Introduction to Performance Tuning

Jennifer Locke
Manager, Technical Account Management, Americas

Gurobi 11.0
Every Solution, Globally Optimized

GUROBI OPTIMIZATION

The Optimization Workflow

- The real-world problem instance is usually derived from a specific planning problem.
- A configurable model generator is used to build the model instance using data sources.
- Gurobi is used to find an optimal solution of the model instance.
- The solution is transferred back to the planning system for further analysis.
- The cycle repeats until a satisfying real-world solution has been found.

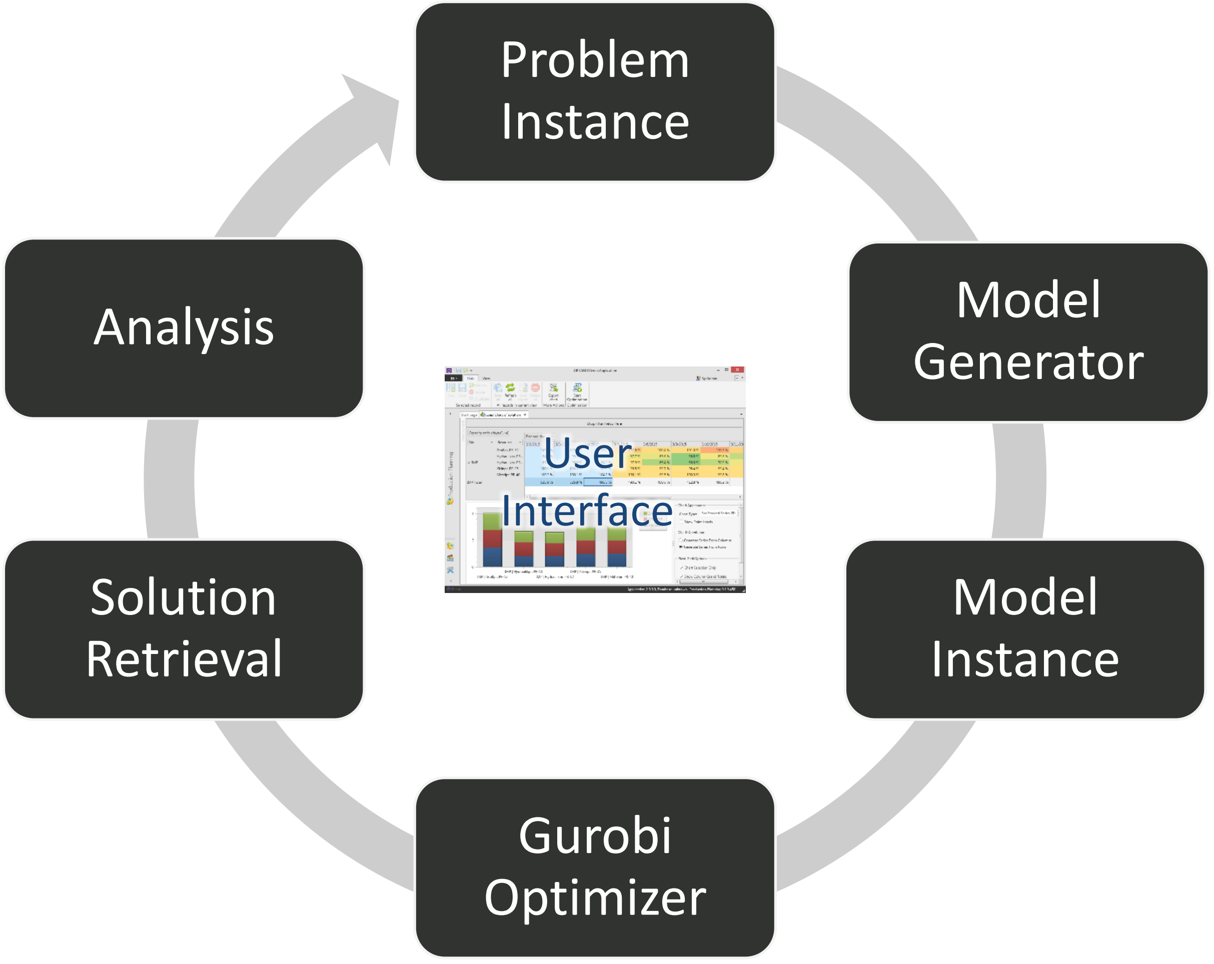


Image: http://optano.net/en/production-planning/

Agenda

01
Tuning opportunities

Performance starts with collecting the data. What other things should be considered to make an application fast?

02
Improving Gurobi performance

How can you change the solver behavior to increase Gurobi's performance?

03
Performance tuning

Best practices, a walk through some examples, and how to use Gurobi's automated tuning tool.

Tuning opportunities

Consider different areas of your application

Overview

System architecture	Hardware	Data access
Model construction	Memory	Network communication overhead
Performance variability	Numerical issues	Benchmarking and profiling

GUROBI
OPTIMIZATION

System Architecture

- Performance problems can arise in different parts of the system architecture
- Measure runtimes yourself in every part of the process
  - Processing input data
  - Model building
  - Model solving
- Keep log files
- Optimization needs physical CPU and memory resources
  –> be aware of competition

Hardware considerations

- What is the best hardware to solve an optimization problem?
    - Answer: “It depends”
- For a large MIP model, you'll get the best performance from a system with
    - The fastest possible clock rate and
    - 4 channels per socket of the fastest available memory.
- There is no hardware recommendation for all models.

You are welcome to submit a request on our portal support.gurobi.com to discuss your specific models.

GUROBI OPTIMIZATION

Data Access

- Common support request
  - “The optimization process is taking too long”
  - Reason: Model building outside of Gurobi takes 30 minutes
  - Model solving takes only a few seconds
- Inefficient data access is the most common reason for slow model construction
  - Long lookup times
  - Insufficient caching / redundant queries
  - Single elements instead of batch processing
- If you would remove all Gurobi API calls, how long would it take?

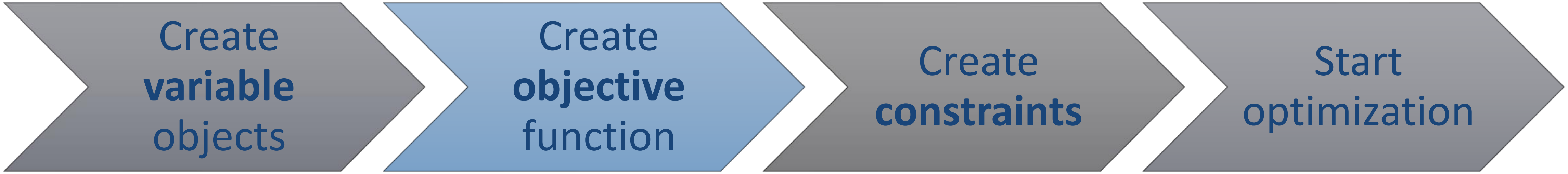
Is Gurobi the bottleneck?

- Run your program with Gurobi parameter Record=1
  - Produces a **recording000.grbr** file
- Then replay the recording file:
  - **`gurobi_cl recording000.grbr`**
- Produces runtime summary at the end…

```
…
*Replay* Gurobi API routine runtime: 2.12s
*Replay* Gurobi solve routine runtime: 23.92s
```


Model Construction

- Are you using the most efficient way to build expressions?
- Look for: bottleneck via a code profiler or measure yourself
- Object-oriented interfaces are thin layer upon C matrix interface
- Building a set of variables and constraints
  - New variables/constraints put in a lazy update queue
  - Queue flushed when you optimize()
  - Use Gurobi model objects for later queries; avoid looking up elements by their names



`addVar()`
`addVars()`

`setObjective()`

`addConstr()`
addConstrs()
`addQConstr()`

`optimize()`

Memory

- Insufficient memory can destroy performance
  - Virtual memory via disk is far slower than RAM
  - Parallel optimization requires more memory
- Look for: memory use via system monitor tools on computer
- Helpful parameters
  - Decrease number of threads
  - Set NodefileStart to store MIP node info on disk
    - Only helpful when solving a MIP that requires many nodes!
  - SoftMemLimit
- Memory is cheap

Network communication overhead

- When using Gurobi Cloud or Compute Server:
  - Be aware of potential network issues when retrieving data
    - Latency, bandwidth and stability
- Latency can become an issue when a lot of messages are sent over the network
  - Typically not an issue, due to lazy update approach
- Compute Server statistics in the log file:

```
Compute Server communication statistics:
Sent: 8.3 MBytes in 244 msgs and 0.76s (10.92 MB/s)
Received: 7.2 MBytes in 304603 msgs and 1.53s (4.71 MB/s)
```

- Use the RS command : grbcluster node latency

```
ADDRESS    LATENCY     NBERR
server1    1.12813ms   0
server2    1.218103ms  0
```


Performance variability

- MIP solvers often need to “flip a coin” to decide what to do next
  - Gurobi uses a pseudo-random number generator with a seed
  - Playing with the seed parameter helps to determine the impact of random algorithmic choices
- Performance variability is intrinsic to MIP
  - Some models behave pretty stable
  - There are also highly pathological models which solve in a fraction of a second for some seeds and cannot be solved within days for other seed values.
- Bottom line: Always pay attention to performance variability when benchmarking models
  - Use different instances of the same model type
  - Run the same model instance with different random seeds

Performance variability

Changes	Runtime
Cuts=2	4.88
Default	5.84

GUROBI
OPTIMIZATION

Numerical Issues

- Numerical problems within models can affect
  - Solution time
  - Solution quality
- It is often very helpful to reformulate and/or rescale the model.
- We don't cover this topic in this presentation.
- We provide a numerics guide and a webinar about this topic.

GUROBI OPTIMIZATION

Profiling & benchmarking

- Model initialization and solution retrieval
  - Export MPS file from API and run it with gurobi_cl
  - See if solution times are much faster
  - Or use Record feature
- What algorithmic part is the bottleneck?
  - Presolve
  - Root relaxation
  - Root node of MIP
  - Other nodes of MIP
  - Log shows time spent in presolve, LP relaxation, MIP root, nodes
- Use the logs to identify the bottleneck

Improving Gurobi performance

How to modify the solver behavior through parameter settings

A look at performance tuning from our Experts team

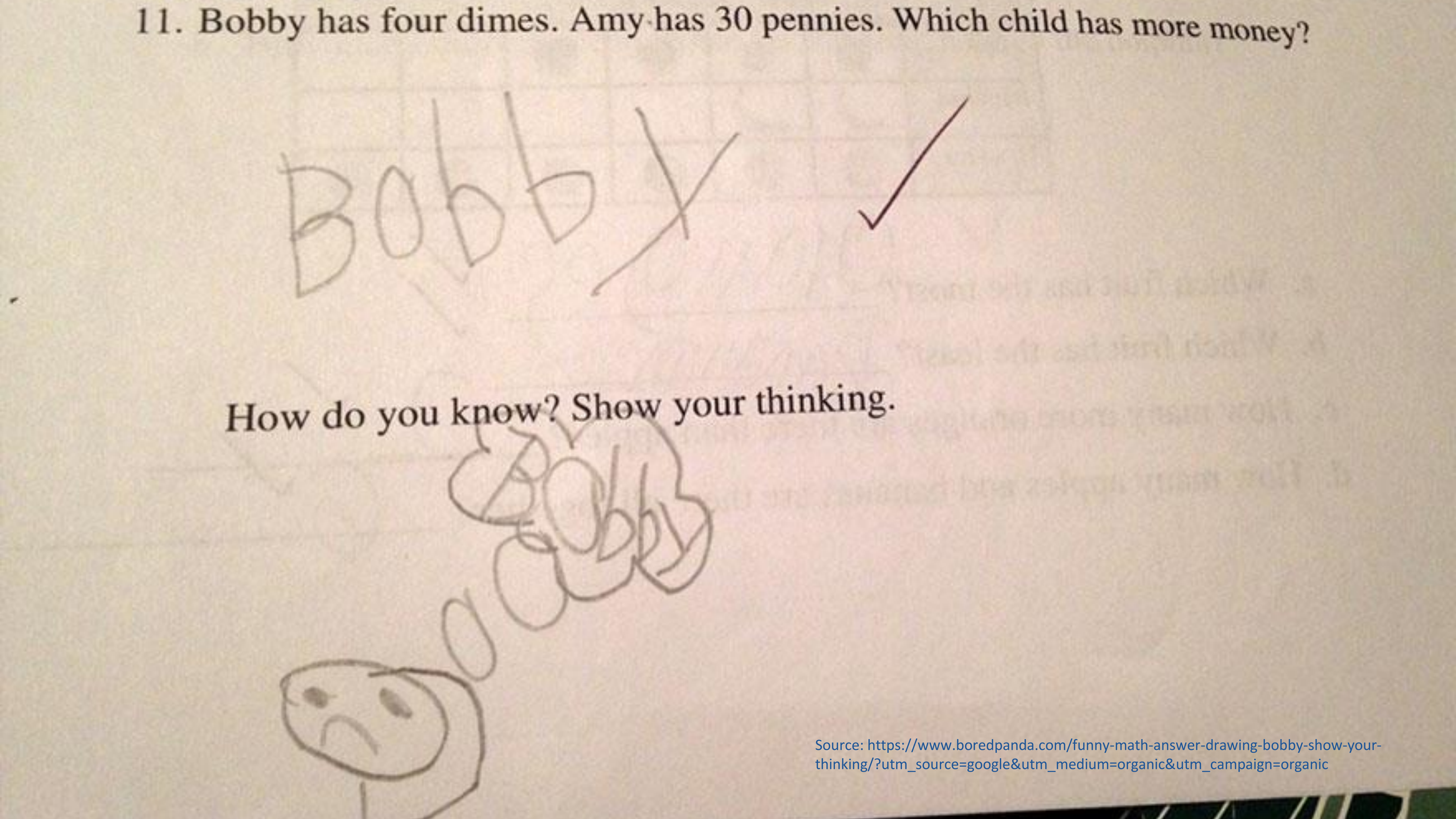
My model is slow!

Let's have a look at it!

You should set MIPFocus=1, this makes your model 50% faster

You are awesome, thanks!

But how did you decide on that particular parameter setting?



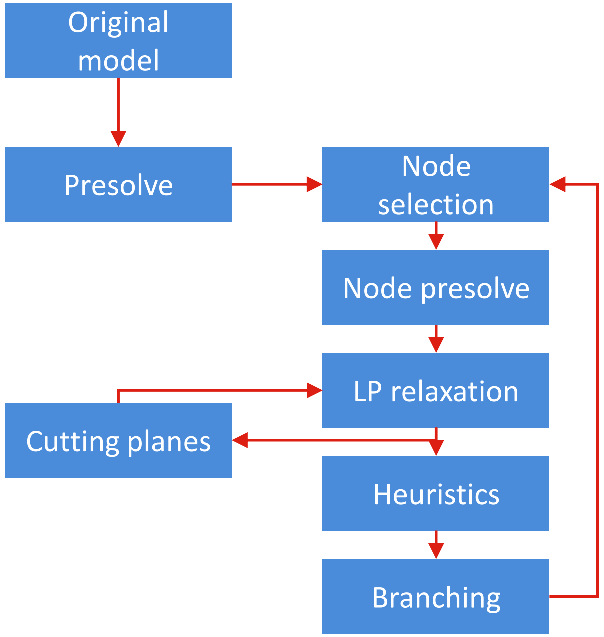
Source: https://www.boredpanda.com/funny-math-answer-drawing-bobby-show-your-thinking/?utm_source=google&utm_medium=organic&utm_campaign=organic

How Gurobi works

Understand how Gurobi proves optimality

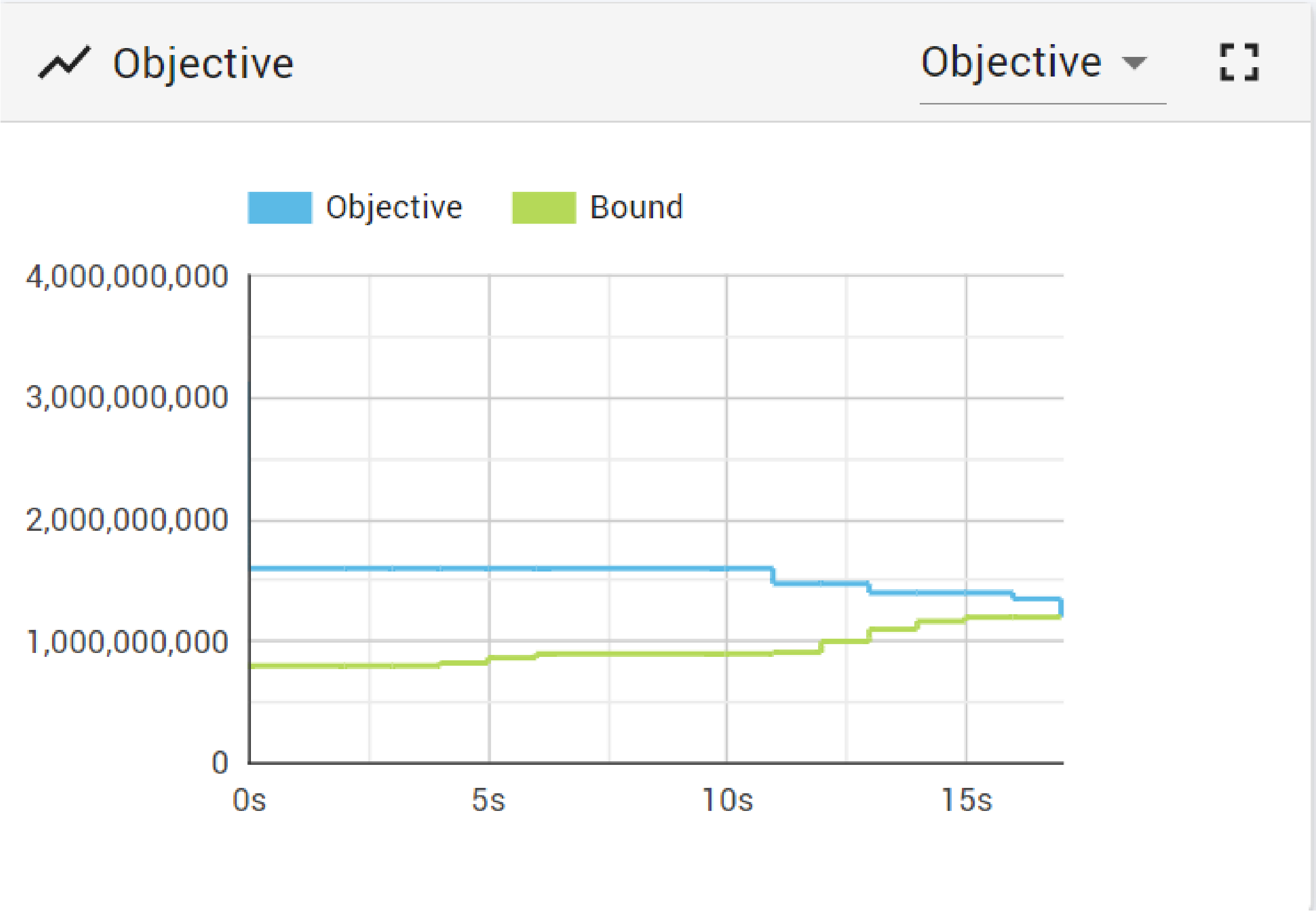
Solver process

- Gurobi consists of many algorithms that all work together
- Parameters are available to influence algorithms
- Key question for each block:
  - Do we spend much time on it?
  - Is it worth the time?
  - How can we change it?

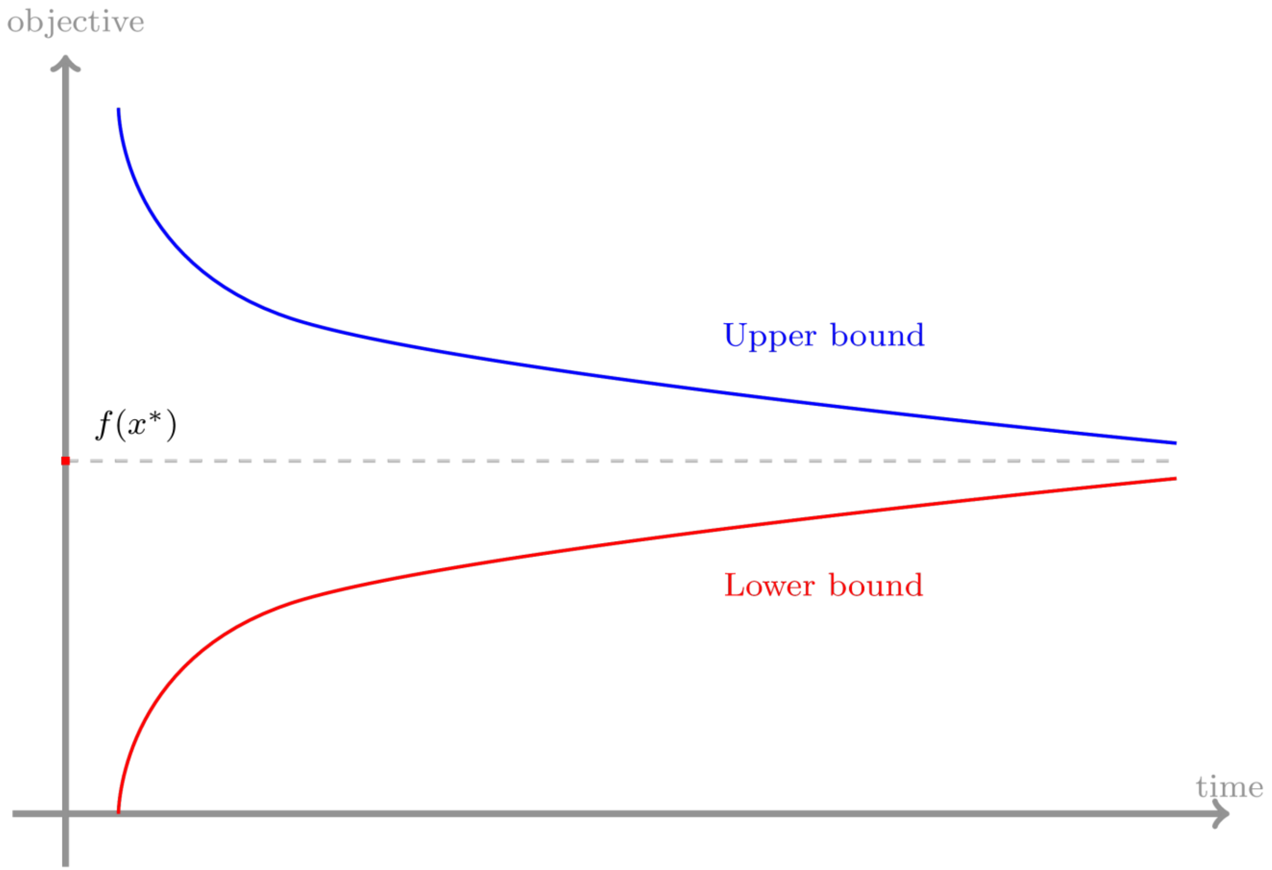


Group	# Parameters	Example
Presolve	15	Presolve
Simplex	8	SimplexPricing
Barrier	6	Crossover
MIP	34	MIPFocus
MIP Cuts	25	CutPasses
Other	27	Threads
Termination	11	TimeLimit

Visualize the path to optimality

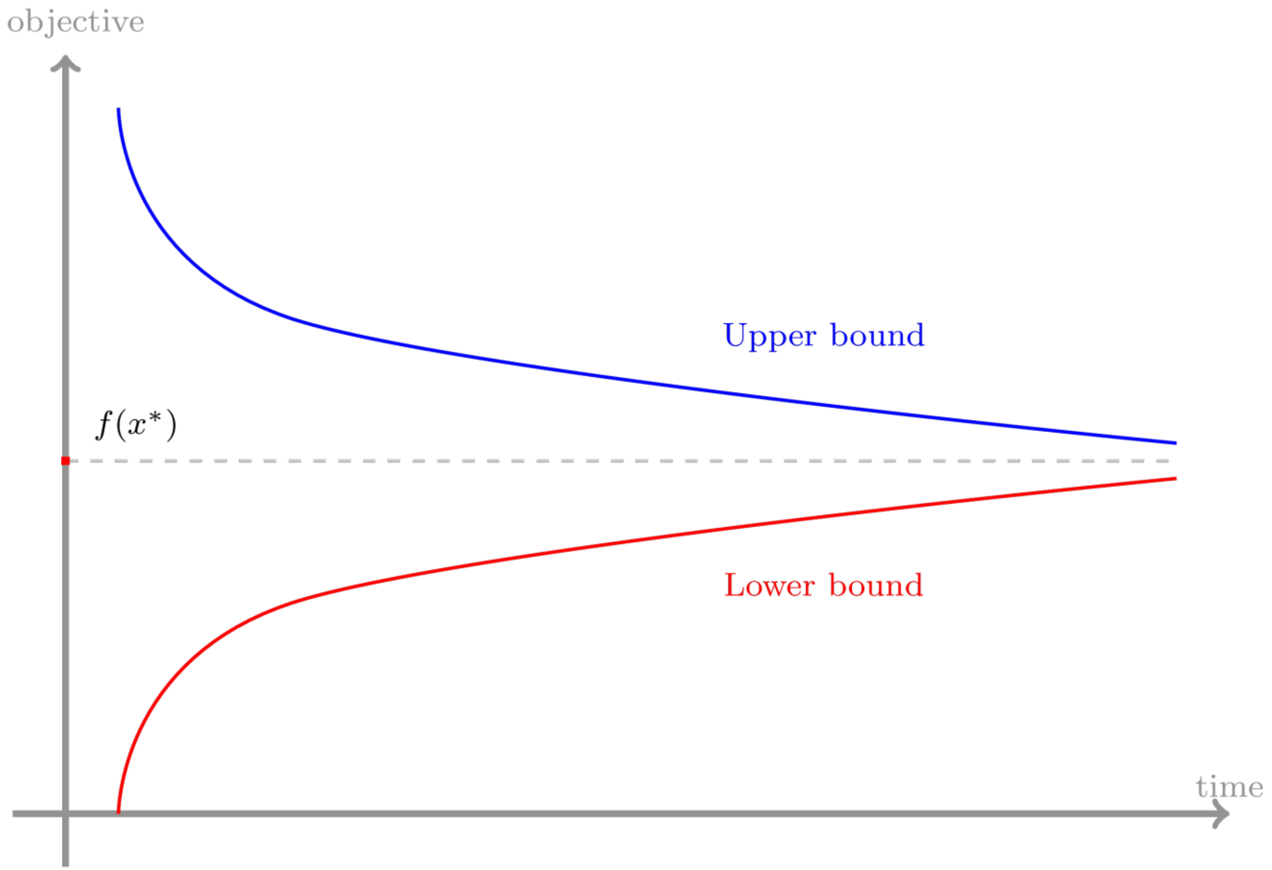


Our way to the optimal solution



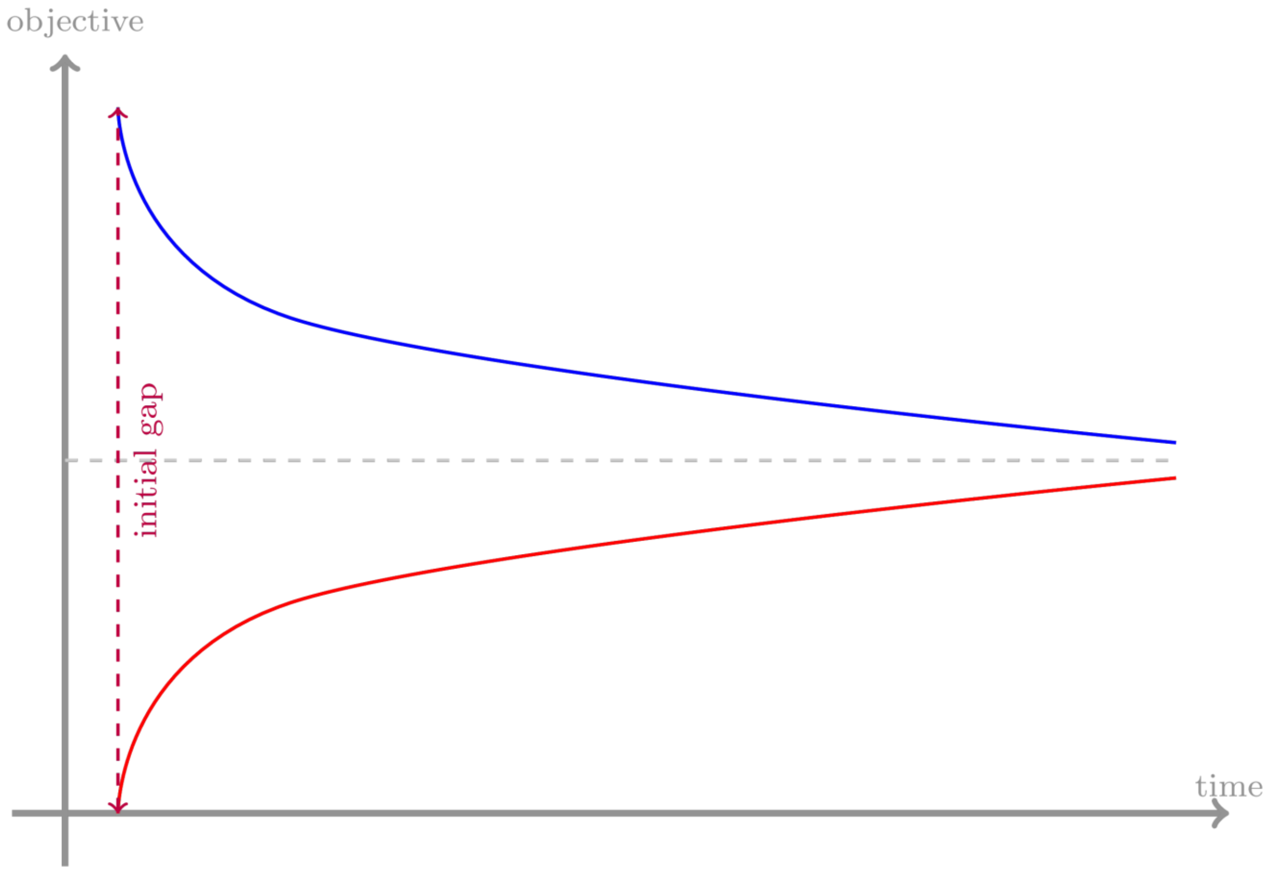
Our way to the optimal solution

How do we find f(x*)?



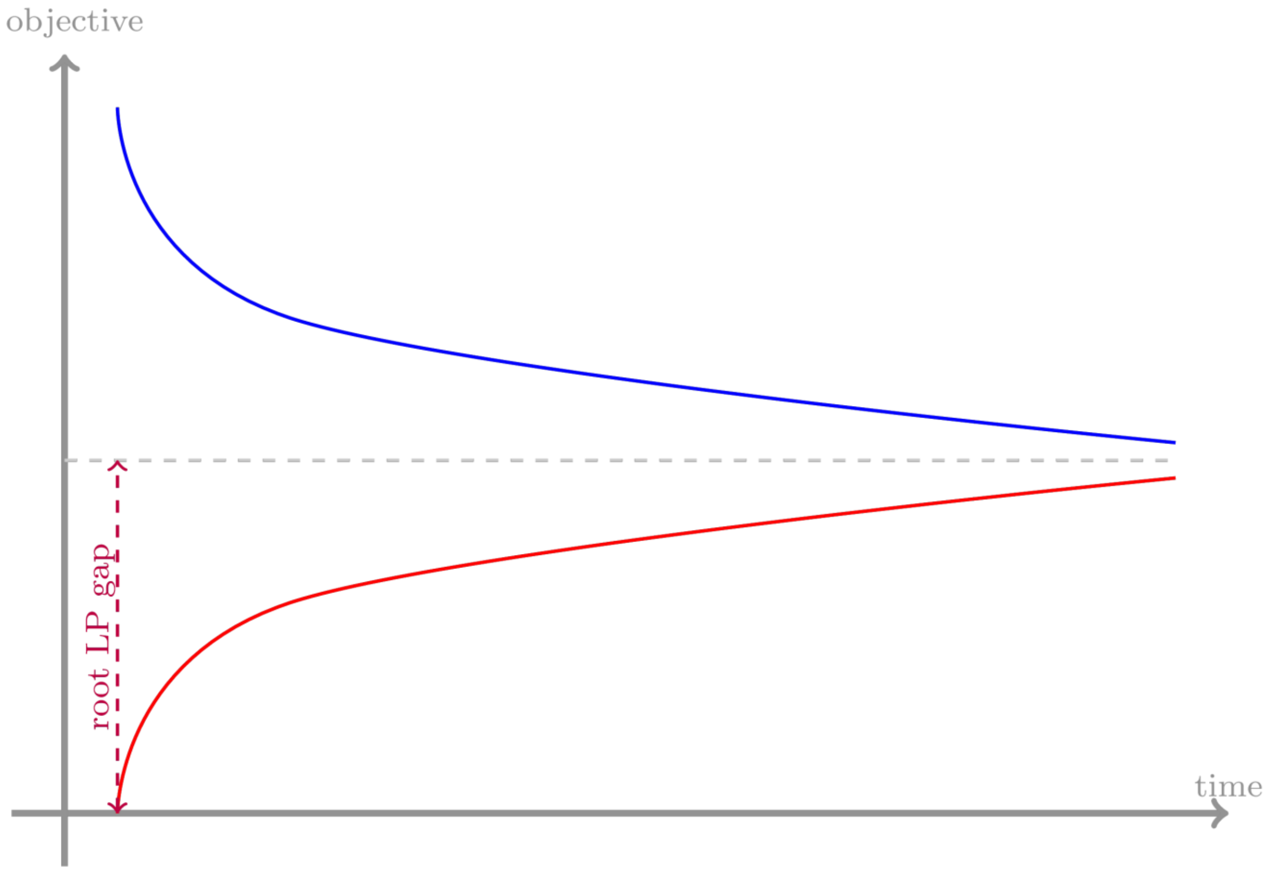
Our way to the optimal solution

How do we find f(x*)?



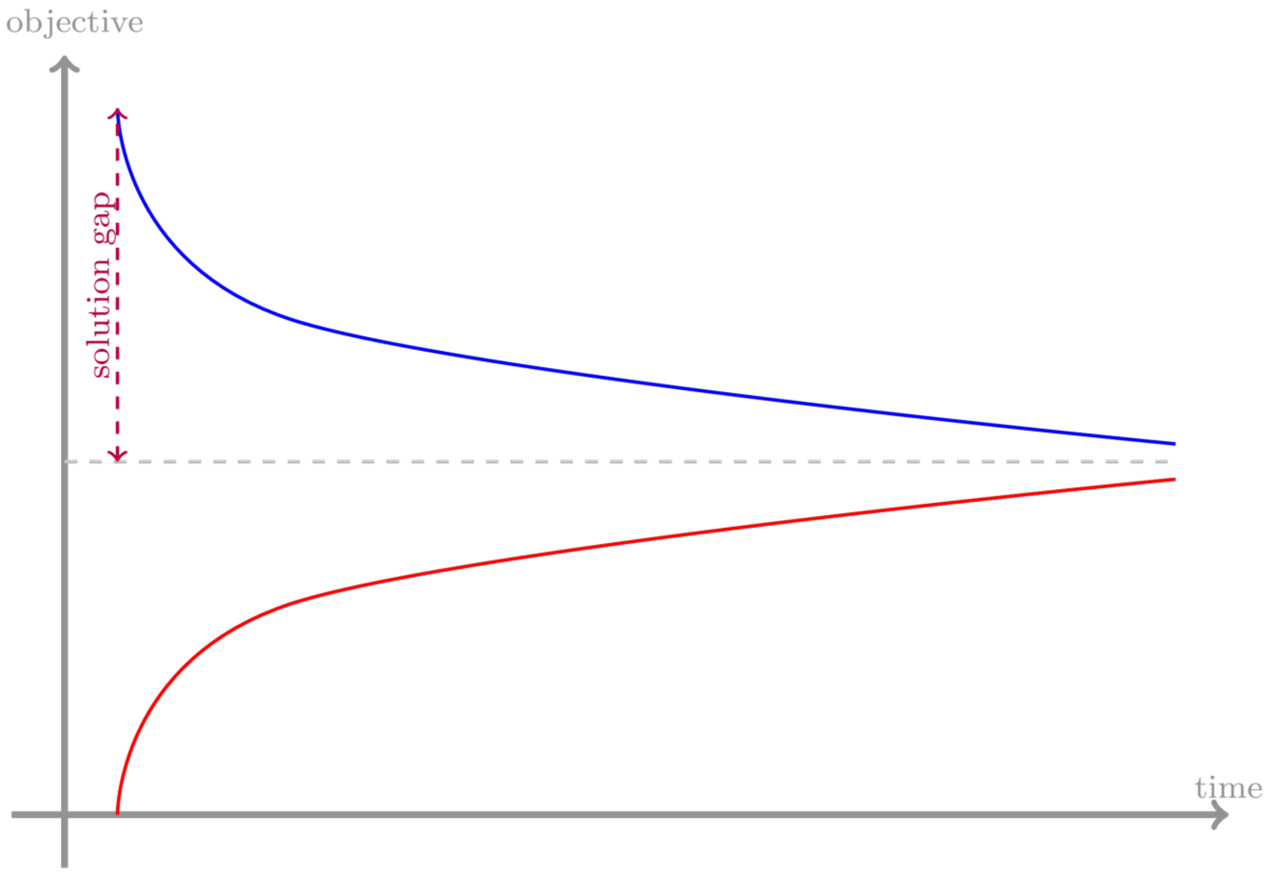
Our way to the optimal solution

How do we find f(x*)?



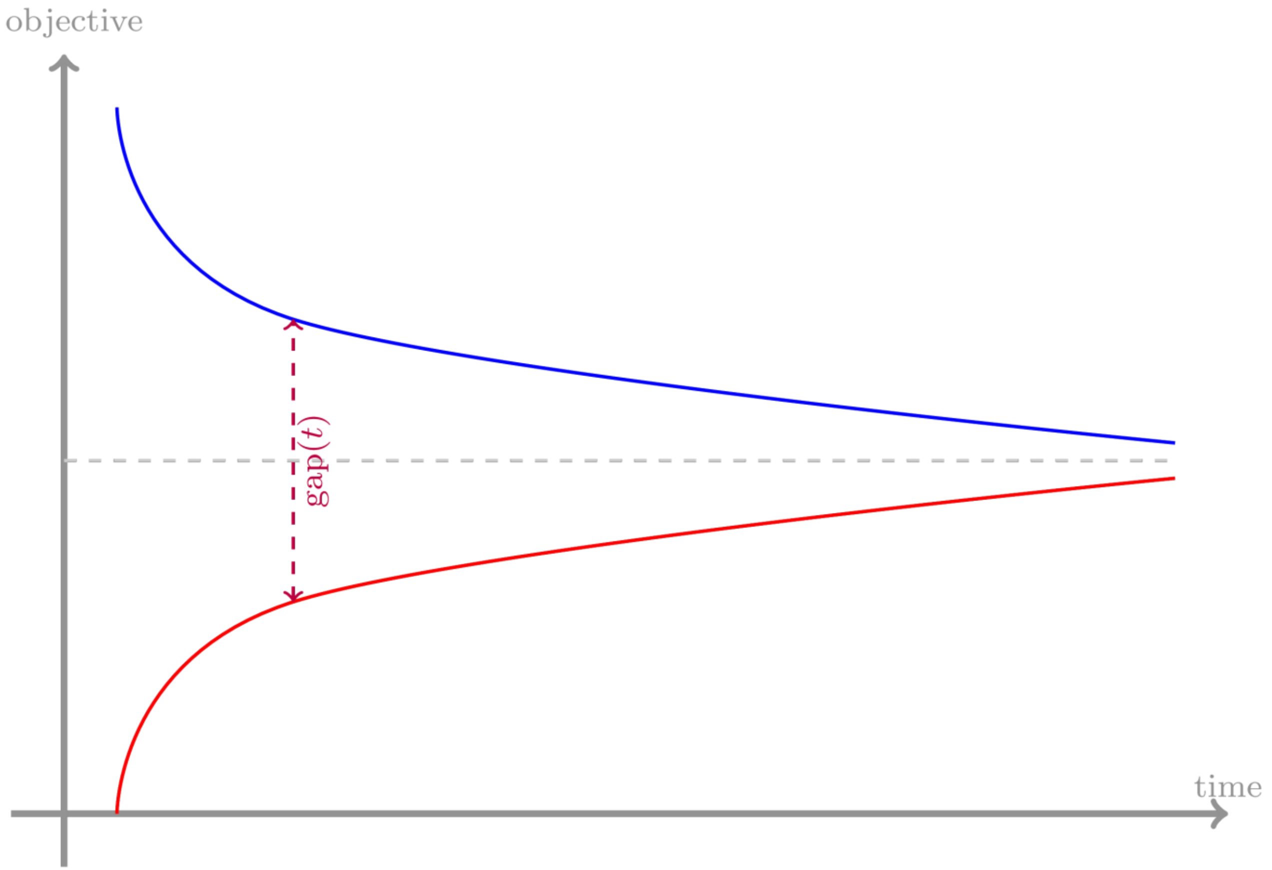
Our way to the optimal solution

How do we find f(x*)?



Our way to the optimal solution

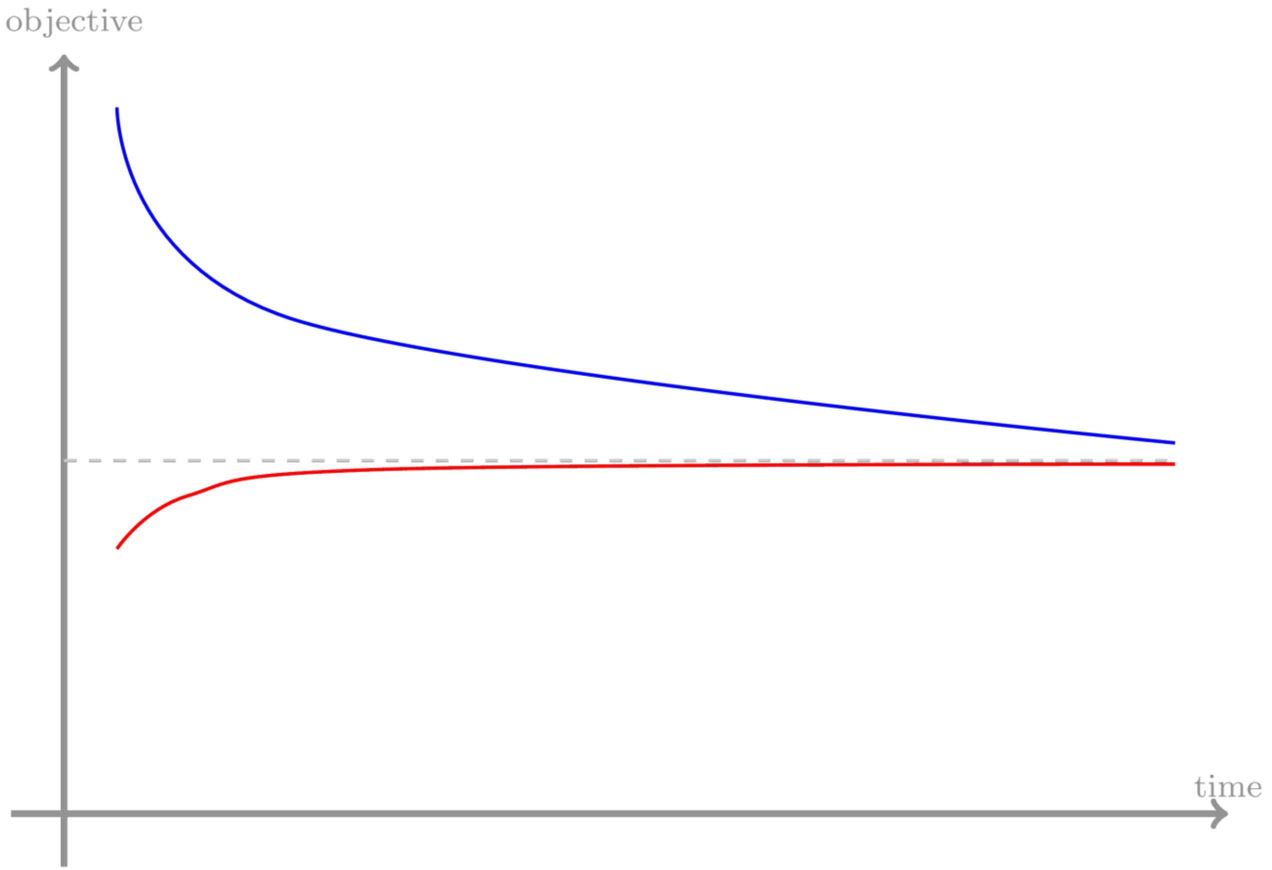
How do we find f(x*)?



Our way to the optimal solution

CASE 1:

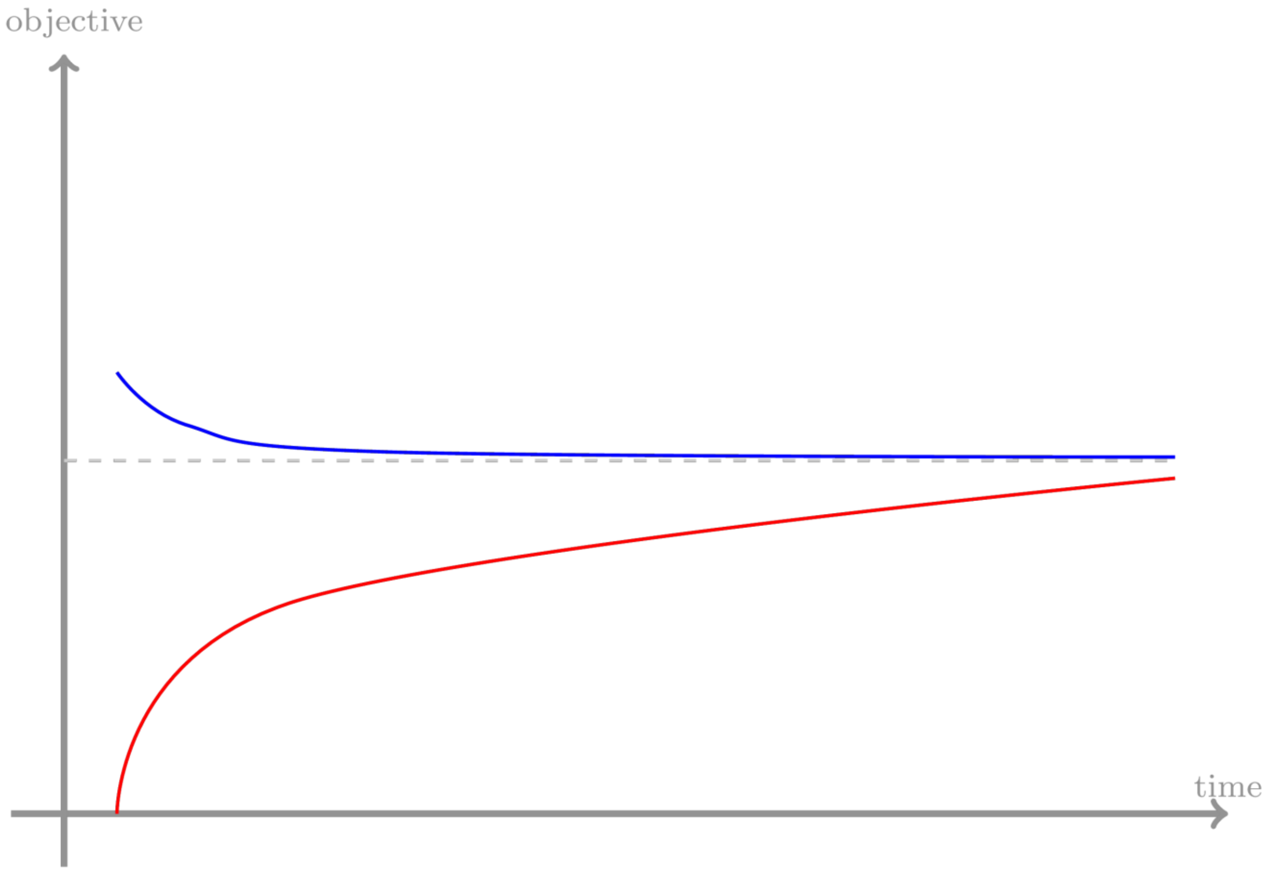
- Lower bound seems OK
- Cuts + initial B&B do most of the job
- Feasible solutions seem to be a struggle
- Change solver parameters
- Generate your own solution
- Consideration reformulating the model



Our way to the optimal solution

CASE 2:

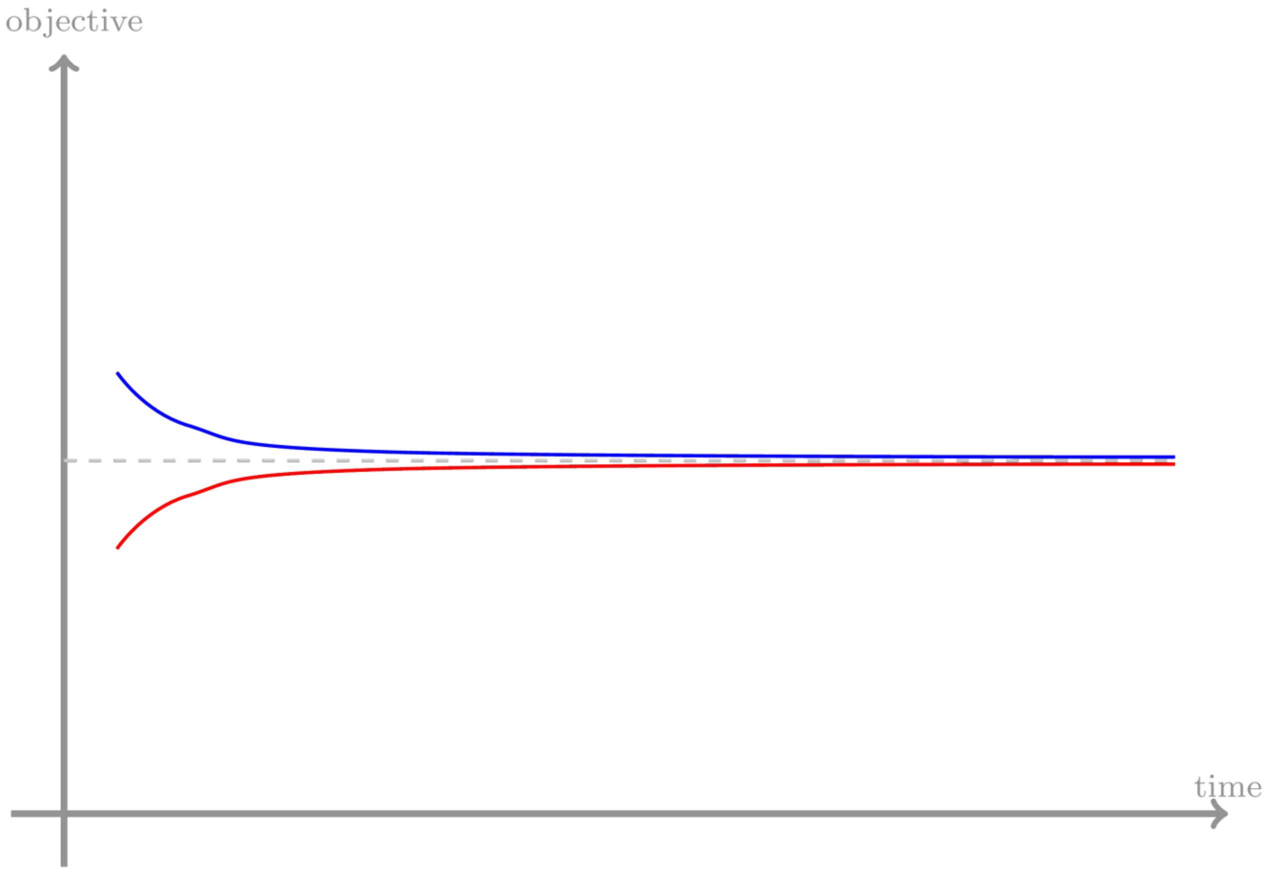
- Upper bound seems OK
- Internal heuristics + user input do the job early on
- Lower bound is horrible!
  - Cuts
  - More pre-processing
  - Reformulate



Our way to the optimal solution

CASE 3:

- Quick bounds and solutions
- Quick depends on the scale!
- Tail-off effect is fairly common
- Termination criteria is key
  - How precise is my data
  - How precise is my model
  - Better than current approach is usually enough



Improving Gurobi performance

How to modify the solver behavior through parameter settings

Parameters

Version 10.0 has 215 parameters

89 parameters related to performance

Termination and Tolerances

Simplex and Barrier

MIP and Cuts

Presolve and Multiple Solutions

Distributed algorithms and Tuning

Other
Cloud, Compute Server, Cluster Manager, WLS, and Token Server

Presolve

Tradeoff

- Spend time up front with hope of simplifying model
- Primary control: Presolve parameter
  - Reduce if spending too much time up front
  - Increase to hope to get a simpler model

Additional parameters for fine-grain control

- PrePasses
- Aggregate
- AggFill
- PreSparsify
- PreDual
- PreDepRow

4 parameters to control SOS formulations

Continuous algorithms

- Primal Simplex
- Dual Simplex
- Barrier
- Concurrent (deterministic or non-deterministic)

Defaults for continuous optimization

LP

Concurrent (non-deterministic)

QP

Barrier

MIP root node

Dual Simplex or (deterministic) concurrent depending on model side

MIP nodes

Dual Simplex

Notable continuous parameters

NormAdjust

Select different simplex pricing norm variants

SimplexPricing

Simplex variable pricing strategy

Method

Selects algorithm used to solve the initial LP relaxation of a MIP or the LP itself.

Crossover

Determines strategy used to produce basis from initial crossover basis

Integer model – What makes it difficult?

01
Time to solve LP/QP relaxation?

Adjust LP parameters

02
Is the bound moving?

Adjust cutting plane generation

Improve node throughput

03
Are feasible solutions found?

Increase heuristics

Change branching strategy

Additional helpful MIP Parameters

GUROBI OPTIMIZATION

VarBranch

Change branching strategy using

Cuts

Fine granular control over 21 different cut types or control all at once using Cuts and CutPasses parameter

Heuristics

Limits the overall amount of time spent in heuristics

Find a first feasible solution

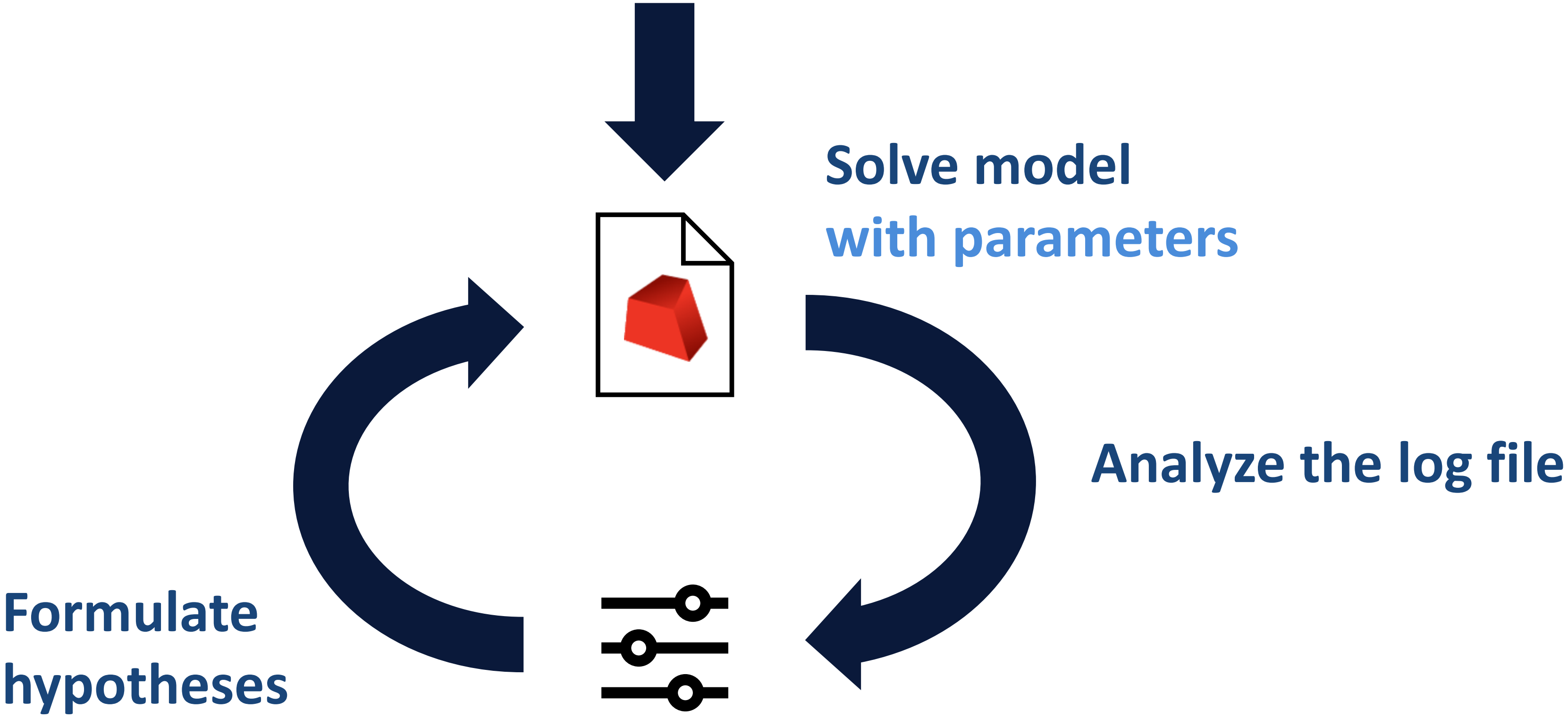
- NoRelHeurTime
- PumpPasses
- MinRelNodes
- ZeroObjNodes

Performance tuning

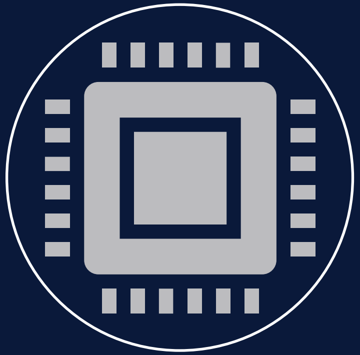
Consider different areas of your application

The art of parameter tuning

Parameters change the runtime of the problem. This means we need runtime information to judge what to choose!



The craft of parameter tuning

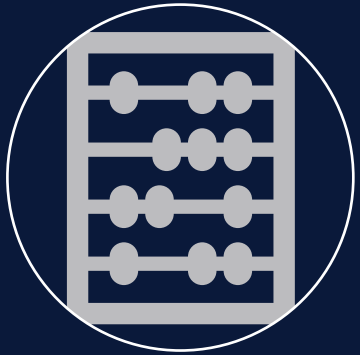


Hardware needs to stay the same

Multiple random seeds

Validate result on multiple models

Don’t overtune



Preferably use parameters that do not “count” something (e.g. CutPasses)

Parameter pitfalls

- Don’t over-tune parameters
  - Default values are carefully selected, based on thousands of models
  - Avoid setting parameters unless they produce a big improvement across multiple test models
  - "Less is more"
- Don’t assume parameters that were effective for another solver are ideal for Gurobi
- If there is a new Gurobi major release, try the defaults first

Manual tuning

Example 1

- A model takes 7-8 minutes to solve to a MIP gap below 0.1%, with some runs taking over 15 minutes.
- Customer would like to reduce this as much as possible, ideally to 1-2 minutes.

	Nodes		Current Node			Objective Bounds			Work	
	Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
	0	0	-2.130e+07	0	385	-	-2.130e+07	-	-	18s
	0	0	-2.130e+07	0	563	-	-2.130e+07	-	-	21s
	0	0	-2.130e+07	0	552	-	-2.130e+07	-	-	22s
	0	0	-2.130e+07	0	545	-	-2.130e+07	-	-	22s
	0	0	-2.130e+07	0	646	-	-2.130e+07	-	-	25s
	0	0	-2.130e+07	0	676	-	-2.130e+07	-	-	26s
	0	0	-2.130e+07	0	568	-	-2.130e+07	-	-	28s
	0	0	-2.130e+07	0	637	-	-2.130e+07	-	-	28s
	0	0	-2.130e+07	0	637	-	-2.130e+07	-	-	28s
	0	0	-2.130e+07	0	623	-	-2.130e+07	-	-	30s
	0	0	-2.130e+07	0	645	-	-2.130e+07	-	-	31s
	0	0	-2.130e+07	0	637	-	-2.130e+07	-	-	31s
	0	0	-2.130e+07	0	622	-	-2.130e+07	-	-	31s
	0	0	-2.130e+07	0	624	-	-2.130e+07	-	-	32s
	0	0	-2.130e+07	0	624	-	-2.130e+07	-	-	32s
	0	0	-2.130e+07	0	603	-	-2.130e+07	-	-	33s
	0	0	-2.130e+07	0	616	-	-2.130e+07	-	-	34s
	0	0	-2.130e+07	0	612	-	-2.130e+07	-	-	34s
	0	0	-2.130e+07	0	620	-	-2.130e+07	-	-	35s
	0	0	-2.130e+07	0	628	-	-2.130e+07	-	-	35s
	0	0	-2.130e+07	0	628	-	-2.130e+07	-	-	36s
	0	0	-2.130e+07	0	616	-	-2.130e+07	-	-	37s
	0	0	-2.130e+07	0	606	-	-2.130e+07	-	-	38s
	0	2	-2.130e+07	0	605	-	-2.130e+07	-	-	46s
	27	32	-2.130e+07	6	632	-	-2.130e+07	-	102	50s
	64	103	-2.130e+07	13	613	-	-2.130e+07	-	73.3	57s
	102	156	-2.130e+07	21	606	-	-2.130e+07	-	63.7	66s
	155	284	-2.130e+07	29	604	-	-2.130e+07	-	79.3	83s
	283	461	-2.131e+07	51	577	-	-2.130e+07	-	91.0	110s
	460	636	-2.131e+07	83	505	-	-2.130e+07	-	94.0	139s
	635	973	-2.131e+07	109	525	-	-2.130e+07	-	100	169s
	972	1488	-2.131e+07	165	420	-	-2.130e+07	-	98.2	196s
	1487	1990	-2.131e+07	261	333	-	-2.130e+07	-	99.3	219s
	1989	2548	-2.131e+07	372	226	-	-2.130e+07	-	100	241s
	2548	3135	-2.131e+07	493	251	-	-2.130e+07	-	95.4	262s
	3135	3791	-2.132e+07	615	199	-	-2.130e+07	-	88.4	281s
	3792	4404	-2.132e+07	779	226	-	-2.130e+07	-	78.1	300s
	4408	5082	-2.132e+07	942	112	-	-2.130e+07	-	72.0	316s
H	4896	5078				-2.13195e+07	-2.130e+07	0.07%	67.2	316s

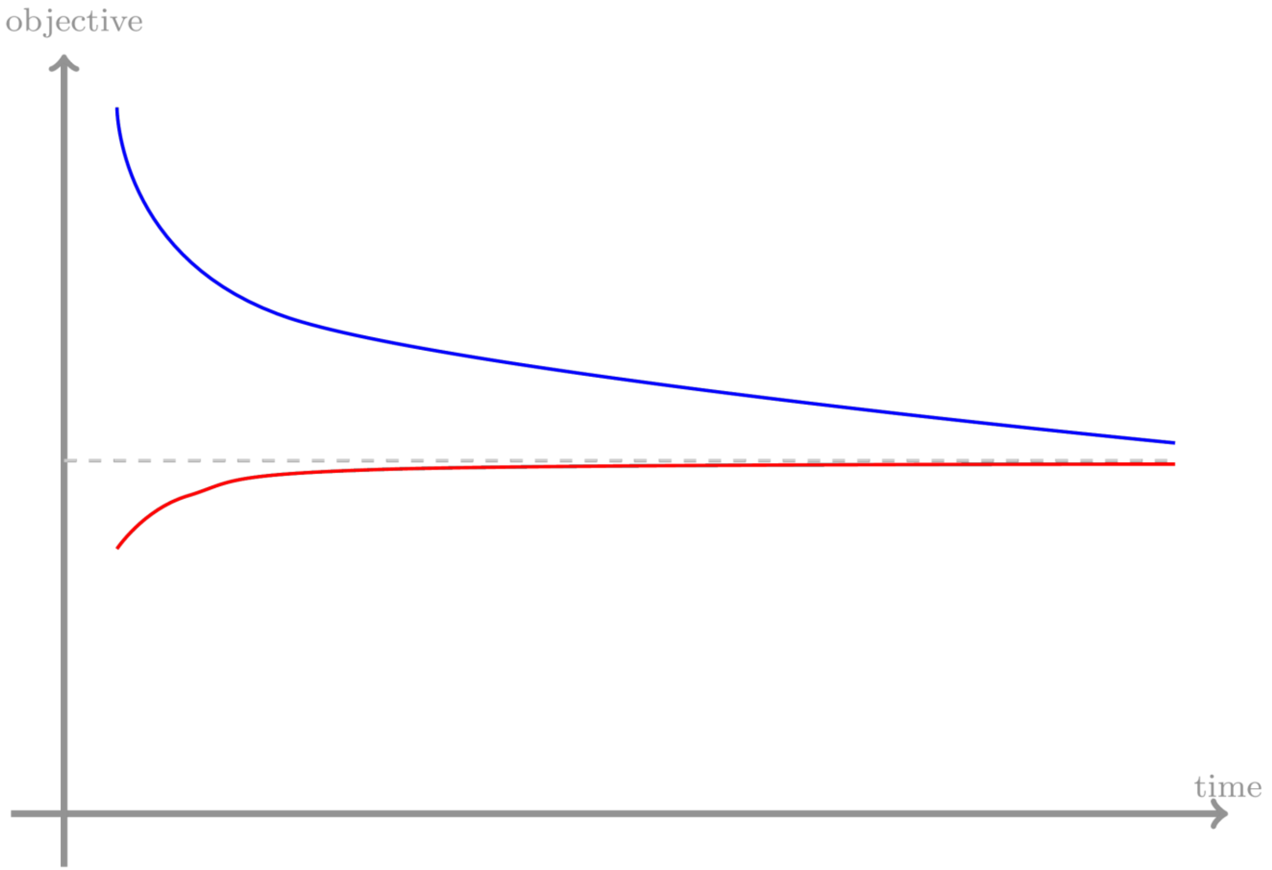
Example 1

Case 1:

- Lower bound seems OK
- Cuts + initial B&B do most of the job
- **Feasible solutions seem to be a struggle**
- Change solver parameters
- Generate your own solution
- Consideration reformulating the model

Try:

- Increase heuristics
- Change branching strategy



MIPFocus parameter

the Gurobi MIP solver strikes a balance between finding new feasible solutions and proving that the current solution is optimal.

- MIPFocus=0 Default
- MIPFocus=1 focus on incumbent, finding feasible solutions quickly
- MIPFocus=2 focus more attention on proving optimality
- MIPFocus=3 to focus on the Best bound

Example 1

- After using MIPfocus=1
- Solves in four minutes

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 -2.130e+07    0  385          - -2.130e+07      -     -   21s
     0     0 -2.130e+07    0  541          - -2.130e+07      -     -   24s
     0     0 -2.130e+07    0  530          - -2.130e+07      -     -   26s
     0     0 -2.130e+07    0  531          - -2.130e+07      -     -   26s
     0     0 -2.130e+07    0  608          - -2.130e+07      -     -   28s
     0     0 -2.130e+07    0  631          - -2.130e+07      -     -   29s
     0     0 -2.130e+07    0  612          - -2.130e+07      -     -   30s
     0     0 -2.130e+07    0  620          - -2.130e+07      -     -   33s
     0     0 -2.130e+07    0  631          - -2.130e+07      -     -   34s
     0     0 -2.130e+07    0  672          - -2.130e+07      -     -   34s
     0     0 -2.130e+07    0  614          - -2.130e+07      -     -   36s
     0     0 -2.130e+07    0  617          - -2.130e+07      -     -   37s
     0     0 -2.130e+07    0  639          - -2.130e+07      -     -   38s
     0     0 -2.130e+07    0  617          - -2.130e+07      -     -   39s
H    0     0                    -3.40760e+07 -2.130e+07  37.5%     -   92s
H    0     0                    -2.81400e+07 -2.130e+07  24.3%     -  139s
H    0     0                    -2.81392e+07 -2.130e+07  24.3%     -  139s
     0     2 -2.130e+07    0  614 -2.814e+07 -2.130e+07  24.3%     -  140s
H   27    32                    -2.80958e+07 -2.130e+07  24.2%   106  142s
H   30    32                    -2.80955e+07 -2.130e+07  24.2%  96.0  142s
    50    76 -2.130e+07   11  592 -2.810e+07 -2.130e+07  24.2%  73.3  145s
H  116   139                    -2.62917e+07 -2.130e+07  19.0%   113  155s
H  119   139                    -2.61379e+07 -2.130e+07  18.5%   112  155s
   138   350 -2.130e+07   24  551 -2.614e+07 -2.130e+07  18.5%   103  164s
H  162   350                    -2.61315e+07 -2.130e+07  18.5%   102  164s
H  213   350                    -2.61254e+07 -2.130e+07  18.5%   105  164s
H  320   350                    -2.61246e+07 -2.130e+07  18.5%   107  164s
   349   558 -2.131e+07   61  512 -2.612e+07 -2.130e+07  18.5%   108  223s
H  400   558                    -2.58595e+07 -2.130e+07  17.6%   104  223s
H  452   558                    -2.13529e+07 -2.130e+07  0.23%   105  223s
H  504   558                    -2.13347e+07 -2.130e+07  0.14%   107  223s
H  556   558                    -2.13315e+07 -2.130e+07  0.13%   105  223s
H  557   601                    -2.13152e+07 -2.130e+07  0.05%   105  245s
```

Example 1

- After using `heuristics=0`
- Talks longer to find the first feasible solution than with MIPFocus=1
- Solves in two minutes

	Nodes		Current Node			Objective Bounds			Work	
	Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
	0	0	-2.130e+07	0	385	-	-2.130e+07	-	-	17s
	0	0	-2.130e+07	0	541	-	-2.130e+07	-	-	18s
	0	0	-2.130e+07	0	530	-	-2.130e+07	-	-	18s
	0	0	-2.130e+07	0	531	-	-2.130e+07	-	-	19s
	0	0	-2.130e+07	0	608	-	-2.130e+07	-	-	21s
	0	0	-2.130e+07	0	631	-	-2.130e+07	-	-	21s
	0	0	-2.130e+07	0	612	-	-2.130e+07	-	-	21s
	0	0	-2.130e+07	0	618	-	-2.130e+07	-	-	24s
	0	0	-2.130e+07	0	624	-	-2.130e+07	-	-	24s
	0	0	-2.130e+07	0	655	-	-2.130e+07	-	-	24s
	0	0	-2.130e+07	0	617	-	-2.130e+07	-	-	26s
	0	0	-2.130e+07	0	599	-	-2.130e+07	-	-	26s
	0	0	-2.130e+07	0	631	-	-2.130e+07	-	-	26s
	0	0	-2.130e+07	0	601	-	-2.130e+07	-	-	28s
	0	2	-2.130e+07	0	600	-	-2.130e+07	-	-	29s
	3	8	-2.130e+07	2	625	-	-2.130e+07	-	223	30s
	129	226	-2.131e+07	23	551	-	-2.130e+07	-	75.7	37s
	225	447	-2.131e+07	38	549	-	-2.130e+07	-	78.0	44s
	446	661	-2.131e+07	79	505	-	-2.130e+07	-	82.3	51s
	660	929	-2.131e+07	133	362	-	-2.130e+07	-	84.1	58s
	928	1322	-2.131e+07	187	333	-	-2.130e+07	-	83.1	66s
	1321	1655	-2.132e+07	268	298	-	-2.130e+07	-	74.3	74s
	1654	2124	-2.132e+07	332	276	-	-2.130e+07	-	72.6	81s
	2123	2661	-2.132e+07	437	208	-	-2.130e+07	-	64.1	89s
	2660	3086	-2.132e+07	610	185	-	-2.130e+07	-	55.0	95s
	3085	3448	-2.132e+07	700	73	-	-2.130e+07	-	51.1	102s
	3447	3801	-2.133e+07	854	61	-	-2.130e+07	-	50.0	109s
	3800	4097	-2.133e+07	902	22	-	-2.130e+07	-	49.8	116s
*	3834	4088		930		-2.13251e+07	-2.130e+07	0.10%	49.5	116s
*	4060	3507		820		-2.13205e+07	-2.130e+07	0.08%	49.5	116s
*	4061	3506		821		-2.13205e+07	-2.130e+07	0.08%	49.5	116s

Example 2

- In this exercise we will play with a model from the MIPLIB collection called [gauja](https://miplib.zib.de/instance_details_neos-3530903-gauja.html).

```
Root relaxation: objective 1.634801e+02, 582 iterations, 0.01 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  163.48010    0  183  205.00000  163.48010  20.3%     -    0s
H    0     0                     179.0000000  163.48010  8.67%     -    0s
H    0     0                     178.0000000  163.48010  8.16%     -    0s
H    0     0                     177.0000000  163.48010  7.64%     -    0s
     0     0  163.48010    0  286  177.00000  163.48010  7.64%     -    0s
H    0     0                     176.0000000  163.48010  7.11%     -    0s
H    0     0                     175.0000000  163.48010  6.58%     -    0s
H    0     0                     174.0000000  163.48010  6.05%     -    0s
H    0     0                     173.0000000  163.48010  5.50%     -    0s
     0     0  163.48010    0  286  173.00000  163.48010  5.50%     -    0s
H    0     0                     172.0000000  163.48010  4.95%     -    1s
H    0     0                     170.0000000  163.48010  3.84%     -    1s
     0     0  163.48010    0  237  170.00000  163.48010  3.84%     -    1s
     0     0  163.48010    0  218  170.00000  163.48010  3.84%     -    1s
     0     2  163.48010    0  212  170.00000  163.48010  3.84%     -    1s
H 1556   707                     169.0000000  163.48010  3.27%  31.8    4s
  1562   711  164.00000  166  289  169.00000  163.49255  3.26%  31.7    5s
  1614   748  167.20000   28  229  169.00000  167.20000  1.07%  45.1   10s
  2296  1063  167.20000   79  230  169.00000  167.20000  1.07%  65.9   15s
H 2891   851                     168.0000000  167.20000  0.48%  79.1   19s
```

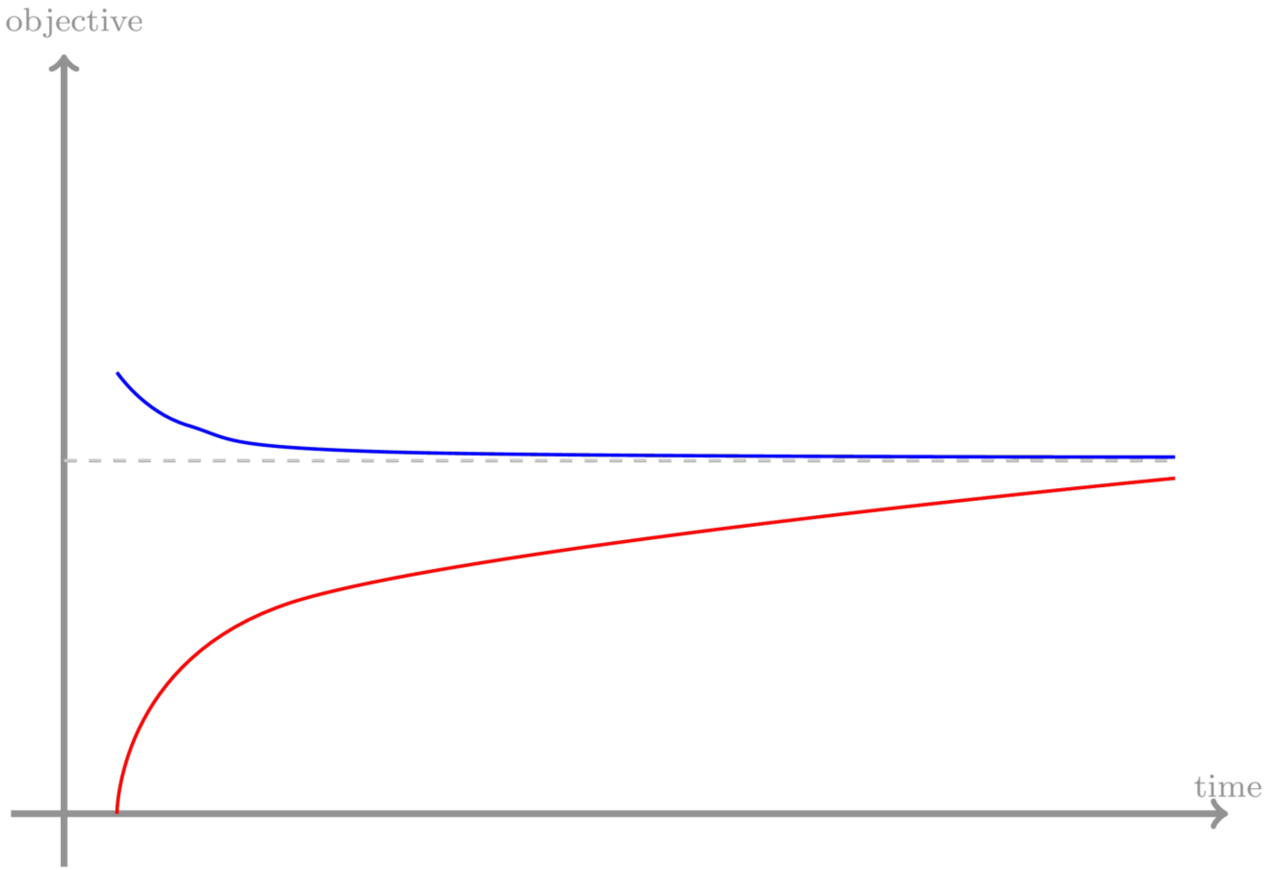
Example 2

CASE 2:

- Upper bound seems OK
- Internal heuristics + user input do the job early on
- Lower bound is horrible!
  - Cuts
  - More pre-processing
  - Reformulate

Try:

- Cuts
- Presolve



Example 2

- After using MIPFocus=3
- Solves twice as fast

```
Root relaxation: objective 1.634801e+02, 2102 iterations, 0.04 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  163.48010    0  175  205.00000  163.48010  20.3%     -    0s
H    0     0                     182.0000000  163.48010  10.2%     -    0s
H    0     0                     181.0000000  163.48010  9.68%     -    0s
H    0     0                     180.0000000  163.48010  9.18%     -    0s
H    0     0                     179.0000000  163.48010  8.67%     -    0s
H    0     0                     177.0000000  163.48010  7.64%     -    0s
H    0     0                     176.0000000  163.48010  7.11%     -    0s
     0     0  163.48010    0  223  176.00000  163.48010  7.11%     -    0s
H    0     0                     175.0000000  163.48010  6.58%     -    1s
H    0     0                     174.0000000  163.48010  6.05%     -    1s
H    0     0                     173.0000000  163.48010  5.50%     -    1s
     0     0  163.48010    0  210  173.00000  163.48010  5.50%     -    1s
H    0     0                     172.0000000  163.48010  4.95%     -    1s
     0     0  163.48010    0  259  172.00000  163.48010  4.95%     -    1s
     0     0  163.48010    0  250  172.00000  163.48010  4.95%     -    1s
     0     2  163.48010    0  211  172.00000  163.48010  4.95%     -    2s
  1108   172 infeasible  116       172.00000  163.53150  4.92%  63.2    5s
H 2036   398                     171.0000000  163.53610  4.36%  60.5    6s
H 2143   484                     170.0000000  163.53610  3.80%  60.7    6s
H 2270   470                     169.0000000  167.20000  1.07%  59.5   10s
H 2270   446                     168.0000000  167.20000  0.48%  59.5   10s
```

Example 2

- After trying cuts=1
- Solves three times as fast

```
Root relaxation: objective 1.634801e+02, 582 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  163.48010    0  183  205.00000  163.48010  20.3%     -    0s
H    0     0                     179.0000000  163.48010  8.67%     -    0s
H    0     0                     178.0000000  163.48010  8.16%     -    0s
H    0     0                     177.0000000  163.48010  7.64%     -    0s
     0     0  163.48010    0  286  177.00000  163.48010  7.64%     -    0s
H    0     0                     176.0000000  163.48010  7.11%     -    0s
H    0     0                     175.0000000  163.48010  6.58%     -    0s
H    0     0                     174.0000000  163.48010  6.05%     -    0s
H    0     0                     173.0000000  163.48010  5.50%     -    0s
     0     0  163.48010    0  286  173.00000  163.48010  5.50%     -    0s
H    0     0                     172.0000000  163.48010  4.95%     -    1s
H    0     0                     170.0000000  163.48010  3.84%     -    1s
     0     0  163.48010    0  237  170.00000  163.48010  3.84%     -    1s
     0     0  163.48010    0  218  170.00000  163.48010  3.84%     -    1s
     0     2  163.48010    0  212  170.00000  163.48010  3.84%     -    1s
H  655   183                     169.0000000  163.48010  3.27%  19.4    3s
   949   331 infeasible  197       169.00000  163.48010  3.27%  29.9    5s
H  950   330                     168.0000000  163.48010  2.69%  29.9    5s
```


Why tune after when upgrading to a new version?

Gurobi 9.5.2

- MIPFocus=3 outperforms Gurobi default settings and Cuts=1

```
Root relaxation: objective 1.634801e+02, 2102 iterations, 0.03 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  163.48010    0  175  205.00000  163.48010  20.3%     -    0s
H    0     0                     183.0000000  163.48010  10.7%     -    0s
H    0     0                     181.0000000  163.48010  9.68%     -    0s
H    0     0                     178.0000000  163.48010  8.16%     -    0s
H    0     0                     177.0000000  163.48010  7.64%     -    0s
H    0     0                     176.0000000  163.48010  7.11%     -    0s
H    0     0                     175.0000000  163.48010  6.58%     -    0s
H    0     0                     174.0000000  163.48010  6.05%     -    0s
     0     0  163.48010    0  238  174.00000  163.48010  6.05%     -    0s
H    0     0                     173.0000000  163.48010  5.50%     -    1s
H    0     0                     172.0000000  163.48010  4.95%     -    1s
     0     0  163.48010    0  223  172.00000  163.48010  4.95%     -    1s
     0     0  163.48010    0  258  172.00000  163.48010  4.95%     -    1s
     0     0  163.48010    0  235  172.00000  163.48010  4.95%     -    1s
     0     2  163.48010    0  211  172.00000  163.48010  4.95%     -    2s
H  313    29                     171.0000000  164.00000  4.09%   135    3s
H  347    40                     170.0000000  164.00000  3.53%   163    4s
H  352    40                     169.0000000  164.00000  2.96%   165    4s
   615   147  164.00000  174  183  169.00000  164.00000  2.96%   205    5s
  2088   680  167.16667  134  158  169.00000  167.16667  1.08%   160   10s
  2118   707  167.20000   27  115  169.00000  167.20000  1.07%  32.5   15s
H 2186   717                     168.0000000  167.20000  0.48%  51.3   19s
```

Gurobi 10 & 11

- Gurobi defaults is as good as Gurobi 9.5.2 MIPFocus=3
- Cuts=1 outperforms Gurobi default settings and MIPFocus=3

```
Root relaxation: objective 1.634801e+02, 582 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  163.48010    0  183  205.00000  163.48010  20.3%     -    0s
H    0     0                     179.0000000  163.48010  8.67%     -    0s
H    0     0                     178.0000000  163.48010  8.16%     -    0s
H    0     0                     177.0000000  163.48010  7.64%     -    0s
     0     0  163.48010    0  286  177.00000  163.48010  7.64%     -    0s
H    0     0                     176.0000000  163.48010  7.11%     -    0s
H    0     0                     175.0000000  163.48010  6.58%     -    0s
H    0     0                     174.0000000  163.48010  6.05%     -    0s
H    0     0                     173.0000000  163.48010  5.50%     -    0s
     0     0  163.48010    0  286  173.00000  163.48010  5.50%     -    0s
H    0     0                     172.0000000  163.48010  4.95%     -    1s
H    0     0                     170.0000000  163.48010  3.84%     -    1s
     0     0  163.48010    0  237  170.00000  163.48010  3.84%     -    1s
     0     0  163.48010    0  218  170.00000  163.48010  3.84%     -    1s
     0     2  163.48010    0  212  170.00000  163.48010  3.84%     -    1s
H  655   183                     169.0000000  163.48010  3.27%  19.4    3s
   949   331 infeasible  197       169.00000  163.48010  3.27%  29.9    5s
H  950   330                     168.0000000  163.48010  2.69%  29.9    5s
```


Example 1+2: Main takeaways

- First decide which bound needs attention
- With new Gurobi versions, always start tuning using *defaults*

Example 3

- This exercise is based on the [Hawea model](https://miplib.zib.de/instance_details_neos-3592146-hawea.html) from miplib.
- Runs with this model take between 1.5 and 2 minutes by default, so you can do these runs yourself.

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 7345804.86    0  208 7.1656e+14 7345804.86   100%     -    0s
H    0     0                    7.990584e+07 7345804.86  90.8%     -    0s
H    0     0                    7.948122e+07 7345804.86  90.8%     -    0s
H    0     0                    7.850507e+07 7345804.86  90.6%     -    0s
H    0     0                    7.179889e+07 7345804.86  89.8%     -    0s
H    0     0                    4.011613e+07 8596299.40  78.6%     -    0s
...
     0     0 1.1582e+07    0  687 3.1712e+07 1.1582e+07  63.5%     -    0s
     0     0 1.1617e+07    0  690 3.1712e+07 1.1617e+07  63.4%     -    0s
     0     0 1.1628e+07    0  697 3.1712e+07 1.1628e+07  63.3%     -    0s
     0     0 1.1636e+07    0  735 3.1712e+07 1.1636e+07  63.3%     -    0s
     0     0 1.1795e+07    0  929 3.1712e+07 1.1795e+07  62.8%     -    0s
     0     0 1.1805e+07    0  929 3.1712e+07 1.1805e+07  62.8%     -    0s
H    0     0                    3.125259e+07 1.1805e+07  62.2%     -    0s
H    0     0                    3.125241e+07 1.1805e+07  62.2%     -    0s
H    0     2                    3.125233e+07 1.1805e+07  62.2%     -    0s
     0     2 1.1805e+07    0  926 3.1252e+07 1.1805e+07  62.2%     -    0s
H   29    35                    2.917931e+07 1.1805e+07  59.5%   760    2s
H   61    64                    2.657340e+07 1.1805e+07  55.6%   540    2s
...
H  386   217                    1.571950e+07 1.2917e+07  17.8%   232    5s
H  391   221                    1.569036e+07 1.2917e+07  17.7%   233    5s
H  687   355                    1.567685e+07 1.3150e+07  16.1%   202    7s
H  758   414                    1.563173e+07 1.3150e+07  15.9%   205    7s
H 1156   534                    1.560619e+07 1.3316e+07  14.7%   176    9s
  1684   602     cutoff   72      1.5606e+07 1.3316e+07  14.7%   138   10s
H 1858   568                    1.554339e+07 1.3316e+07  14.3%   130   10s
H 1975   559                    1.554338e+07 1.3316e+07  14.3%   124   11s
H 2012   526                    1.554303e+07 1.3316e+07  14.3%   123   12s
  7109  2496 1.5134e+07   35   18 1.5543e+07 1.3586e+07  12.6%  66.6   15s
 24445  8104 1.5322e+07   37   26 1.5543e+07 1.4188e+07  8.72%  39.5   20s
...
 29807  8936 1.4774e+07   30 1753 1.5543e+07 1.4320e+07  7.87%  37.3   45s
 29820  8944 1.5422e+07   37 1769 1.5543e+07 1.4320e+07  7.87%  37.3   50s
....
 35448  8335 1.5407e+07   38  173 1.5529e+07 1.4320e+07  7.78%  58.3   85s
H35557  7852                    1.550345e+07 1.4320e+07  7.63%  58.5   85s
 37064  7631 1.5392e+07   33  129 1.5503e+07 1.4320e+07  7.63%  61.7   90s
 38764  7325 1.4611e+07   38  451 1.5503e+07 1.4320e+07  7.63%  65.4   95s
 41085  6537 1.4697e+07   35  151 1.5503e+07 1.4442e+07  6.85%  68.5  101s
 42931  5926 1.5041e+07   37  190 1.5503e+07 1.4613e+07  5.75%  70.7  105s
H45341  4435                    1.547377e+07 1.4822e+07  4.21%  73.0  109s
*45342  3999              38    1.546580e+07 1.4822e+07  4.16%  73.0  109s
 46003  3568 1.5278e+07   41  156 1.5466e+07 1.4868e+07  3.86%  73.4  111s
 49530  1276 1.5419e+07   45  430 1.5466e+07 1.5088e+07  2.44%  72.8  115s
```


Example 3

What to test

- `Presolve (0, 1 or 2)`
- `MIPFocus (1, 2 or 3)`
- `Cuts (0, 1, 2 or 3)`

`Cuts=0` yields more than 5x speedup

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 7345804.86    0  208 7.1656e+14 7345804.86   100%     -    0s
H    0     0                    7.990584e+07 7345804.86  90.8%     -    0s
H    0     0                    7.948122e+07 7345804.86  90.8%     -    0s
H    0     0                    7.850507e+07 7345804.86  90.6%     -    0s
H    0     0                    7.179889e+07 7345804.86  89.8%     -    0s
     0     0 7525946.79    0  208 7.1799e+07 7525946.79  89.5%     -    0s
H    0     0                    7.179841e+07 7525946.79  89.5%     -    0s
H    0     0                    7.101213e+07 7525946.79  89.4%     -    0s
H    0     2                    7.089075e+07 7525946.79  89.4%     -    0s
     0     2 7525946.79    0  208 7.0891e+07 7525946.79  89.4%     -    0s
H   31    39                    4.244898e+07 8470498.05  80.0%   158    0s
H   78    87                    4.217187e+07 8628284.61  79.5%   170    0s
H   79    87                    4.031236e+07 8628284.61  78.6%   169    0s
...
H 7266  2296                    1.567737e+07 1.3444e+07  14.2%  46.9    5s
H 7268  2098                    1.558982e+07 1.3444e+07  13.8%  46.9    5s
H10372  3427                    1.556375e+07 1.3662e+07  12.2%  42.7    6s
H15024  4989                    1.554157e+07 1.3839e+07  11.0%  38.3    7s
 27850  8413 1.4715e+07   46   81 1.5542e+07 1.4202e+07  8.62%  32.8   10s
H29250  8198                    1.552163e+07 1.4238e+07  8.27%  32.7   10s
H29252  8063                    1.551125e+07 1.4238e+07  8.21%  32.7   10s
H29270  7947                    1.550345e+07 1.4238e+07  8.16%  32.6   11s
H29299  7261                    1.546580e+07 1.4238e+07  7.94%  32.7   11s
 52357  7991     cutoff   38      1.5466e+07 1.5130e+07  2.17%  30.6   15s
```


Example 4

- This exercise is real case model . We were given a model that takes longer using Gurobi than one of our competitors.
- Average runtimes with default settings using are around 186s, mostly spent in the root relaxation.
- Trying Method parameters to explore which method best fit

```
Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   34697    6.1244562e+04   0.000000e+00   6.704184e+07      5s
   48981    3.2502279e+04   0.000000e+00   1.202921e+08     10s
   58034    2.6211348e+04   0.000000e+00   8.309235e+07     15s
   66138    2.0088283e+04   0.000000e+00   2.454424e+08     20s
   72909    1.5873109e+04   0.000000e+00   1.840156e+08     25s
   79048    1.3433434e+04   0.000000e+00   1.885575e+08     30s
   87913    8.4290853e+03   0.000000e+00   1.985723e+07     35s
   92717    6.3783417e+03   0.000000e+00   5.087409e+07     40s
   96641    6.1779853e+03   0.000000e+00   1.150974e+08     45s
  100722    5.8994659e+03   0.000000e+00   5.005310e+08     50s
  105487    5.1598514e+03   0.000000e+00   1.493867e+07     55s
  109817    4.7312521e+03   0.000000e+00   1.804181e+07     60s
  113762    4.6536219e+03   0.000000e+00   3.288902e+07     65s
  117803    4.2505445e+03   0.000000e+00   2.367348e+07     70s
  121639    4.1103480e+03   0.000000e+00   5.111156e+07     75s
  125379    4.0203901e+03   0.000000e+00   4.242315e+07     80s
  129498    3.8670063e+03   0.000000e+00   2.723751e+07     85s
  133463    3.7987049e+03   0.000000e+00   4.546264e+07     90s
  137700    3.7412582e+03   0.000000e+00   1.860931e+08     95s
  141755    3.6749783e+03   0.000000e+00   5.518917e+07    100s
  145819    3.6208250e+03   0.000000e+00   9.849596e+07    105s
  149825    3.4301639e+03   0.000000e+00   1.265932e+08    110s
  154705    3.3319682e+03   0.000000e+00   2.934038e+08    115s
  158952    3.2576891e+03   0.000000e+00   2.541347e+07    120s
  163349    3.2154306e+03   0.000000e+00   1.223512e+08    125s
  168678    3.1506356e+03   0.000000e+00   2.567265e+07    130s
  173172    3.0846458e+03   0.000000e+00   1.859406e+09    135s
Concurrent spin time: 0.00s
```


Example 4

- Using `Method=2` (Barrier) for the root relaxation for this model
- LP time reduced
  - 135s -> 10s
- Total solution time reduced
  - 186s -> 30s

```
                       Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0  -3.26241426e+06 -1.48322406e+08  1.03e+04 1.67e+02  1.13e+04     5s
   1  -5.86087009e+05 -1.33377506e+08  2.43e+03 1.08e+02  2.85e+03     5s
   2  -1.85498122e+05 -9.20877670e+07  8.09e+02 5.47e+00  9.71e+02     5s
   3  -5.37175921e+04 -4.75753312e+07  2.63e+02 8.21e-01  3.41e+02     5s
   4   1.07927311e+04 -1.65343656e+07  3.13e+01 7.72e-13  6.77e+01     5s
   5   2.40185432e+04 -8.00454047e+06  6.82e+00 7.24e-13  2.89e+01     5s
   6   3.48325758e+04 -5.21210857e+06  6.84e-01 8.24e-13  1.56e+01     5s
   7   3.82227684e+04 -3.45723745e+06  3.45e-01 8.38e-13  9.67e+00     6s
   8   3.94405592e+04 -2.68628221e+06  1.97e-01 1.05e-12  7.13e+00     6s
...
  23   8.78807094e+01 -2.76619959e+03  1.03e-05 8.69e-13  6.41e-03     8s
  24  -9.39561391e+01 -1.59911655e+03  4.76e-06 8.70e-13  3.38e-03     8s
  25  -1.46363338e+02 -1.42103995e+03  3.70e-06 8.61e-13  2.86e-03     8s
  26  -2.25188637e+02 -9.57030939e+02  2.19e-06 6.68e-13  1.64e-03     8s
  27  -2.29121350e+02 -9.20556013e+02  2.12e-06 7.74e-13  1.55e-03     8s
  28  -3.10484728e+02 -7.59502256e+02  7.07e-07 7.69e-13  1.01e-03     8s
  29  -3.15260063e+02 -5.72249339e+02  6.40e-07 9.60e-13  5.77e-04     8s
  30  -3.50528259e+02 -4.79759660e+02  2.19e-07 7.92e-13  2.90e-04     8s
  31  -3.63815873e+02 -4.19731247e+02  5.98e-08 7.19e-13  1.25e-04     9s
  32  -3.68082088e+02 -3.81969185e+02  1.49e-08 8.75e-13  3.11e-05     9s
  33  -3.70317729e+02 -3.72174258e+02  1.34e-09 8.25e-13  4.16e-06     9s
  34  -3.70634151e+02 -3.70982614e+02  1.02e-10 9.65e-13  7.80e-07     9s
  35  -3.70636385e+02 -3.70861113e+02  2.80e-09 7.60e-13  5.03e-07     9s
  36  -3.70639205e+02 -3.70684903e+02  3.06e-09 8.43e-13  1.02e-07     9s
  37  -3.70644027e+02 -3.70645769e+02  7.63e-09 8.73e-13  3.90e-09     9s
  38  -3.70645291e+02 -3.70645293e+02  5.72e-12 8.18e-13  3.91e-12    10s

Barrier solved model in 38 iterations and 9.56 seconds (7.31 work units)
Optimal objective -3.70645291e+02
```


Exercises 3+4: Main takeaways

- It’s not always clear which bound needs attention. Try both using MIPFocus!
- Pay attention to where the time is spent
- Carefully check the LP method used; depends on #threads and model!

Some takeaways

Situation	First step	Additional steps
Presolve takes a lot of time, without making progress	• Presolve=0|1 • Limit PrePasses*	• If a QP, try PreMIQCPForm and PreQLinearize • PreDual and Aggregate sometimes also make a big difference
LP relaxation accounts for a lot of time	• Explicitly set Method • Try No Relaxation Heuristic (NoRelHeurTime)	• Try different Presolve parameter values • Examine additional simplex/barrier parameters
Root node takes a lot of time	Cuts=0	Limit CutPasses*
Difficult node LPs (low node throughput)	Try all LP methods (NodeMethod)	Different pricing schemes for simplex (e.g. SimplexPricing=2 (devex))
Primal bound does not move after many nodes	• Heuristics=0.25|0.5 • MIPFocus=1	• Check which heuristics help (e.g. RINS) and increase selectively • No Relaxation Heuristic • Provide initial solution, even if bad • BranchDir
No solution is found	• No Relaxation Heuristic (NoRelHeurTime) • Zero out objective	• Use feasRelax() to create a feasible relaxation of the model
Dual bound does not move	• MIPFocus=2|3 • Cuts=2|3 • Presolve=2	Different pricing schemes for simplex (e.g. SimplexPricing=2 (devex))

Automated tuning

Automated tools to help with tuning

Parameter Tuning Tool

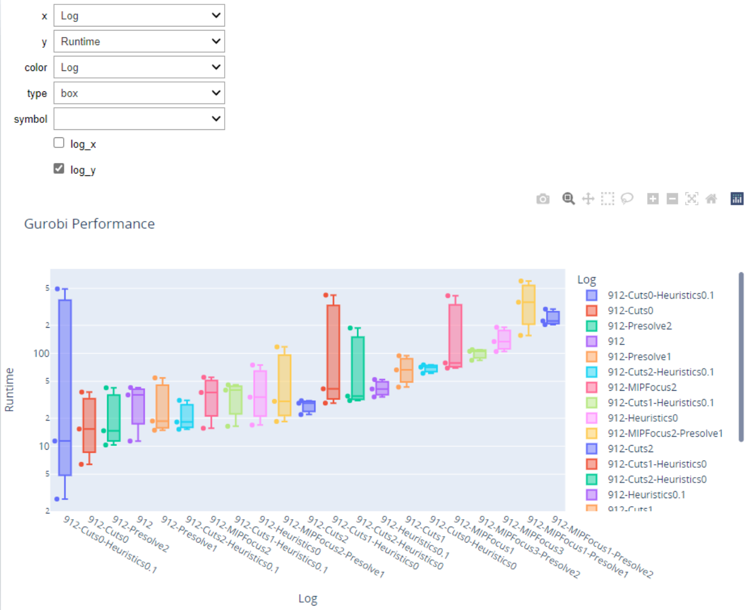
Performs multiple solves on your model with different parameter settings to search for settings that improve performance

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

Tested 142 parameter sets in 198.66s

Baseline parameter set: mean runtime 2.18s

        Default parameters

 # Name              0        1        2      Avg      Max  Std Dev
 0 MISC07        1.76s    2.56s    2.21s    2.18s    2.56s     0.33

Improved parameter set 1 (mean runtime 0.46s):

        Heuristics 0
        MIPFocus 2
        Cuts 0
        PrePasses 5

 # Name              0        1        2      Avg      Max  Std Dev
 0 MISC07        0.46s    0.45s    0.47s    0.46s    0.47s     0.01
```

For more information, please see the Parameter Tuning Tool documentation

Extract and Visualize log information

Use Gurobi's open source tool: grblogtools



For more information, see this video webinar and the grblogtools github.

How to use the Parameter tuning tool

Use the command line client (`grbtune`)

Example:

```
grbtune Logfile=tune.log TuneOutput=3
        TuneTrials=5 TuneTimeLimit=432000
        TimeLimit=600 TuneCriterion=2
        Method=1 model1.mps model2.mps
```

- Tune two models for 5 days
- Show full logfile output of each run,
- Test each model with 5 different random seeds
- Write output to tune.log
- Run each model for up to 10 minutes
- Always use dual simplex
- Tune for best feasible solution within the time limit

Use the API to tune a model

Example

```
m=read('model.mps')
m.params.Method=1
m.params.TuneTimeLimit=86400
m.tune()
```

- Tune for one day
- Use the dual simplex method

Tuning tool output

Tries multiple parameter combinations, looks for improving set

```
Testing candidate parameter set 16...

        MIPFocus 2
        VarBranch 1

Solving with random seed #1 ... runtime 3.01s+

Progress so far: baseline runtime 3.91s, best runtime 1.83s
Total elapsed tuning time 99s (18s remaining)
```

Reports best results found when completes

```
Improved parameter set 1 (mean runtime 136.14s):

        MIPFocus 2
        Cuts 2
        PrePasses 8
```

#	Name	0	1	2	Avg	Max	Std Dev
0	a1c1s1_co	121.69s	133.19s	153.53s	136.14s	153.53s	13.16

Tuning metrics

- Primary tuning criterion minimizes the runtime (wall-clock time)
- When MIP models that don't solve to optimality within the specified time limit, we need a secondary criterion
- Use **`TuneCriterion`** parameter:
  - `TuneCriterion=0` – Ignores secondary criterion
  - `TuneCriterion=1` – Use optimality gap as secondary criterion (current default)
  - `TuneCriterion=2` – Use objective of the best feasible solution found
  - `TuneCriterion=3` – Use best objective bound

Don‘t overtune…

Example output:

```
Improved parameter set 1 (MIP gap 2.33% ):

        SimplexPricing 3
        Heuristics 0.001
        MIPFocus 2
        RINS 2500
        VarBranch 0
        CutPasses 5
        PrePasses 1

Improved parameter set 2 (MIP gap 2.54% ):

        SimplexPricing 3
        Heuristics 0.001
        MIPFocus 2
        RINS 500
        VarBranch 0
        PrePasses 1

Improved parameter set 3 (MIP gap 2.70% ):

        SimplexPricing 3
        Heuristics 0.001
        MIPFocus 2
        VarBranch 0
        PrePasses 1

Improved parameter set 4 (MIP gap 3.94% ):

        Heuristics 0.001
        MIPFocus 2
        VarBranch 0
        PrePasses 1

Improved parameter set 5 (MIP gap 4.56% ):

        MIPFocus 2
        VarBranch 0
        PrePasses 1

Improved parameter set 6 (MIP gap 7.33% ):

        PumpPasses 10
        VarBranch 0

Improved parameter set 7 (MIP gap 12.4% ):

        Method 0
```

Best practices

TuneTrials parameter

Never do a single trial

Three trials may not be large enough

More trials may be needed for

- Models with high performance variability
- Models solve within seconds

Understand parameters

Understand why certain parameters are helpful

Remember, Gurobi Experts can help

Only set parameters that really help performance

Look for 10-20% performance gain

Avoid random results

TuneTimeLimit parameter

Use enough time to explore the parameter space

Examine at least a few hundred parameter combinations

GUROBI OPTIMIZATION

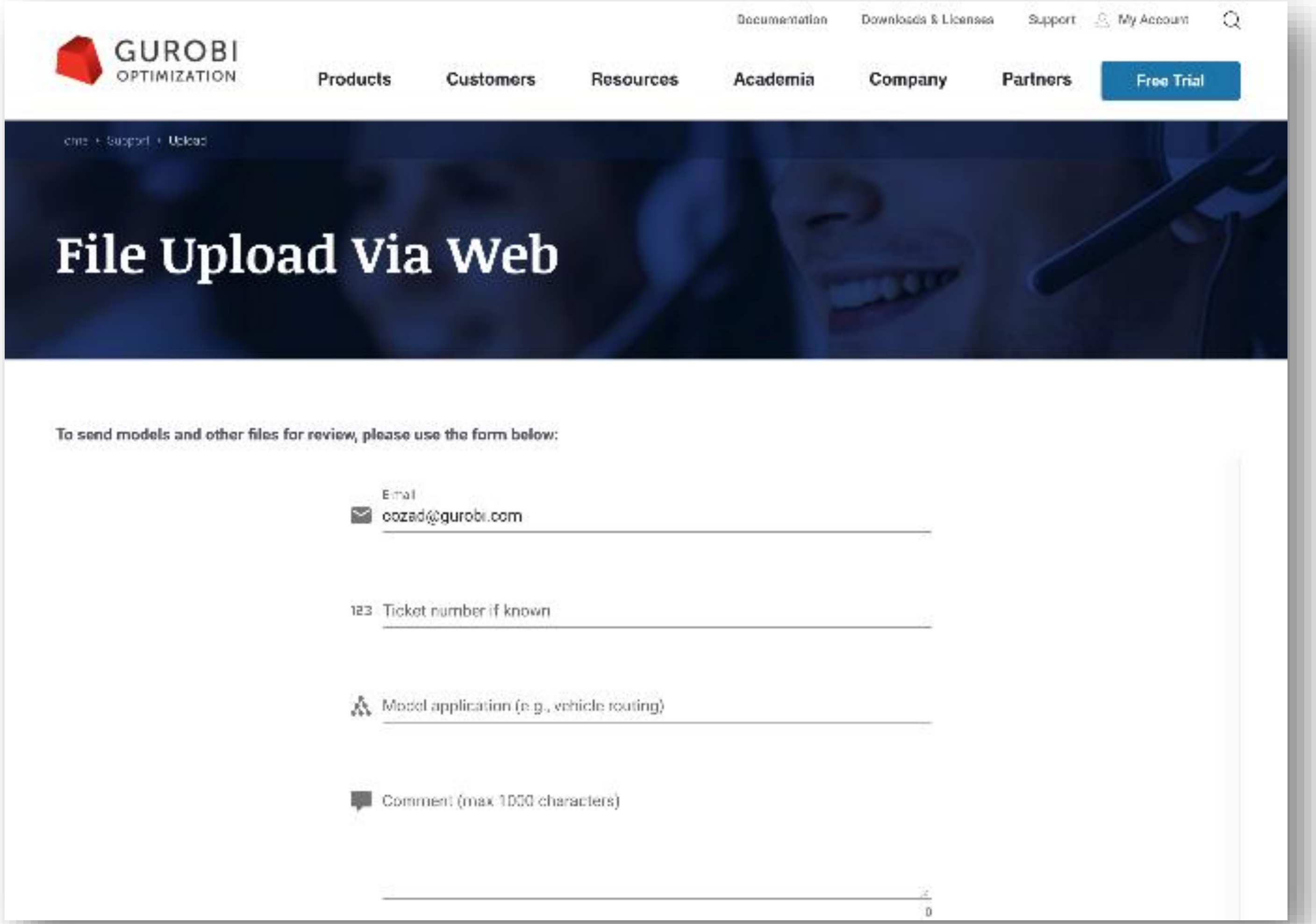
Selecting and sending models to tune

Consider

- Which models to send?
- How many instances to send?
  We recommend 2-3.

How to send models to Gurobi for Tuning

- Export your models as an MPS or LP file
- Send compressed files through our upload page
- Add details to help us understand your case:
  - Model application
  - Termination criteria (MIPGap, TimeLimit, etc.)
  - Current parameters used
  - What you are hoping to achieve
- For more details see the instructions in
  [How do I upload or send files to Gurobi?]



Thank you

Questions?

Jennifer Locke
Manager, Technical Account Management, Americas
locke@gurobi.com
www.gurobi.com

Gurobi 11.0
Every Solution, Globally Optimized

Agenda

1:30 - 1:40 Welcome Remarks

1:40 - 2:25 What's new in Gurobi 11

2:25 - 3:10 Hidden Gems

3:10 - 3:30 Break

3:30 - 4:15 Performance Tuning

4:15 - 5:00 Lessons in Successful Delivery from a Large-scale Workforce Scheduling Project

5:00 - 5:30 Q&A

Please join us for Happy Hour following the event

