard Detail

- Timeline
  - CPU usage
  - Memory usage
  - Job in queue
  - Running jobs
- Drilldown
  - Zoom, pan
  - Node selection

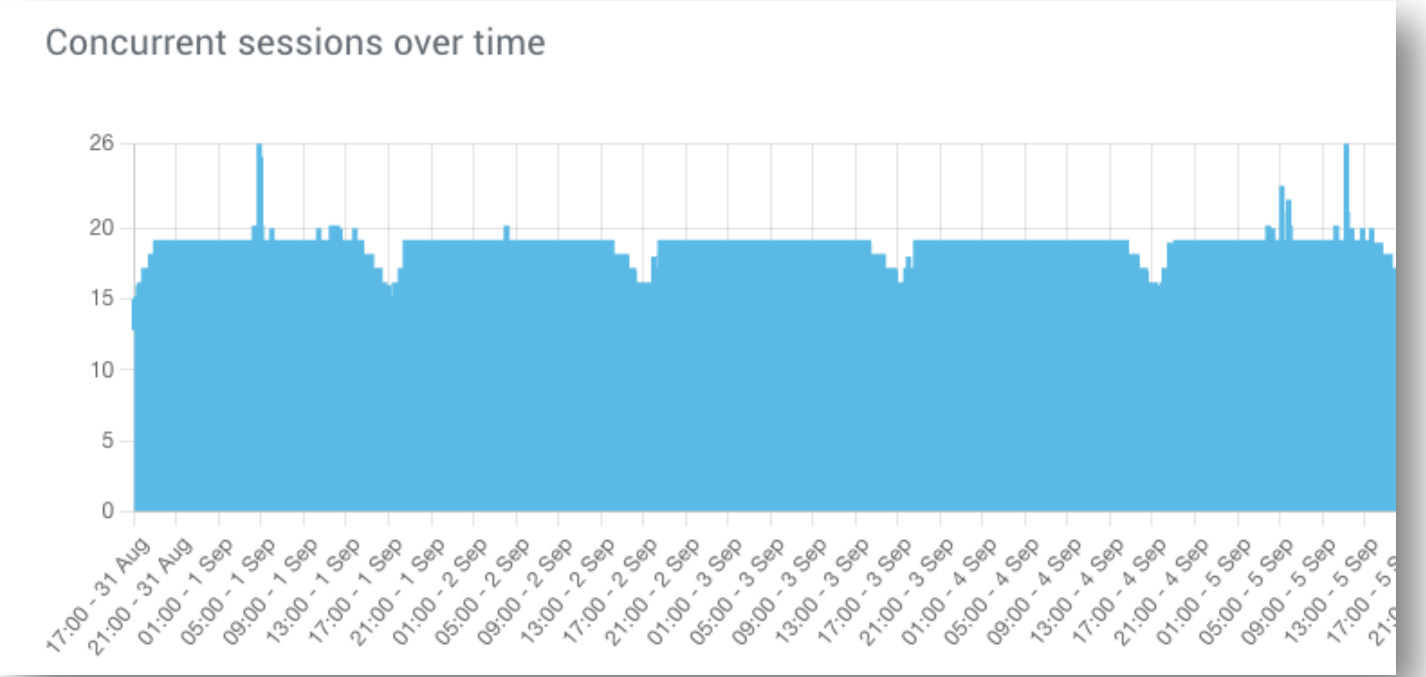


New WLS Licensing Options

- WLS = Web License Server
  - Gurobi-hosted license server
  - Limits or counts:
    - Cores
    - Concurrent solves
  - Use from anywhere
  - Limited communication from client
    - Hostname, IP Address, Core count, OS, Version
    - No model data
- New options for physical machines & VM's
  - Physical Machines (Linux, Windows, Mac)
  - Virtual Machines
  - Containers



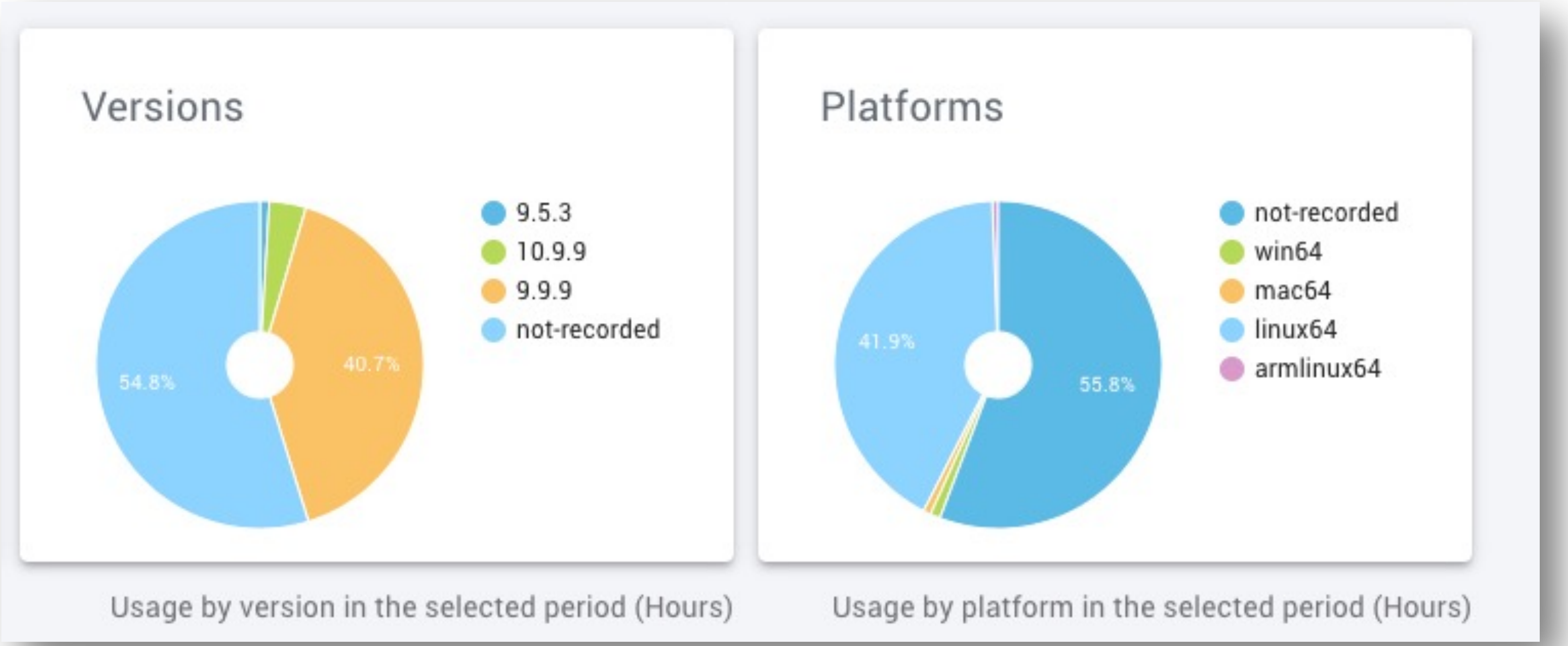
WLS Sessions



- Reports now show sessions instead of containers
- Sessions = Continuous usage of Gurobi in a container or a machine

WLS Metrics

- Reports of versions and platforms



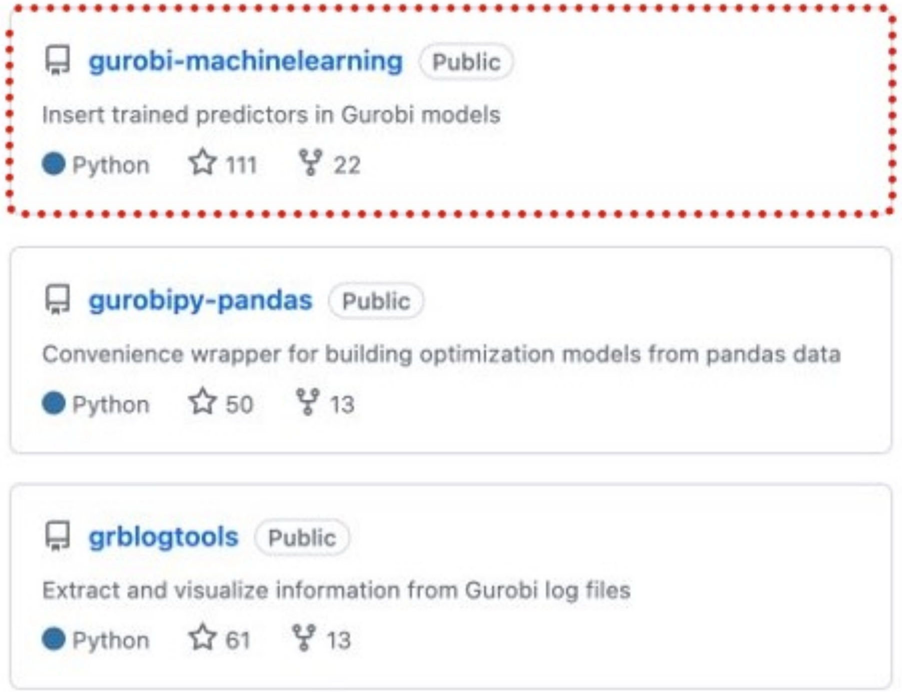
Innovative Open-Source Projects

Gurobi 10.0 – Open-Source GitHub Repositories

- Gurobi Machine Learning
  - Use machine learning model as MIP constraint
  - Special session later in this workshop
- Gurobipy pandas
  - Gurobipy model building patterns with pandas
  - Separate session later in this workshop

- Coming soon...
  - Gurobi OptiMods
    - Optimization modules for specific applications
  - Numerical issues assessment tool
    - Analyze root cause of numerical issues

github.com/Gurobi



Summary

Summary

Performance and API Improvements

Major advances in the underlying algorithmic framework

Enterprise Development & Deployment Experience

New tools for model deployment, monitoring and advanced diagnosis

Innovative Open-Source Projects

Pandas, Machine Learning, and more.

Gurobipy-pandas
Pandas

Enables convenient gurobipy model building patterns with pandas

Zed Dean
Technical Account Manager

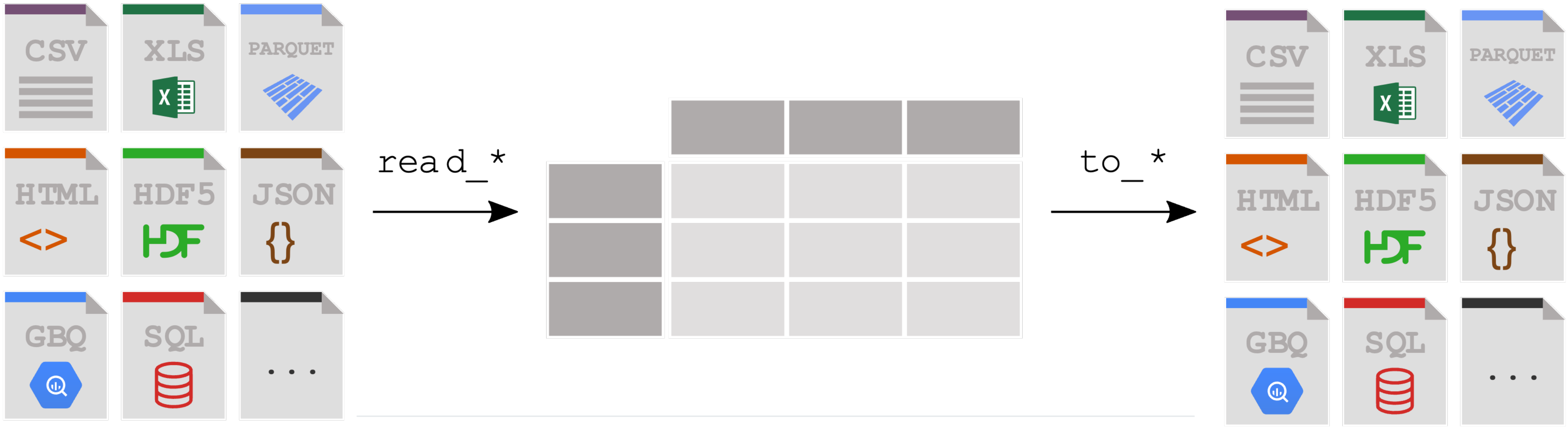
GUROBI
OPTIMIZATION

GUROBI
OPTIMIZATION

Agenda

- Rationale
- What is gurobipy-pandas ?
- Who is it for ?
- Installation & dependencies
- Usage Jupyter Notebook
- Free Function vers Pandas accessor
- Arithmetic Expressions
- Performance

How do I read and write tabular data with Gurobi ?



Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

Source:How do I read and write tabular data? — pandas 2.0.0 documentation (pydata.org)

Our Goal

- Simplify the process of importing Data Tables into optimization model.

What is gurobipy-pandas

gurobipy-pandas is a convenient (optional) wrapper to connect pandas with gurobipy..

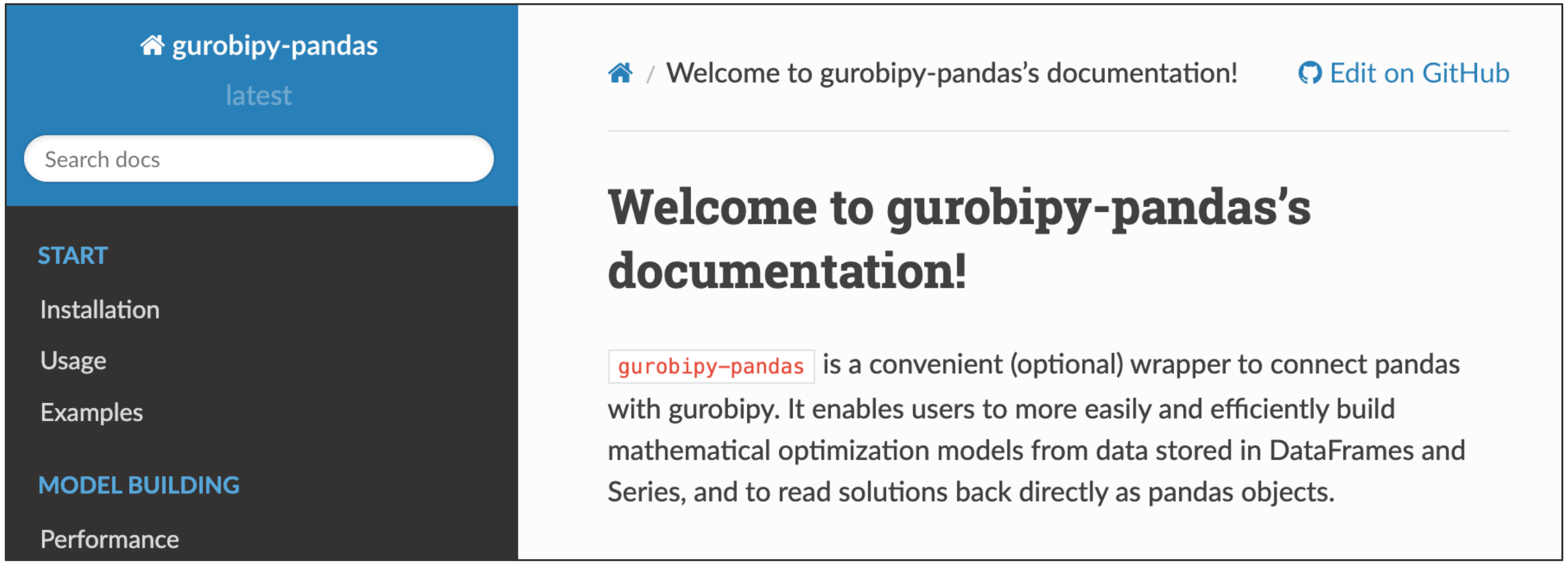
Audience

gurobipy-pandas is aimed at experienced pandas users who are familiar with methods to transform, group, and aggregate data stored in dataframes. It expects some familiarity with optimization modelling, but does not require deep experience with gurobipy

Gurobipy and pandas

Documentation and examples for users of the PyData stack

- Open source, with documentation available on readthedocs.com
  - Github repository: https://github.com/gurobi/gurobipy-pandas
  - Documentation: https://gurobi-optimization-gurobipy-pandas.readthedocs-hosted.com
- Complete model building examples as Jupyter notebooks
- Guidance for writing performant gurobipy-pandas code



gurobipy-pandas Dependencies & Installation

GUROBI OPTIMIZATION

Dependencies

- gurobipy: Python modelling interface for the Gurobi Optimizer
- pandas: powerful Python data analysis toolkit

- !pip install gurobipy-pandas

Standard Imports

- Most gurobipy-pandas applications will start with the following imports.

```
>>> import pandas as pd
>>> import gurobipy as gp
>>> from gurobipy import GRB
>>> import gurobipy_pandas as gppd
```


Usage

A mathematical optimization model has five components, namely:

- Sets and indices.
- Parameters.
- Decision variables.
- Constraints
- Objective function(s).

- Optimization models define all data, variables, and constraints over indexes:

$$
\begin{aligned}
\max \quad & \sum_{i \in I} c_i x_i \\
\text{s.t.} \quad & \sum_{i \in I} a_i x_i \leq b \\
& x_i \in \{0, 1\} \quad \forall i \in I
\end{aligned}
$$

- These mathematical indices provide a clear way to structure data in code.

Gurobipy and pandas

Easier model building with the popular Python data analytics package

GUROBI OPTIMIZATION

- Create pandas Series and DataFrames of Gurobi variables
- Use pandas operations to combine variables and data into constraints
- Extract solution data as pandas Series
- No need to manually translate between pandas and gurobipy!

```
In [14]: x = gppd.add_vars(model, allowed_pairs, vtype=GRB.BINARY, obj="value", name="x")
         x.head()

Out[14]: project  team
         0        4       <gurobi.Var x[0,4]>
         1        4       <gurobi.Var x[1,4]>
         2        0       <gurobi.Var x[2,0]>
                  1       <gurobi.Var x[2,1]>
                  2       <gurobi.Var x[2,2]>
         Name: x, dtype: object
```


© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 53

Usage Jupyter Notebook Demo

- Create Model
- Create variables
- Use Pandas’s built-in Expressions
- Create Constraints
- Set Objective
- Solve
- Extract solutions

Two different ways to build the model with gppd

GUROBI OPTIMIZATION

Using Free functions

- **varsname=gppd.add_vars**(*model, pandas_obj, *, name=None, lb=0.0, ub=1e+100, obj=0.0, vtype='C', index_formatter='default'*)¶
- **contraintsname=gppd.add_constrs**(*model, lhs, sense, rhs, *, name=None, index_formatter='default'*)

Using Methods

- **Dataframe.gppd.add_vars**(*model, *, name, lb=0.0, ub=1e+100, obj=0.0, vtype='C', index_formatter='default'*)
- **add_constrs**(*model, lhs, sense=None, rhs=None, *, name, index_formatter='default'*)

Arithmetic Expressions

- - Pandas handles this for us
- - We always leverage pandas-native functions
- - Common operations:
-     - summation
-     - arithmetic operations
-     - groupby (split-apply-combine) operations
- - Let's explore some common mathematical expressions

Performance

`gurobipy-pandas` won't magically make your model building code fast

- Best used where your inputs and outputs *already* use pandas
- Data *must* be properly-organized and prepared
- Prescribed style aims to *avoid performance pitfalls*

- Performance — gurobipy-pandas 1.0.0 documentation (readthedocs-hosted.com)

Thank You

For more information: gurobi.com

Zed Dean
Technical Account Manager

Gurobi Machine Learning

Using trained machine learning predictors in Gurobipy

Xavier Nodet
Development Manager

GUROBI
OPTIMIZATION

Agenda

Motivating example

Gurobi Machine Learning

Other use-cases

Gurobi 10.0

Motivating example

Optimizing avocado shipments and prices

Motivating Example: Price Optimization

- Selling avocados in the US
  - Market is split in regions R
  - Total supply S
  - Maximize profit:
    - (sales – shipping costs – unsold penalty)

  with given
    - shipping costs c_r and waste penalty w

- We decide about the prices p_r
- and estimate the demands d_r from the prices

https://youtu.be/AJRP9pPBx6s

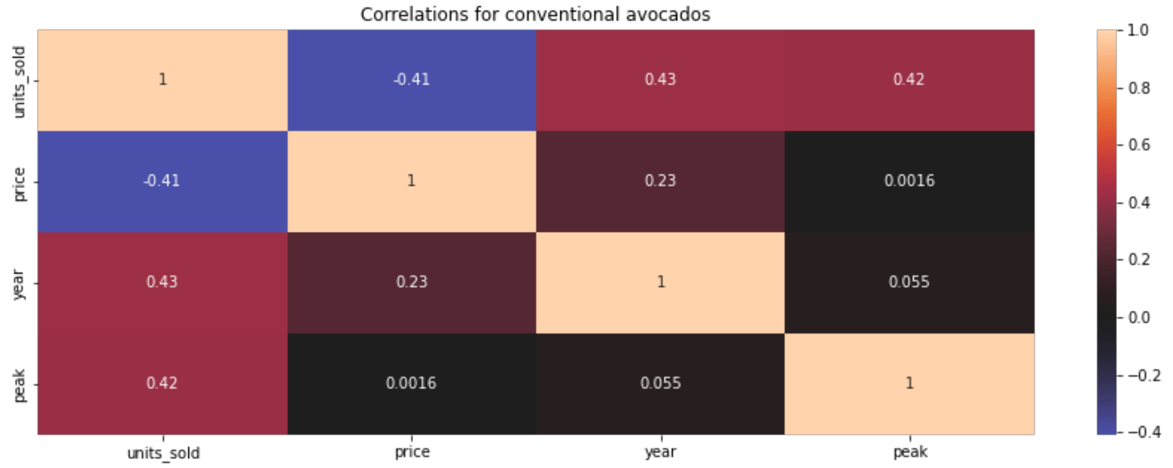
Motivating Example: Estimating Demand

- Historical data of avocado sales from Hass Avocado Board (HAB) available on Kaggle and HAB website
- Features correlated to demand: year, peak season, region, price
- Linear regression gives reasonably good prediction of demand with those:

$$d = g(year, peak, r, p)$$

- In the case of linear regression, g is an affine function

$$d = \phi^T(year, peak, r, p) + \phi_0$$



Motivating Example: Retrieving ML results

- Retrieve results from linear regression model

```
coef_dict = model.fit().params.to_dict()
```

- Build linear expression defining expected demand from price

```
d = {r: (coef_dict['Intercept'] +
         coef_dict['price'] * p[r] +
         coef_dict['C(region)[T.%s]'%r] +
         coef_dict['year_index'] * (year - 2015) +
         coef_dict['peak'] * peak_or_not)
     for r in R}
```

- Upper bound on sales is expected demand

```
m.addConstrs((s[r] <= d[r] for r in R))
```


Motivating Example: Optimization Model

- Indices
  - $\mathrm{r} \in R$, the set of regions
- Constants
  - c_r transportation costs
  - w penalty for waste
  - S total supply
  - Year and season of interest

- Variables
  - $\mathrm{p_r}$ selling price per unit
  - $\mathrm{d_r}$ demand
  - u total unsold products

$\max \sum_r (p_r - c_r) d_r - w * u$ *(maximize revenue)*

$s.t.$

$\sum_r d_r + u = S$ *(allocate shipments)*

$d_r = g(year, peak, r, p_r)$ for $\mathrm{r} \in R$ *(define demand with regression model)*

Motivating Example: What if?

- With g an affine function, the resulting model is a non-convex QP
- Solved fast with Gurobi
- But what if we need a more accurate prediction with a more complex regression:
  - Decision tree, Neural network, …

$\max \sum_r (p_r - c_r) d_r - w * u$ *(maximize revenue)*

$s.t.$

$\sum_r d_r + u = S$ *(allocate shipments)*

$d_r = g(year, peak, r, p_r)$ for $r \in R$ *(define demand with regression model)*

Motivating Example: Goals

1. Simplify the process of importing a trained machine learning model built with a popular ML package into an optimization model.

2. Improve algorithmic performance to enable the optimization model to explore a sizable space of solutions that satisfy the relationships between variables captured with the ML model.

Gurobi Machine Learning

An open-source Python package

Gurobi Machine Learning

- Open source python package
- https://github.com/Gurobi/gurobi-machinelearning
- https://gurobi-machinelearning.readthedocs.io/
- Apache License 2.0
- Initial release 1.0.0 last November
- Version 1.1.1 recently two days ago

- Experimental package
- Not supported through usual Gurobi processes

Known regression models

GUROBI OPTIMIZATION

scikit learn

- Linear/Logistic/PLS regression
- Decision trees
- Neural network with ReLU activation
- Random Forests
- Gradient Boosting
- Preprocessing:
  - Simple scaling
  - Polynomial features of degree 2
  - Column transformers
- pipelines to combine them

K Keras

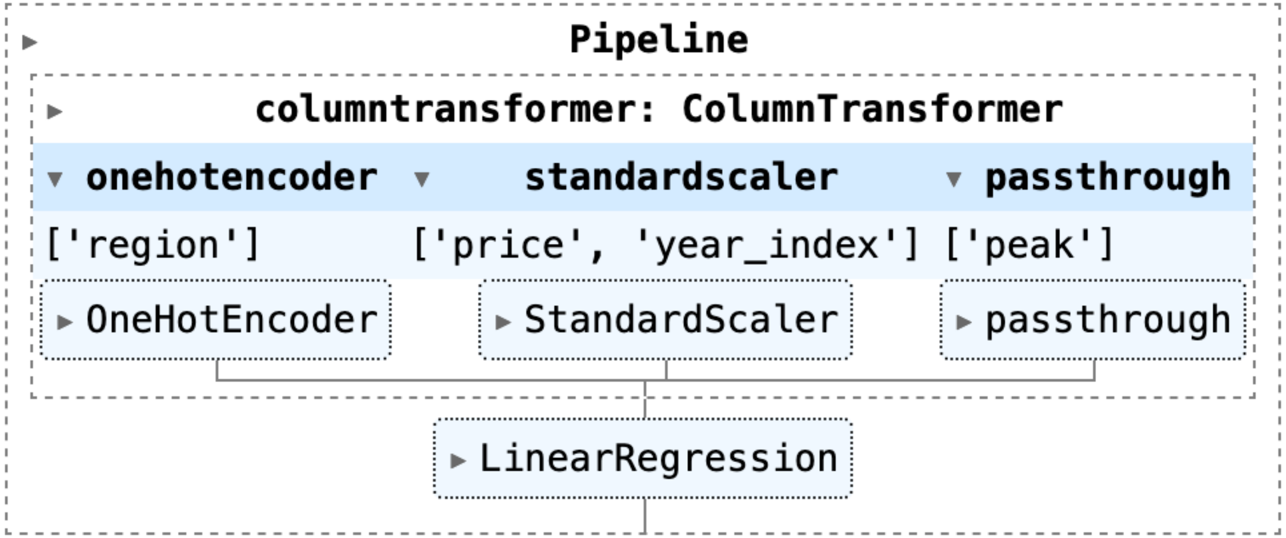
- Dense layers
- ReLU layers
- Object Oriented, functional or sequential

PyTorch

- Dense layers
- ReLU layers
- Only torch.nn.Sequential models

Example: Define and train ML model

```
...
lin_reg = make_pipeline(columntransformer, LinearRegression())
lin_reg.fit(X_train, y_train)
```



Example: Creating the variables

```
import gurobipy_pandas as gppd

data = pd.concat(...)

m = gp.Model("Avocado_Price_Allocation")
p = gppd.add_vars(m, data, lb=0.0, ub=2.0)
d = gppd.add_vars(m, data)
u = m.addVar()


m.setObjective(((p - c) * d).sum() - w * u, GRB.MAXIMIZE)
m.addConstr(d.sum() + u == S)
```

- Variables
  - p_r selling price per unit
  - d_r demand
  - u total unsold products

Example: Adding regression constraints

$$d_r = g(year, peak, r, p_r) \text{ for r} \in R.$$

```
fixed = pd.DataFrame(
    data={
        "year": 2020,
        "peak": 1,
        "region": regions,
    },
    index=regions)
feats = pd.concat(
    [fixed, p],
    axis=1)
```

	year	peak	region	price
Great_Lakes	2020	1	Great_Lakes	<gurobi.Var price[Great_Lakes]>
Midsouth	2020	1	Midsouth	<gurobi.Var price[Midsouth]>
Northeast	2020	1	Northeast	<gurobi.Var price[Northeast]>
Northern_New_England	2020	1	Northern_New_England	<gurobi.Var price[Northern_New_England]>
SouthCentral	2020	1	SouthCentral	<gurobi.Var price[SouthCentral]>
Southeast	2020	1	Southeast	<gurobi.Var price[Southeast]>
West	2020	1	West	<gurobi.Var price[West]>
Plains	2020	1	Plains	<gurobi.Var price[Plains]>

Example: Adding Regression Constraints

```
from gurobi_ml import add_predictor_constr

pred_constr = add_predictor_constr(m, lin_reg, feats, d)
pred_constr.print_stats()
```

```
Model for pipe:
88 variables
24 constraints
Input has shape (8, 4)
Output has shape (8, 1)

Pipeline has 2 steps:

--------------------------------------------------------------------------------
Step            Output Shape    Variables              Constraints
                                                Linear    Quadratic      General
================================================================================
col_trans            (8, 10)           24           16            0            0

lin_reg               (8, 1)           64            8            0            0

--------------------------------------------------------------------------------
```

Example: Optimizing

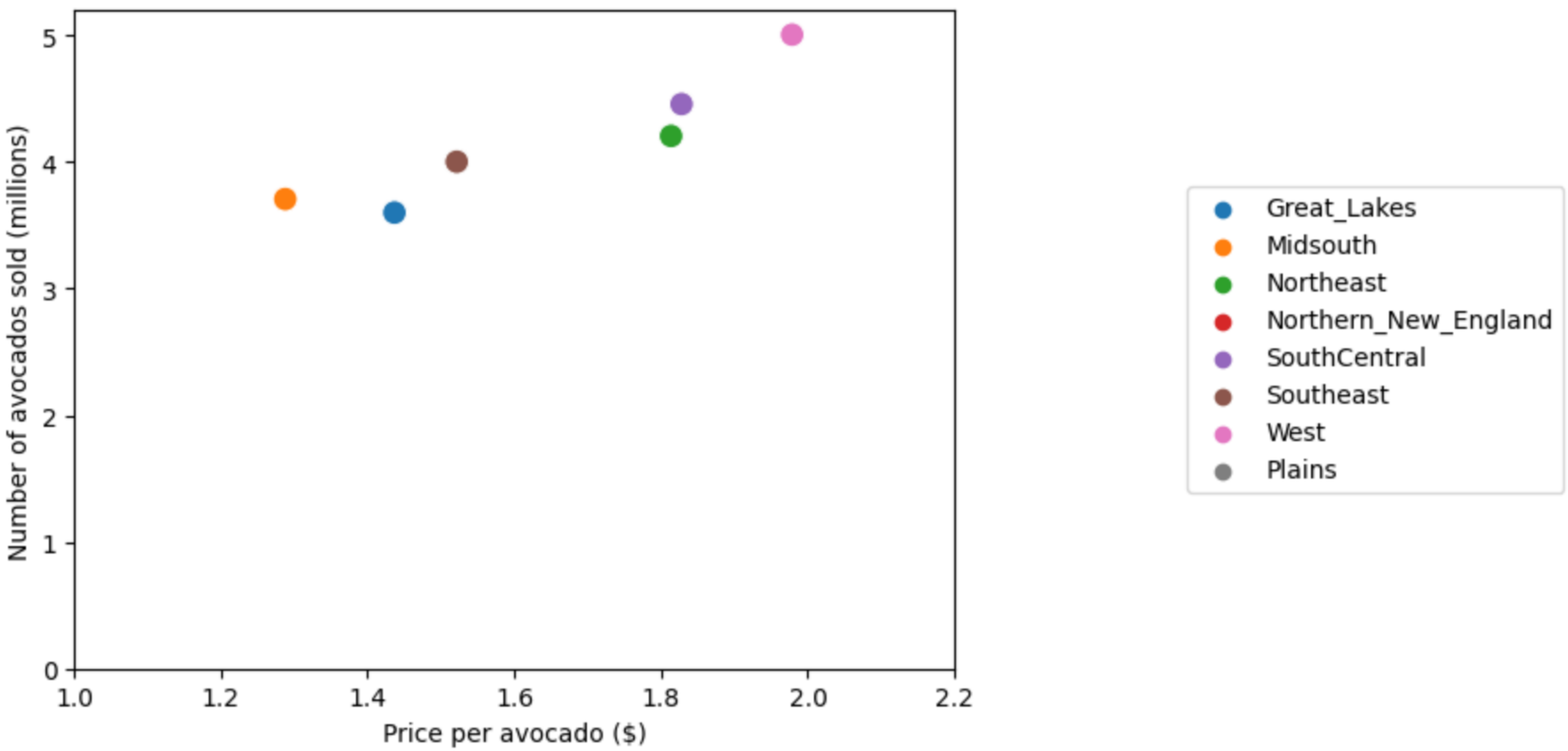
```
m.Params.NonConvex = 2
m.optimize()
```

```
Explored 1 nodes (75 simplex iterations) in 0.04 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 2: 38.7675 36.5918

Optimal solution found (tolerance 1.00e-04)
Best objective 3.876747585682e+01, best bound 3.876937455959e+01, gap 0.0049%
```


Example: Solution




```
Optimal net revenue: 38.1 million, unsold avocados: 0.34 millions
```


Remarks

- Function `add_predictor_constr` creates the formulation for regression model and returns a *modeling object*.
- If input of `add_predictor_constr` has several rows, it introduces one sub-model for each row
- Query statistics about modeling object, remove it, query solution after solve and error in solutions.
- Models for logistic regression use a piecewise linear approximation and can have modeling error (controlled by parameters).
- Models for decision tree can also introduce small errors at threshold values of node splitting (can be controlled).

GUROBI OPTIMIZATION

Other use-cases

More reasons to use
`gurobi machine learning`

Student enrollment

- Train a ML model to predict student enrollment in college from scholarship offered (“merit”), SAT and GPA scores.
- Build an optimization model to decide:
  - Scholarship offered to each student
- To maximize
  - The (predicted) number of students enrolled
- Subject to:
  - Budget constraint and bounds on individual merit.

- Use `gurobi machine learning` to set the value of the *enroll* variable, given the *merit* offered and the SAT and GPA scores of each student

Surrogate models

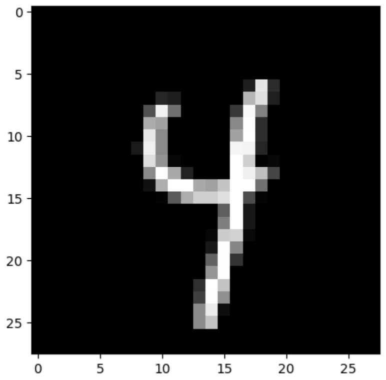
- When the modeling of a complex process requires highly non-linear functions, or simulation
- Approximate this function using e.g. a Neural Network with ReLU activation
- Use `gurobi machine learning` to create a ‘surrogate constraint’

- For a network with polynomial features, the resulting model is a non-convex QCP

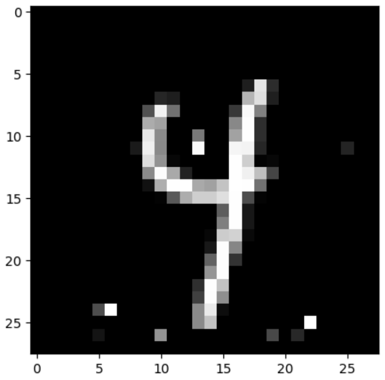
Adversarial Machine Learning

- Suppose we have a trained neural network,
- And a well-classified example x
- Create another example $\bar{x}$ close to x that is classified with a different label

- Example from the MNIST handwritten digits database



Classified as a 4



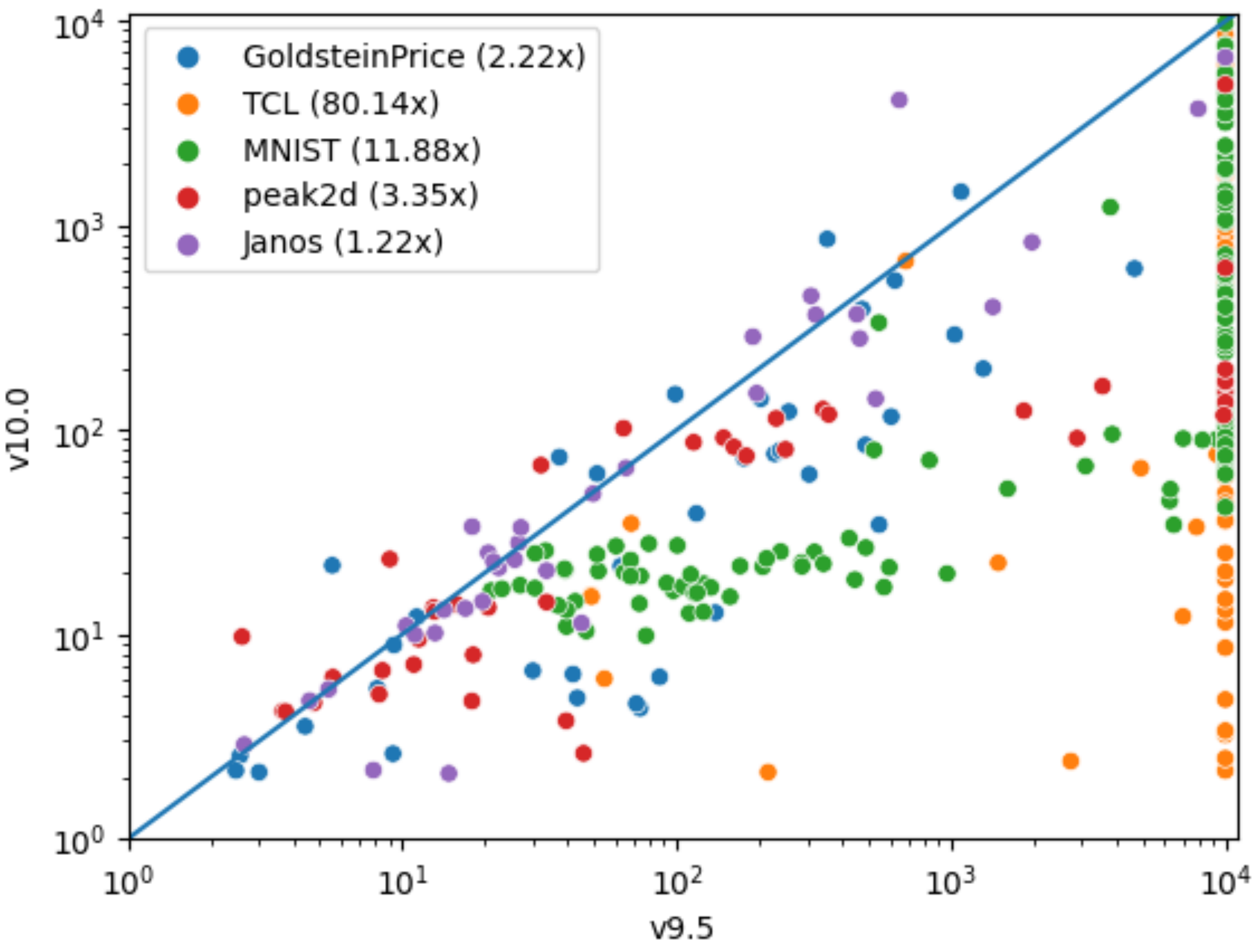
Classified as a 9

GUROBI OPTIMIZATION

Gurobi 10.0

Performance improvements for models with ML predictor constraints

Gurobi 9.5 vs Gurobi 10.0



- Models incorporating Neural Networks
- Performance gains come mostly from Optimization Based Bound Tightening (OBBT)

- 478 instances
- 10.000 seconds time limit
- Intel(R) Xeon(R) CPU E3-1240 CPUs, 4 cores, 4 threads
- Solved means 0.01% gap reached
- Models not solved by any version were excluded

Conclusion

Conclusion

- Gurobi Machine Learning
  https://github.com/Gurobi/gurobi-machinelearning
- This is experimental
- We seek your input
- Performance for models with Neural Networks in Gurobi 10

Dangers and Pitfalls

- ML models we can hope to handle is still limited (e.g. no categorical features)
- Methodological issues:
  - How to decide which prediction model to use?
  - How to make sure that optimization doesn't misuse results of the predictor?

Notebook Example: Avocado Price Optimization

www.gurobi.com/jupyter_models/avocado-price-optimization/

Jerry Yurchisin
Data Science Strategist

GUROBI
OPTIMIZATION

INFORMS GUROBI TUTORIALS:

Gurobi's Newest Educational Resources: Where Data Meets Decisions - An Overview of Our Free Jupyter Notebook Data Science Example Library
Monday, April 17, 2022 from 10:30 – 11:20AM – Cottonwood 10

Gurobi Machine Learning: Incorporate your Machine Learning Models into Optimization
Tuesday, April 18, 2022 from 9:10 – 10:00AM – Cottonwood 11

Thank You

For more on our Notebook Example Library:
www.gurobi.com/jupyter_models/

Jerry Yurchisin
Data Science Strategist

GUROBI OPTIMIZATION

QUESTIONS?

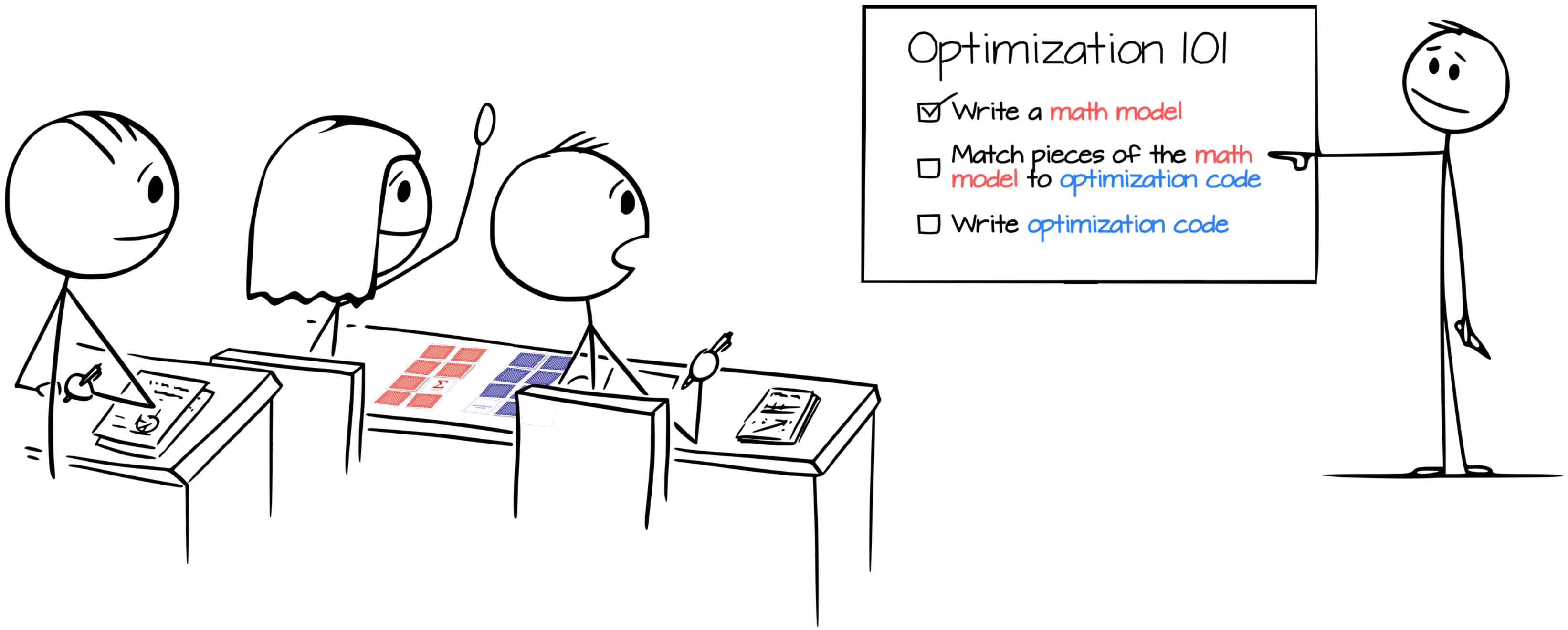
Thank You

For more information: gurobi.com

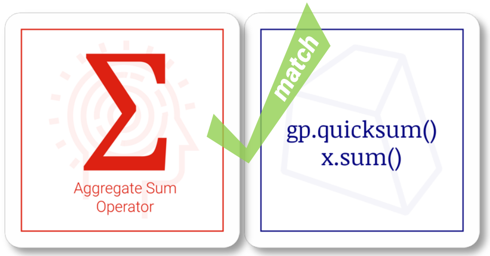
VISIT US AT BOOTH #217 TO LEARN MORE!

Gurobipy Card Game

Download the complete game at gurobi.com/cardgame



This game asks learners to learn to write code by first stepping away from the computer to match the pieces of a **math model** to **optimization code**.



GUROBI OPTIMIZATION

GIVEAWAY

Thank You

For more information: gurobi.com

VISIT US AT BOOTH #217 TO LEARN MORE!

Examples of New Capabilities

GUROBI OPTIMIZATION

```
x = m.addMVar(3); m.update()
(x - 1) @ x
✓ 0.6s

<MQuadExpr () >
array( -1.0 C21 +
C23 ^ 2 ])
```

```
# was not possible in the past
X = m.addMVar((2,3))
Y = m.addMVar((2,3))
m.addConstr(X <= Y)  # 6 linear cons

<(2, 3) matrix constraint *awaiting mode
```

```
<gurobi.LinExpr: 0.5758868904573695 C38 + 0.6852
+ 0.45842518951552946 C40 + 0.4745041184453236 C
0.34573908244061197 C42 + 0.41142171691861207 C4
0.2752198513581124 C44 + 0.28487298676878264 C45
0.3994472077972281 C46 + 0.47533317578729417 C47
0.3179733120690802 C48 + 0.32912599390956354 C49>
```

```
# was not possible in the past
X = m.addMVar(3,3)
A = np.random.rand(3,3)
B = np.random.rand(3,3)
m.addConstr(A @ X == B)  # 9 linear constraints A[i,:] @ X[:,j]

<(3, 3)
```

```
# was not possible in the
x = m.addMVar((4,1), lb=
X = m.addMVar((4,4), lb=
m.addConstr(X == x @ x.T)

<(4, 4) matrix quadratic con
```

```
m = gp.Model()
z = m.addMVar(8)
X = m.addMVar((8, 3), lb=-np.inf)
# Add eight standard cones
m.addConstr(z**2 >= (X**2).sum(axis=1))
✓ 0.5s

<(8,) matrix quadratic constraint *awaiting mo
update*>
```

Numpy Copies & Views

- Create View through indexing

```
x = np.arange(10);x
```

0.1s Python

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
y = x[1:3]  # creates a view though indexing
x[1:3] = [10, 11]
print(x,y)
print(x.base)
print (y.base) # y has a base,
```

0.9s Python

```
[ 0 10 11  3  4  5  6  7  8  9] [10 11]
None
[ 0 10 11  3  4  5  6  7  8  9]
```

```
x[1:3] = [10, 11] # every change in base will effect y
print(y,x)
```

0.1s Python

```
[10 11] [ 0 10 11  3  4  5  6  7  8  9]
```

- Create Copy through advance indexing

```
x = np.arange(9)
y = x.reshape(3, 3);y
```

0.6s Python

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

```
y.base  # .reshape() creates a view
```

0.1s Python

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

```
z = y[[2, 1]] ;z  # Advance indexing
```

0.1s Python

```
array([[6, 7, 8],
       [3, 4, 5]])
```

```
z.base is None  # advanced indexing creates a copy
```

0.8s Python

```
True
```

Copies & Views

```python
m = gp.Model()
mle = 5 * m.addMVar(4); m.update();print(mle)
leading_part_1 = mle[:2];print(leading_part_1)
leading_part_2 = mle[[0,1]];print(leading_part_2)
leading_part_1 += 99 ;print(leading_part_1);print(mle)
leading_part_2 += 1 ;print(mle)
# the later doesn't modify mle
```

✓ 0.1s Slide Type: subslide Python

```
<MLinExpr (4,)  >
array([ 5.0 C0,  5.0 C1,  5.0 C2,  5.0 C3])
<MLinExpr (2,)>
array([ 5.0 C0,  5.0 C1])
<MLinExpr (2,)>
array([ 5.0 C0,  5.0 C1])
<MLinExpr (2,)>
array([ 99.0 + 5.0 C0,  99.0 + 5.0 C1])
<MLinExpr (4,)>
array([ 99.0 + 5.0 C0,  99.0 + 5.0 C1,  5.0 C2,  5.0 C3])
<MLinExpr (4,)>
array([ 99.0 + 5.0 C0,  99.0 + 5.0 C1,  5.0 C2,  5.0 C3])
```

- In case of doubt, always copy your slice...

```python
expr = 2 * m.addMVar((2,2)) + 1; m.update()
first_col = expr[:, 0].copy()
first_col += 1  # Leaves expr untouched
print(expr)
```

✓ 0.1s Slide Type: fragment Python

```
<MLinExpr (2, 2)>
array([[ 1.0 + 2.0 C112,  1.0 + 2.0 C113],
       [ 1.0 + 2.0 C114,  1.0 + 2.0 C115]])
```

Improved matrix-friendly gurobipy

Gurobi 10.0

GUROBI OPTIMIZATION

- More modeling capabilities for users familiar with NumPy conventions
  - Enable numpy-style axis sums
  - >>> x = m.addMVar((2,4), name='x', vtype=GRB.BINARY)
  - >>> x.sum(axis=1)
  - <MLinExpr (3,)>
  - ([<gurobi.LinExpr: x[0,0] + x[0,1] + x[0,2] + x[0,3]>,
  - <gurobi.LinExpr: x[1,0] + x[1,1] + x[1,2] + x[1,3]>
  - >>> m.addConstr(x.sum(axis=1) <= 1)

$$\sum_j x_{ij} \leq 1 \;\; \forall i \in I$$

  - Enable broadcasting operations for constraint building

$$x_{ij} \leq y_j \quad \forall\, i \in I, j \in J$$

  - >>> x = m.addMVar((3,4), name='x', vtype=gp.GRB.BINARY)
  - >>> y = m.addMVar(4, name='y', vtype=gp.GRB.BINARY)
  - >>> m.addConstr(x <= y) # 12 constraints added

- Shape of operation now consistent with analogous ndarray operations
- Elementwise multiplication works across all matrix-friendly objects and ndarray
- Mvar
  - Extract a diagonal from an MVar X:
    - X.diagonal(offset).
  - Convert a list of Var objects to an MVar:
    - x = MVar.fromlist(varlist)
  - Sum along an axis of an MVar X:
    - X.sum(axis=...)
  - Elementwise squaring of an Mvar X:
    - pow(X, 2), X**2
  - MLinExpr
    - All-zero expression: MLinExpr.zeros(shape)
    - Sum along an axis of an MLinExpr mle:
      - mle.sum(axis=..
    - New class MQuadExpr • For modeling multidimensional quadratic constraints • Similar

Gurobi 10.0 – Gurobipy

Other new matrix-friendly features/methods

- General
  - Shape of operation resultants now consistent with analogous ndarray operations
  - Elementwise multiplication works across all matrix-friendly objects and ndarray
- MVar
  - Extract a diagonal from an MVar X : X.diagonal(offset).
  - Convert a list of Var objects to an MVar: x = MVar.fromlist(varlist)
  - Sum along an axis of an MVar X: X.sum(axis=…)
  - Elementwise squaring of an Mvar X: pow(X, 2), X**2
- MLinExpr
  - All-zero expression: MLinExpr.zeros(shape)
  - Sum along an axis of an MLinExpr mle: mle.sum(axis=…)
- New class MQuadExpr
  - For modeling multidimensional quadratic constraints
  - Similar features/methods as MLinExpr
- New class MQConstr
  - Multi-dimensional constraint handle returned from model.addConstr(…) for quadratic expressions
  - Similar features/methods as MConstr