Advances in Model-Based Optimization with AMPL

Robert Fourer

`4er@ampl.com`

AMPL Optimization Inc.
www.ampl.com — +1 773-336-AMPL

INFORMS Annual Meeting

Indianapolis, Indiana — 15-19 October 2022
Technology Tutorials Track

Advances in Model-Based Optimization with AMPL

Optimization has been fundamental to OR and Analytics for as long as there have been computers, yet we are still finding ways to make optimization software more natural to use, faster to run, and easier to integrate with application systems. This presentation offers a quick tour of ways that AMPL's modeling framework has been enhanced to support optimization in today's challenging applications. Topics include:

- Expressing objectives and constraints more directly and understandably
- Exchanging data and results more directly and efficiently, with spreadsheets and with database systems
- Building better interfaces to applications using snapshots, callbacks, and other new features of AMPL's APIs for popular programming languages
- Deploying optimization in cloud environments and containers

To complement these feature advances, the presentation concludes by describing ways that AMPL is making model-based optimization more accessible, through the new Community Edition, a rewritten NEOS Server client, and free Model Colaboratory examples for teaching and learning optimization.

Outline

Model-based optimization

New in model-based optimization with AMPL

- *more natural:* Writing models more like you think about them
- *faster, easier:* Exchanging data and results directly and efficiently with spreadsheets and database systems
- *faster:* Interfacing to enterprise systems through APIs for popular programming languages
- *easier:* Deploying optimization in containers and in the cloud

New ways to try AMPL now

- Community Edition
- NEOS Server client
- Model Colaboratory

Example:

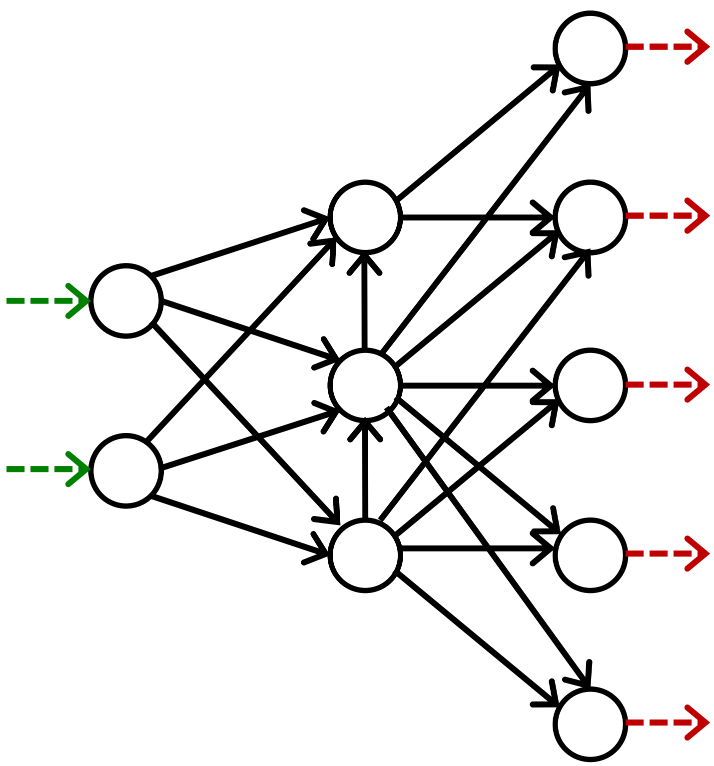
Multi-Product Network Flow

Motivation

- Ship products efficiently to meet demands

Context

- a transportation network
  - ∗ nodes ◯ representing cities
  - ∗ arcs ⟶ representing roads
- supplies - - -> at nodes
- demands - - -> at nodes
- capacities on arcs
- shipping costs on arcs



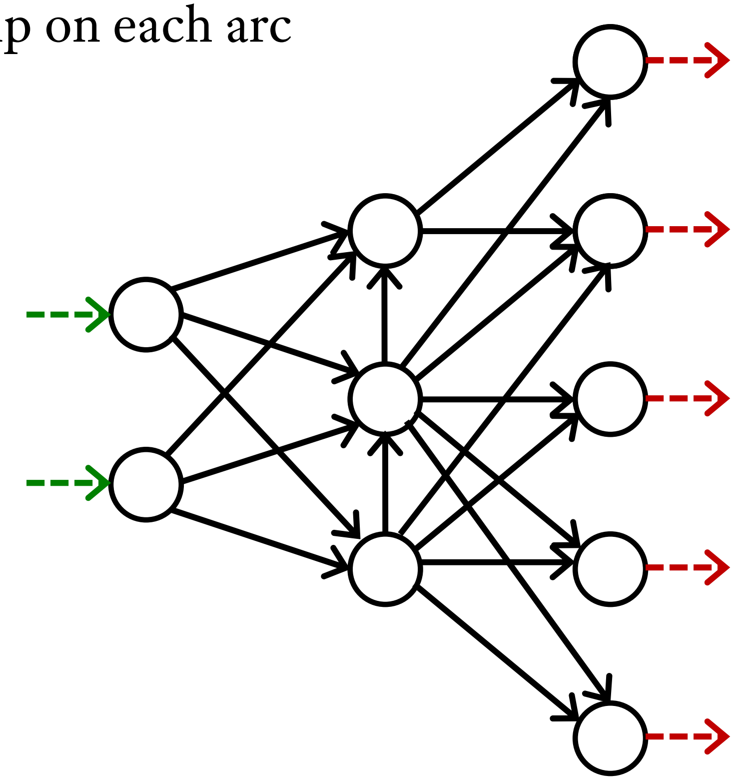
Example:
Multi-Product Network Flow

Decide

- how much of each product to ship on each arc

So that

- shipping costs are kept low
- shipments on each arc respect capacity of the arc
- supplies, demands, and shipments are in balance at each node



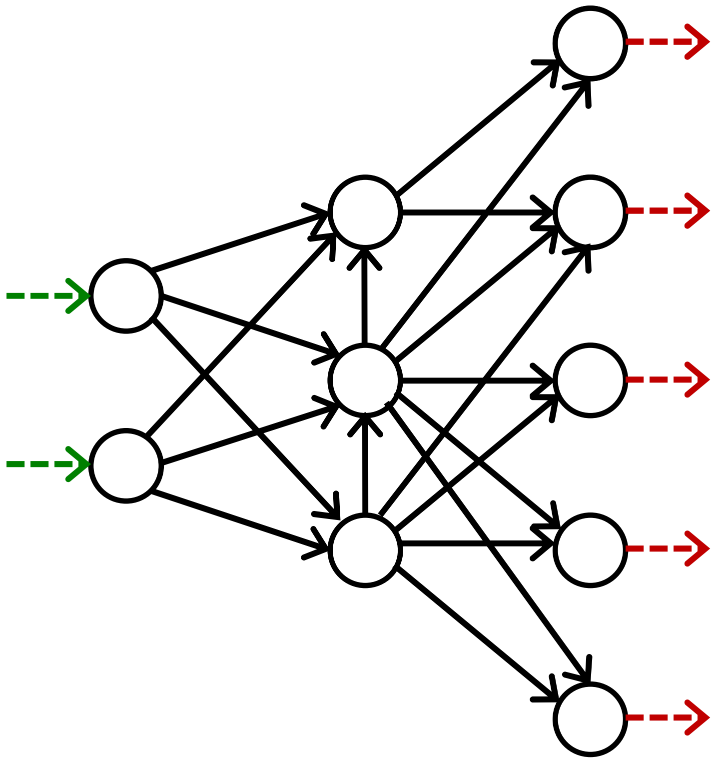
Example with complications: Multi-Product Network Flow

Decide also

- whether to use each arc

So that

- variable plus fixed shipping costs are kept low
- shipments are not too small
- not too many arcs are used



Model-Based Optimization

Formulate a minimum shipping cost model

- *decision variables:* What arcs are used and how much is shipped
- *objective:* Total fixed and variable costs
- *constraints:* Equations that the variables must satisfy to meet the requirements of the problem

Apply model-based optimization software

- *modeling language:* Write a formulation that a computer system can read
- *data:* Read costs, capacities, supplies, demands, and limits that define a specific case to be solved
- *solver:* Send to an off-the-shelf optimization engine that accepts a broad class of problems

Multi-Product Flow

Formulation *(data)*

Given

P set of products

N set of network nodes

$A \subseteq N \times N$ set of arcs connecting nodes

and

u_{ij} capacity of arc from i to j, for each $(i,j) \in A$

s_{pj} supply/demand of product p at node j, for each $p \in P$, $j \in N$
> 0 implies supply, < 0 implies demand

c_{pij} cost per unit to ship product p on arc (i,j),
for each $p \in P$, $(i,j) \in A$

d_{ij} fixed cost for using the arc from i to j, for each $(i,j) \in A$

m smallest total shipments on any arc that is used

n largest number of arcs that may be used

Linearized Formulation *(variables, objective)*

Determine

X_{pij} amount of commodity p to be shipped on arc (i, j), for each $p \in P$, $(i, j) \in A$

Y_{ij} 1 if any amount is shipped from node i to node j, 0 otherwise, for each $(i, j) \in A$

to minimize

$$\sum_{p \in P} \sum_{(i,j) \in A} c_{pij} X_{pij} + \sum_{(i,j) \in A} d_{ij} Y_{ij}$$

total cost of shipments

Multi-Product Flow

Linearized Formulation *(constraints)*

Subject to

$\sum_{p \in P} X_{pij} \leq u_{ij} Y_{ij},$ for all $(i,j) \in A$

when the arc from node i to node j is used for shipping,
total shipments must not exceed capacity, and Y_{ij} must be 1

$\sum_{p \in P} X_{pij} \geq m Y_{ij},$ for all $(i,j) \in A$

when the arc from node i to node j is used for shipping,
total shipments from i to j must be at least m

$\sum_{(i,j) \in A} X_{pij} + s_{pj} = \sum_{(j,i) \in A} X_{pji},$ for all $p \in P, j \in N$

shipments in plus supply/demand must equal shipments out

$\sum_{(i,j) \in A} Y_{ij} \leq n$

At most n arcs can be used

Multi-Product Flow

Linearized Model in AMPL

Symbolic data, variables, objective

```
set PRODUCTS;
set NODES;

set ARCS within {NODES,NODES};
param capacity {ARCS} >= 0;

param inflow {PRODUCTS,NODES};
param min_ship >= 0;
param max_arcs >= 0;

param var_cost {PRODUCTS,ARCS} >= 0;
var Flow {PRODUCTS,ARCS} >= 0;

param fix_cost {ARCS} >= 0;
var Use {ARCS} binary;

minimize TotalCost:
   sum {p in PRODUCTS, (i,j) in ARCS} var_cost[p,i,j] * Flow[p,i,j] +
   sum {(i,j) in ARCS} fix_cost[i,j] * Use[i,j];
```

Multi-Product Flow

Linearized Model in AMPL

Constraints

```
subject to Capacity {(i,j) in ARCS}:
   sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j] * Use[i,j];

subject to Min_Shipment {(i,j) in ARCS}:
   sum {p in PRODUCTS} Flow[p,i,j] >= min_ship * Use[i,j];

subject to Conservation {p in PRODUCTS, j in NODES}:
   sum {(i,j) in ARCS} Flow[p,i,j] + inflow[p,j] =
   sum {(j,i) in ARCS} Flow[p,j,i];

subject to Max_Used:
   sum {(i,j) in ARCS} Use[i,j] <= max_arcs;
```

$$\sum_{p \in P} X_{pij} \leq u_{ij} Y_{ij}, \text{ for all } (i,j) \in A$$

Data Instance in AMPL Text Format

Limits

```
set PRODUCTS := Bands Coils ;
set NODES := Detroit Denver Boston 'New York' Seattle ;

param: ARCS: capacity:

         Boston 'New York' Seattle :=
  Detroit   100       80       120
  Denver    120      120       120   ;

param inflow:

         Detroit Denver Boston 'New York' Seattle :=
  Bands      50      60      -50      -50        -10
  Coils      60      40      -40      -30        -30;

param min_ship := 15 ;

param max_arcs := 4 ;
```

Multi-Product Flow

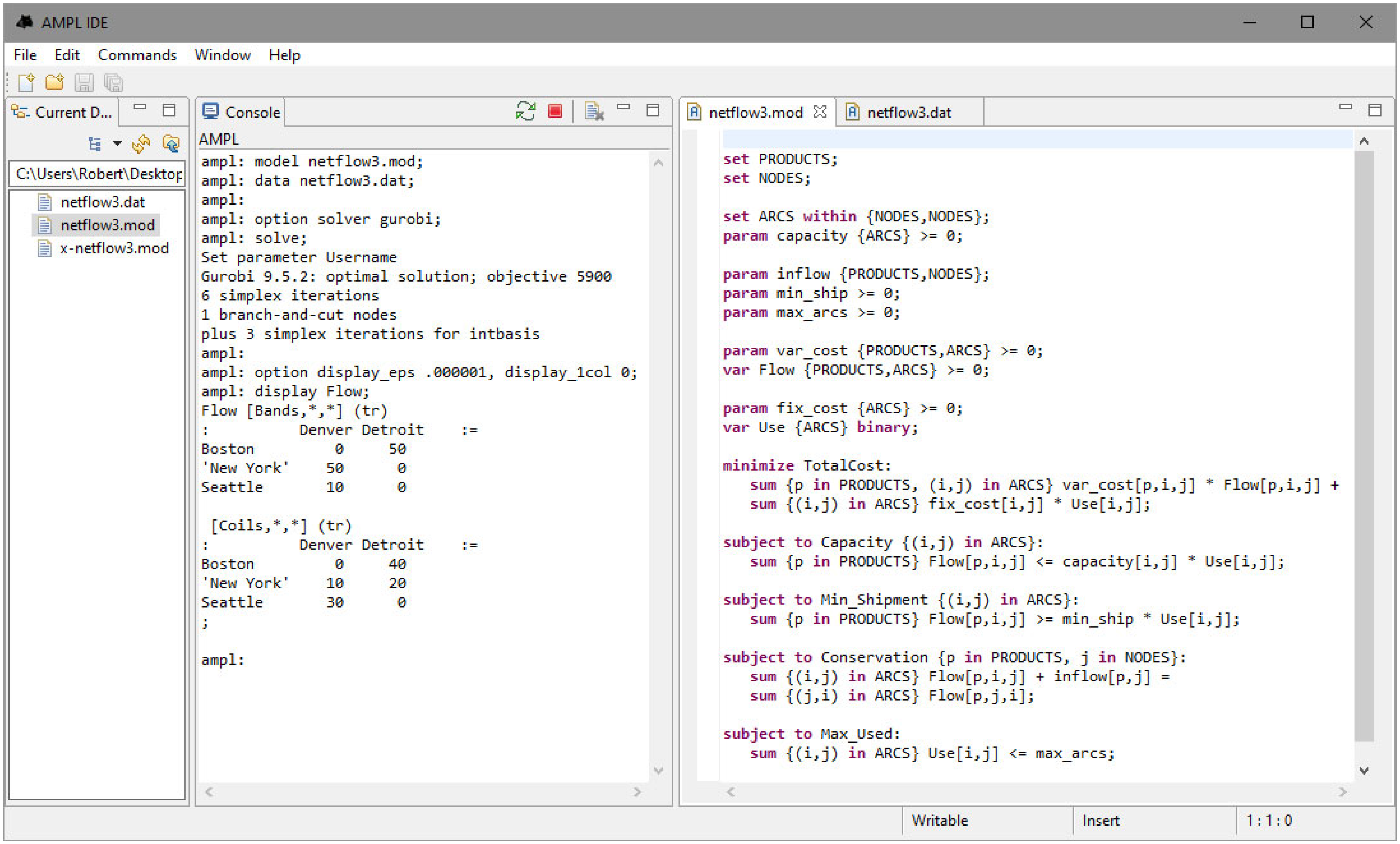
Data Instance in AMPL Text Format

Costs

```
param var_cost:
 [Bands,*,*]:  Boston 'New York' Seattle :=
     Detroit     10        20        60
     Denver      40        40        30
 [Coils,*,*]:  Boston 'New York' Seattle :=
     Detroit     20        20        80
     Denver      60        70        30 ;

param fix_cost default 75 ;
```

Multi-Product Flow

Optimization by a “MIP” Solver *(gurobi)*



Formulating Models
More Like You Think About Them

Describe an optimization problem

- ❖ In a form *you find natural or convenient*
- ❖ Using readily recognized expressions

Send it to a solver

- ❖ In a form *the solver will accept*
- ❖ Relying on the modeling software to translate

Get back a result

- ❖ In the form you originally used

Formulating

Positive Shipments Incur Fixed Costs

Linearized formulation

```
sum {(i,j) in ARCS} fix_cost[i,j] * Use[i,j]
```

Natural formulation

```
sum {(i,j) in ARCS}
   if exists {p in PRODUCTS} Flow[p,i,j] > 0 then fix_cost[i,j]
```

Formulating

Shipments Can't Be Too Small

Linearized formulation

```
sum {p in PRODUCTS} Flow[p,i,j] >= min_ship * Use[i,j];
sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j] * Use[i,j];
```

Natural formulation

```
sum {p in PRODUCTS} Flow[p,i,j] = 0 or
min_ship <= sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j]
```

Can't Use Too Many Arcs

Linearized formulation

```
sum {(i,j) in ARCS} Use[i,j] <= max_arcs;
```

Natural formulation

```
count {(i,j) in ARCS} (sum {p in PRODUCTS} Flow[p,i,j] > 0) <= max_arcs;
```

Formulating

Optimization by Same MIP Solver *(x-gurobi)*

AMPL IDE

File Edit Commands Window Help

Current D... — C:\Users\Robert\Desktop — netflow3.dat, netflow3.mod, x-netflow3.mod

Console — AMPL

```
ampl: model x-netflow3.mod;
ampl: data netflow3.dat;
ampl:
ampl: option solver x-gurobi;
ampl: solve;
x-Gurobi 9.5.2: Set parameter Username
                x-Gurobi 9.5.2: optimal solution;
ampl:
ampl: display Total_Cost;
Total_Cost = 5900

ampl: option display_eps .000001, display_1col 0;
ampl: display Flow;
Flow [Bands,*,*] (tr)
:          Denver Detroit    :=
Boston        0      50
'New York'   50       0
Seattle      10       0

 [Coils,*,*] (tr)
:          Denver Detroit    :=
Boston        0      40
'New York'   10      20
Seattle      30       0
;

ampl:
```

x-netflow3.mod | netflow3.dat

```
set PRODUCTS;
set NODES;

set ARCS within {NODES,NODES};
param capacity {ARCS} >= 0;

param inflow {PRODUCTS,NODES};
param min_ship >= 0;
param max_arcs >= 0;

param fix_cost {ARCS} >= 0;
param var_cost {PRODUCTS,ARCS} >= 0;

var Flow {PRODUCTS,ARCS} >= 0;

minimize Total_Cost:
  sum {p in PRODUCTS, (i,j) in ARCS} var_cost[p,i,j] * Flow[p,i,j] +
  sum {(i,j) in ARCS}
    if exists {p in PRODUCTS} Flow[p,i,j] > 0 then fix_cost[i,j];

subject to Shipment_Limits {(i,j) in ARCS}:
  sum {p in PRODUCTS} Flow[p,i,j] = 0 or
  min_ship <= sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j];

subject to Conservation {p in PRODUCTS, j in NODES}:
  sum {(i,j) in ARCS} Flow[p,i,j] + inflow[p,j] =
  sum {(j,i) in ARCS} Flow[p,j,i];

subject to Limit_Used:
  count {(i,j) in ARCS}
    (sum {p in PRODUCTS} Flow[p,i,j] > 0) <= max_arcs;
```

```
minimize total_fuelcost:
  sum{(i,j) in A} sum{k in V} X[i,j,k] *
    ((if H[i,k] <= 300 then dMor[i,j] else
      if H[i,k] <= 660 then dAft[i,j] else
      if H[i,k] <= 901 then dEve[i,j]) * 5 +
     (if H[i,k] <= 300 then tMor[i,j] else
      if H[i,k] <= 660 then tAft[i,j] else
      if H[i,k] <= 901 then tEve[i,j]) * 0.0504);
```

```
subj to NoPersonIsolated
         {l in TYPES['loc'], r in TYPES['rank'], j in 1..numberGrps}:
  sum {i in LOCRANK[l,r]} Assign[i,j] = 0 or
  sum {i in LOCRANK[l,r]} Assign[i,j] +
    sum {a in ADJACENT[r]} sum {i in LOCRANK[l,a]} Assign[i,j] >= 2;
```

Formulating

Supported Extensions and Solvers

Operators and functions

- Conditional: `if-then-else`; `==>`, `<==`, `<==>`
- Logical: `or`, `and`, `not`; `exists`, `forall`
- Piecewise linear: `abs`; `min`, `max`; `<<breakpoints; slopes>>`
- Counting: `count`; `atmost`, `atleast`, `exactly`; `numberof`
- Comparison: `>`, `<`, `!=`; `alldiff`
- Complementarity: `complements`
- Nonlinear: `*`, `/`, `^`; `exp`, `log`; `sin`, `cos`, `tan`; `sinh`, `cosh`, `tanh`
- Set membership: `in`

Solvers

- Gurobi, COPT, HiGHS, *Xpress coming soon, . . .*

Modeling guide

- *https://amplmp.readthedocs.io/en/latest/rst/model-guide.html*

Formulating

How It Works with MIP Solvers

Read problem instance from AMPL nl file

- Store initially as linear coefficients + expression trees
- Analyze trees to determine if linearizable

Generate linearizations

- Walk trees to build linearizations (flatten)
- Define auxiliary variables (usually zero-one)
- Generate equivalent constraints

Solve

- Send to solver through its API
- Convert optimal solution back to the original AMPL variables
- Write solution to AMPL

Formulating

Special Alternatives in *x-Gurobi*

Apply our linearization (`count`)

- Use Gurobi's linear API

Have Gurobi linearize (`or`, `abs`)

- Simplify and "flatten" the expression tree
- Use Gurobi's "general constraint" API
  - **`addGenConstrOr`** (resbinvar, [binvars])
    tells Gurobi: resbinvar = 1 iff at least one item in [binvars] = 1
  - **`addGenConstrAbs`** (resvar, argvar)
    tells Gurobi: resvar = |argvar|

Have Gurobi piecewise-linearize (`log`)

- Replace univariate nonlinear functions by p-l approximations
- Use Gurobi's "function constraint" API
  - **`addGenContstrLog`** (xvar, yvar)
    tells Gurobi: yvar = a piecewise-linear approximation of log(xvar)

Formulating

Implementation Issues

Is an expression repeated?

- Detect common subexpressions

```
subject to Shipment_Limits {(i,j) in ARCS}:

sum {p in PRODUCTS} Flow[p,i,j] = 0 or
min_ship <= sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j];
```

Is there a simplified formulation?

- Yes for min-max — no for max-max

```
minimize Max_Cost:
  max {i in PEOPLE} sum {j in PROJECTS} cost[i,j] * Assign[i,j];
```

```
maximize Max_Value:
  sum {t in T} max {n in N} weight[t,n] * Value[n];
```

Formulating

Implementation Issues *(cont'd)*

Does an exact linearization exist?

- Depends on constraint set: yes if "closed" — no if "open"
- *Usually* yes for >= or <= but no for > or < . . .

```
var Flow {ARCS} >= 0;
var Use {ARCS} binary;

subj to Use_Definition {(i,j) in ARCS}:
   Use[i,j] = 0 ==> Flow[i,j] = 0;
```

```
subj to Use_Definition {(i,j) in ARCS}:
   Flow[i,j] = 0 ==> Use[i,j] = 0 else Use[i,j] = 1;
```

Implementation Issues *(cont'd)*

Does an exact linearization exist?

- Depends on constraint set: yes if "closed" — no if "open"
- *Usually* yes for >= or <= but no for > or < . . .

```
var Flow {ARCS} >= 0;
var Use {ARCS} binary;

subj to Use_Definition {(i,j) in ARCS}:
   Use[i,j] = 0 ==> Flow[i,j] = 0 else Flow[i,j] >= 0;
```

```
subj to Use_Definition {(i,j) in ARCS}:
   Use[i,j] = 0 ==> Flow[i,j] = 0 else Flow[i,j] > 0;
```

Implementation Issues *(cont'd)*

Does an exact linearization exist?

- Depends on constraint set: yes if "closed" — no if "open"
- *Usually* yes for >= or <= but no for > or < . . .

```
var Flow {ARCS} >= 0;
var Use {ARCS} binary;

subj to Use_Definition {(i,j) in ARCS}:
   Use[i,j] = 0 ==> Flow[i,j] = 0 else Flow[i,j] >= 0;
```

```
subj to Use_Definition {(i,j) in ARCS}:
   Use[i,j] = 0 ==> Flow[i,j] = 0 else Flow[i,j] >= 1e-4;
```

Formulating

Solver Efficiency Issues

Bounds on subexpressions

❖ Define auxiliary variables that can be bounded

```
var x {1..2} <= 2, >= -2;

minimize Goldstein-Price:
  (1 + (x[1] + x[2] + 1)^2
    * (19 - 14*x[1] + 3*x[1]^2 - 14*x[2] + 6*x[1]*x[2] + 3*x[2]^2))
* (30 + (2*x[1] - 3*x[2])^2
    * (18 - 32*x[1] + 12*x[1]^2 + 48*x[2] - 36*x[1]*x[2] + 27*x[2]^2));
```

```
var t1 >= 0, <= 25;   subj to t1def: t1 = (x[1] + x[2] + 1)^2;
var t2 >= 0, <= 100;  subj to t2def: t2 = (2*x[1] - 3*x[2])^2;

minimize Goldstein-Price:
  (1 + t1
    * (19 - 14*x[1] + 3*x[1]^2 - 14*x[2] + 6*x[1]*x[2] + 3*x[2]^2))
* (30 + t2
    * (18 - 32*x[1] + 12*x[1]^2 + 48*x[2] - 36*x[1]*x[2] + 27*x[2]^2));
```

Solver Efficiency Issues *(cont'd)*

Simplification of logic

❖ Replace an iterated **exists** with a **sum**

```
minimize TotalCost: ...
  sum {(i,j) in ARCS}
    if exists {p in PRODUCTS} Flow[p,i,j] > 0 then fix_cost[i,j];
```

```
minimize TotalCost: ...
  sum {(i,j) in ARCS}
    if sum {p in PRODUCTS} Flow[p,i,j] > 0 then fix_cost[i,j];
```

Formulating

Solver Efficiency Issues *(cont'd)*

Creation of common subexpressions

- Substitute a stronger bound from a constraint

```
subject to Shipment_Limits {(i,j) in ARCS}:
   sum {p in PRODUCTS} Flow[p,i,j] = 0 or
   min_ship <= sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j];

minimize TotalCost: ...
   sum {(i,j) in ARCS}
      if sum {p in PRODUCTS} Flow[p,i,j] > 0
         then fix_cost[i,j];
```

```
minimize TotalCost: ...
   sum {(i,j) in ARCS}
      if sum {p in PRODUCTS} Flow[p,i,j] >= min_ship
         then fix_cost[i,j];
```

. . . consider automating all these improvements

Exchanging data and results with spreadsheets and database systems

Direct spreadsheet (.xlsx) file interface

- Works with all .xlsx files on Windows, Linux, macOS
- Supports "two-dimensional" spreadsheet tables

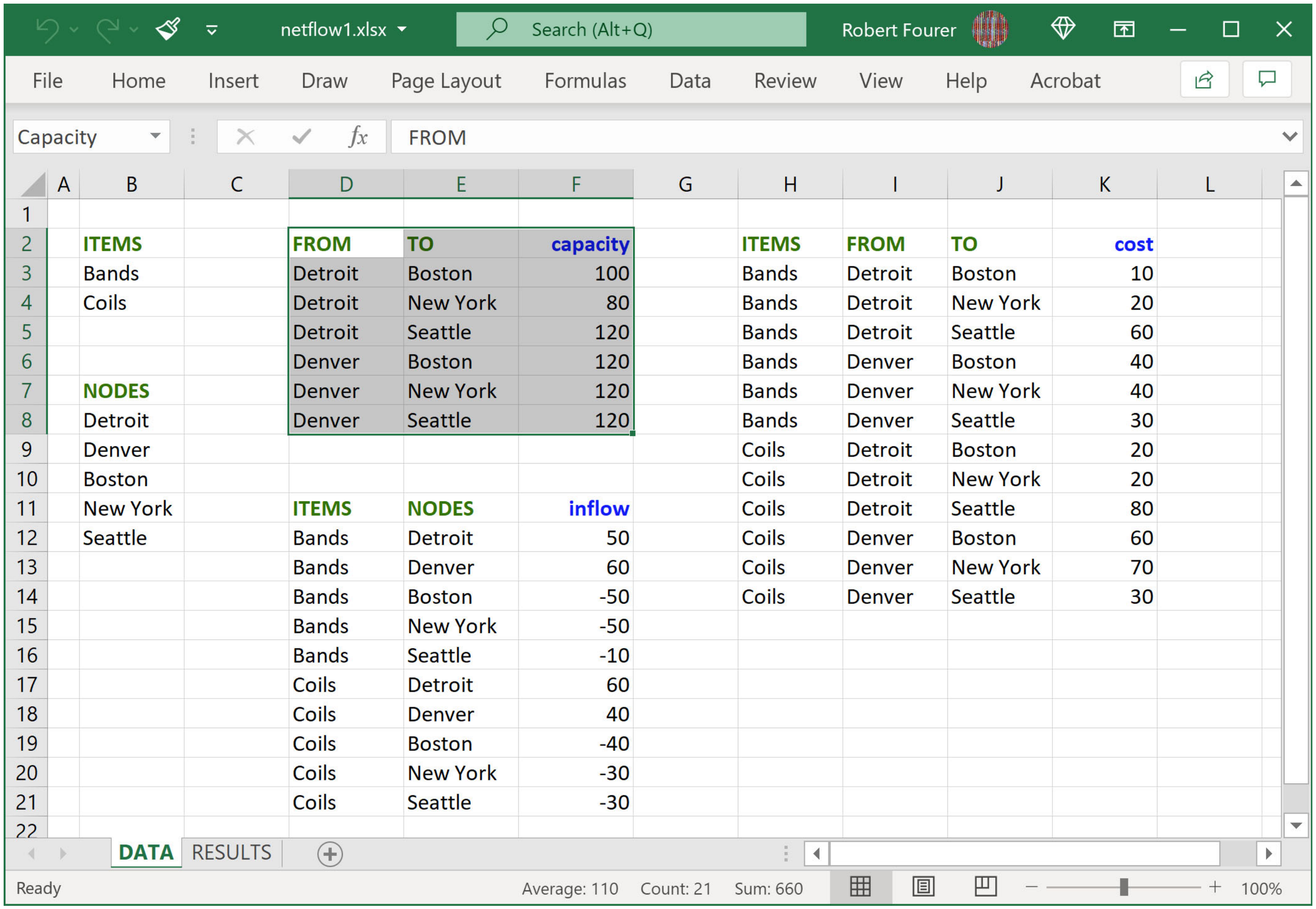
Direct comma-separated value (.csv) file interface

New ODBC interface for database systems

- Support for Microsoft ODBC, unixODBC, iODBC
- Faster operation
- Extended update features

Direct Spreadsheet Interface

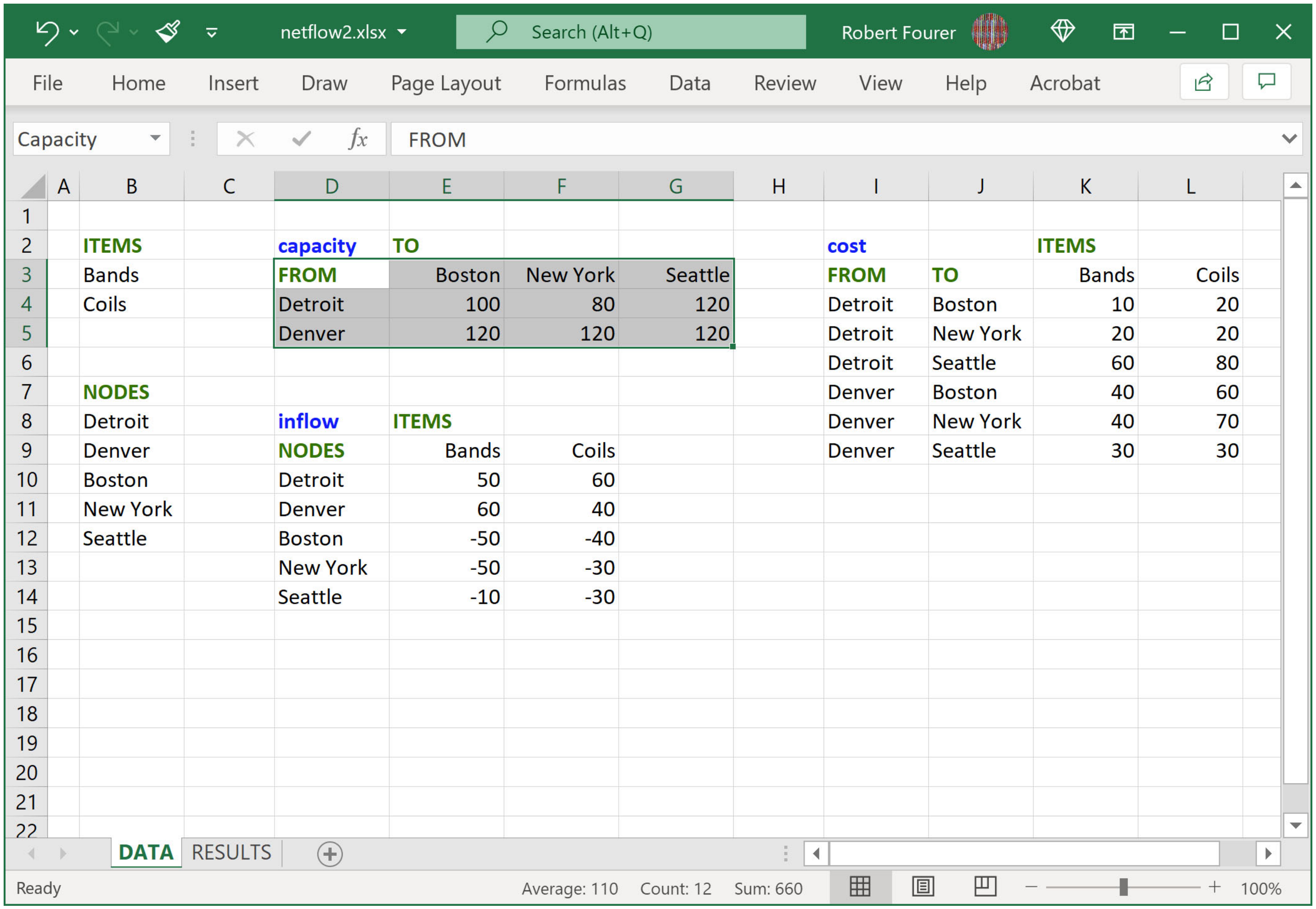
"1D" spreadsheet ranges



	B	D	E	F	H	I	J	K
1								
2	ITEMS	FROM	TO	capacity	ITEMS	FROM	TO	cost
3	Bands	Detroit	Boston	100	Bands	Detroit	Boston	10
4	Coils	Detroit	New York	80	Bands	Detroit	New York	20
5		Detroit	Seattle	120	Bands	Detroit	Seattle	60
6		Denver	Boston	120	Bands	Denver	Boston	40
7	NODES	Denver	New York	120	Bands	Denver	New York	40
8	Detroit	Denver	Seattle	120	Bands	Denver	Seattle	30
9	Denver				Coils	Detroit	Boston	20
10	Boston				Coils	Detroit	New York	20
11	New York	ITEMS	NODES	inflow	Coils	Detroit	Seattle	80
12	Seattle	Bands	Detroit	50	Coils	Denver	Boston	60
13		Bands	Denver	60	Coils	Denver	New York	70
14		Bands	Boston	-50	Coils	Denver	Seattle	30
15		Bands	New York	-50				
16		Bands	Seattle	-10				
17		Coils	Detroit	60				
18		Coils	Denver	40				
19		Coils	Boston	-40				
20		Coils	New York	-30				
21		Coils	Seattle	-30				
22								

Direct Spreadsheet Interface

"2D" spreadsheet ranges



Direct spreadsheet interface

Data Handling

Script (input)

```
model netflow1.mod;

table Products IN "amplxl" "netflow2.xlsx" "Items":
    PRODUCTS <- [ITEMS];

table Nodes IN "amplxl" "netflow2.xlsx":
    NODES <- [NODES];

table Capacity IN "amplxl" "netflow2.xlsx":
    ARCS <- [FROM,TO], capacity;

table Inflow IN "amplxl" "netflow2.xlsx":
    [ITEMS,NODES], inflow;

table Cost IN "amplxl" "netflow2.xlsx":
    [ITEMS,FROM,TO], cost;

load amplxl.dll;

read table Products; read table Nodes;
read table Capacity; read table Inflow; read table Cost;
```

Direct spreadsheet interface

Data Handling

Script (input)

```
model netflow1.mod;

table Products IN "amplxl" "netflow2.xlsx" "Items":
    PRODUCTS <- [ITEMS];

table Nodes IN "amplxl" "netflow2.xlsx":
    NODES <- [NODES];

table Capacity IN "amplxl" "netflow2.xlsx" "2D":
    ARCS <- [FROM,TO], capacity;

table Inflow IN "amplxl" "netflow2.xlsx" "2D":
    [ITEMS,NODES], inflow;

table Cost IN "amplxl" "netflow2.xlsx" "2D":
    [ITEMS,FROM,TO], cost;

load amplxl.dll;

read table Products; read table Nodes;
read table Capacity; read table Inflow; read table Cost;
```

Direct spreadsheet interface

Data Handling

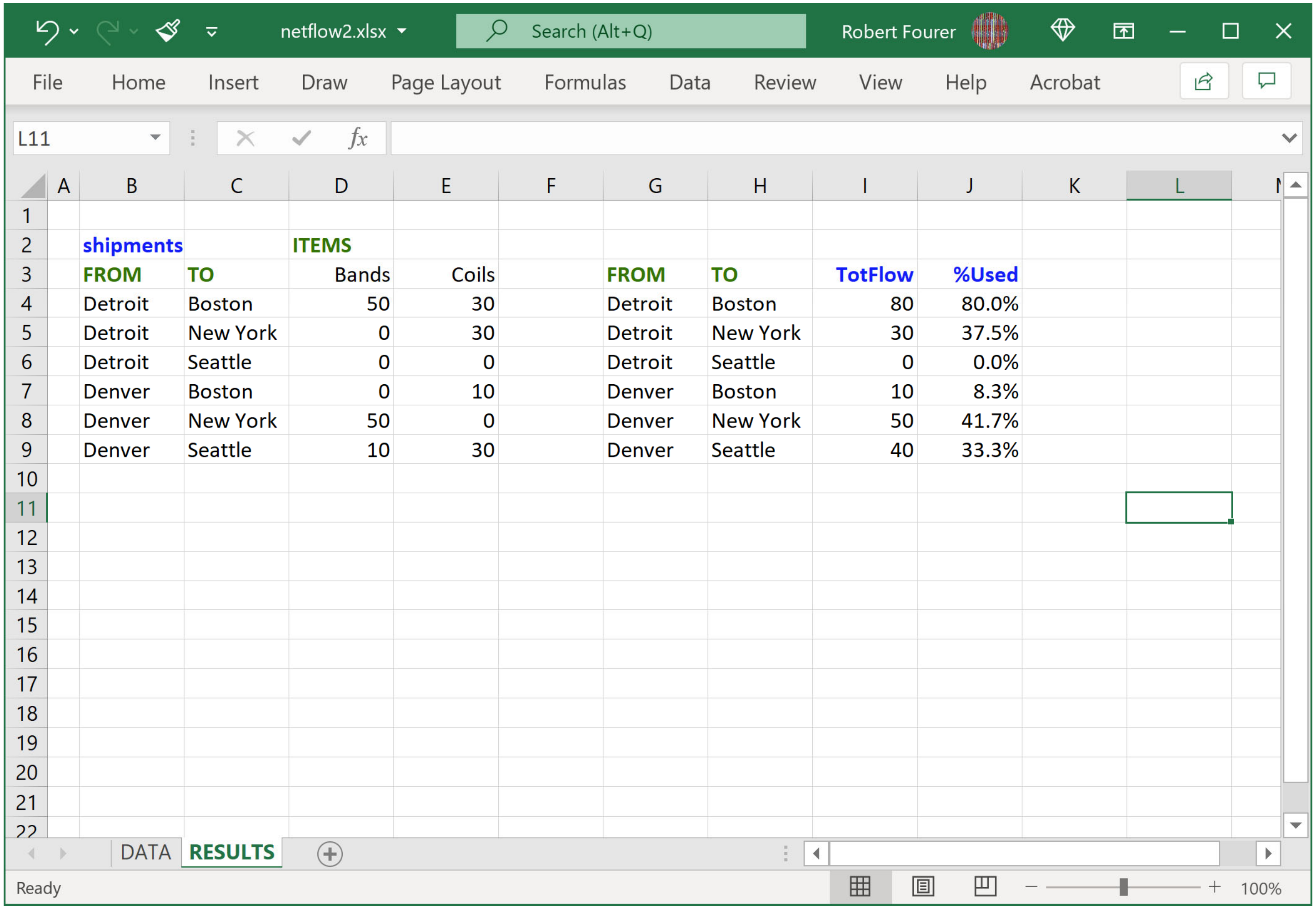
Script (output)

```
option solver gurobi;
solve;

table Results OUT "amplxl" "netflow1.xlsx" "2D":
    [ITEMS,FROM,TO], Flow;

table Summary OUT "amplxl" "netflow1.xlsx":
    {(i,j) in ARCS} -> [FROM,TO],
    sum {p in PRODUCTS} Flow[p,i,j] ~ TotFlow,
    sum {p in PRODUCTS} Flow[p,i,j] / capacity[i,j] ~ "%Used";

write table Results;
write table Summary;
```

Direct spreadsheet interface

Data Results

"2D" spreadsheet range



	A	B	C	D	E	F	G	H	I	J
1										
2		shipments		ITEMS						
3		FROM	TO	Bands	Coils		FROM	TO	TotFlow	%Used
4		Detroit	Boston	50	30		Detroit	Boston	80	80.0%
5		Detroit	New York	0	30		Detroit	New York	30	37.5%
6		Detroit	Seattle	0	0		Detroit	Seattle	0	0.0%
7		Denver	Boston	0	10		Denver	Boston	10	8.3%
8		Denver	New York	50	0		Denver	New York	50	41.7%
9		Denver	Seattle	10	30		Denver	Seattle	40	33.3%

New ODBC Interface

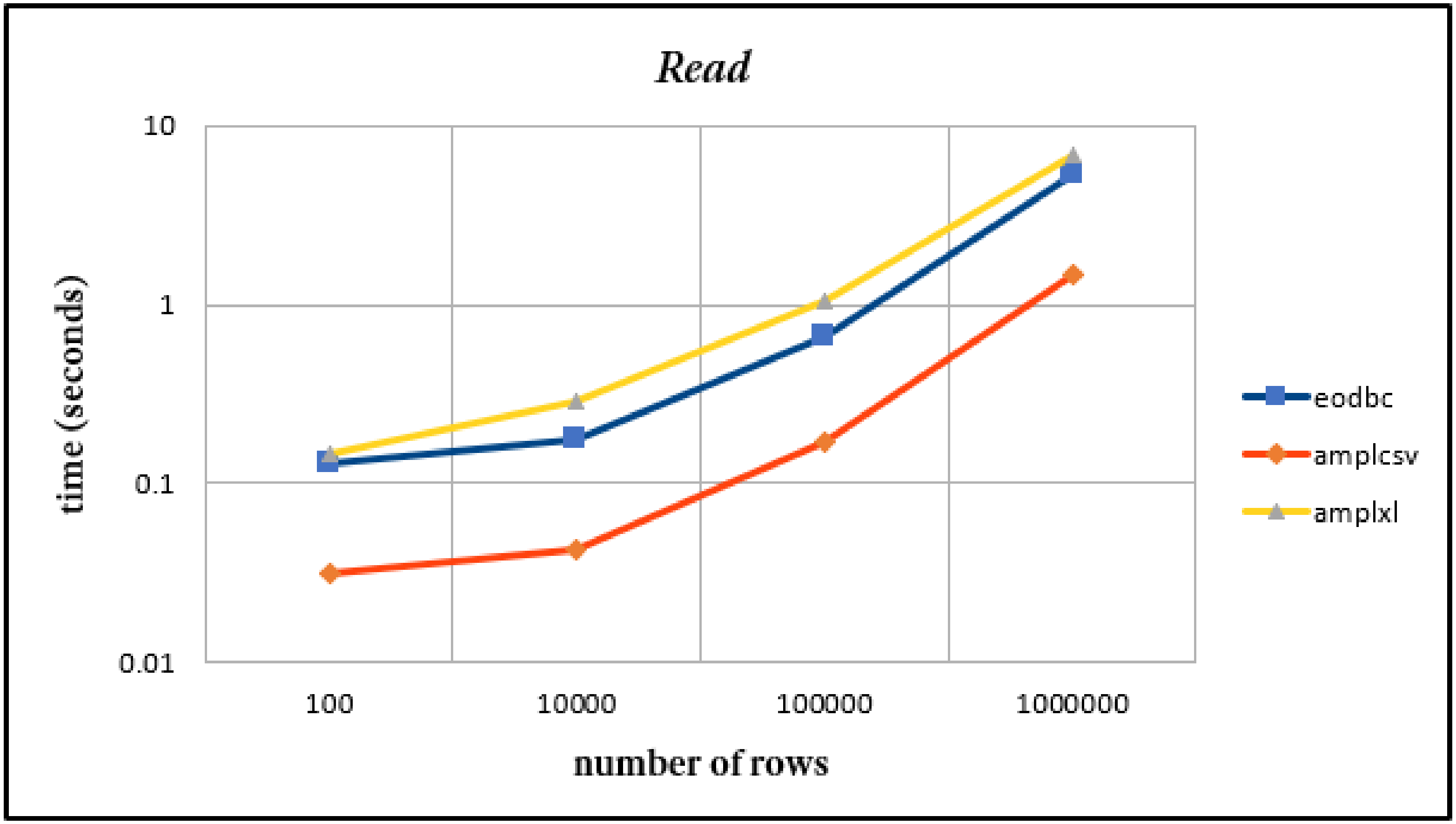
Open Database Connectivity

- Standard API for accessing database management systems
- Many database systems have ODBC drivers
- Supported by AMPL table statements
  - Can include SQL queries

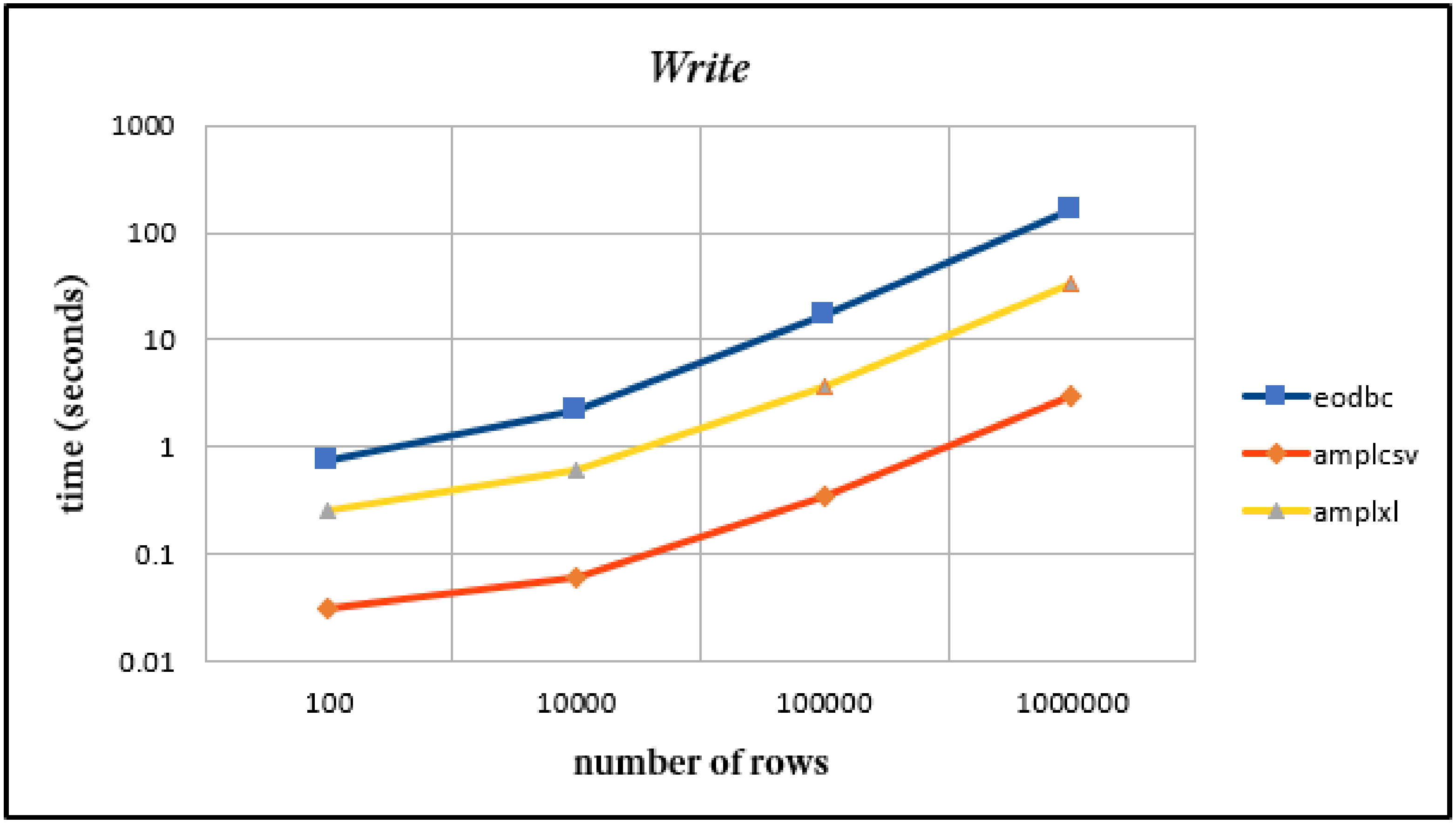
Enhancements

- Faster writes
- Table rewrite support
  - Preserve the column data types
- Table update support
  - Modify only selected records of a large table
- Table "upsert" support (experimental)
  - Update a record if it already exists, otherwise insert it
  - Requires a database-specific SQL statement

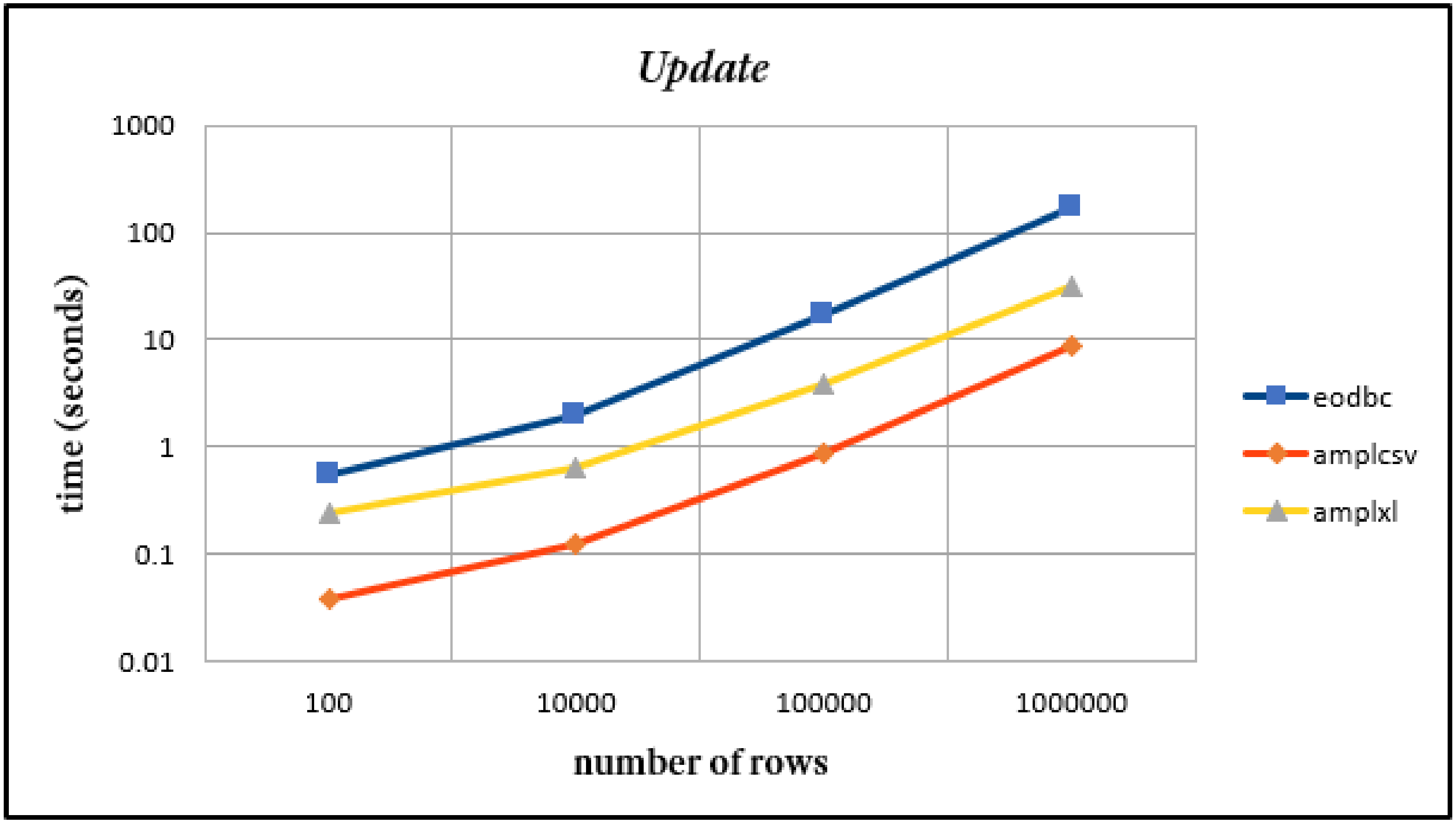
Reading Efficiency



Writing Efficiency



Updating Efficiency



Interfacing to Enterprise Systems through AMPL APIs

Embed AMPL in applications

- ❖ Program in C++, C#, Java, MATLAB, Python, R
- ❖ Call AMPL to do optimization, via
  API (application programming interface) library routines
  - ∗ Import data and export solutions
  - ∗ Read AMPL models & execute AMPL commands
  - ∗ Invoke solvers

Recent enhancements

- ❖ Snapshot utility
- ❖ Solver integration

Snapshot Utility

Save & restore an AMPL session

- Save key state information in a text file
  - Model declaration, data, current solution, options, . . .
- Restore the state as a starting point for later sessions

Valuable in API programming

- Debug API programs
  - Check correctness of model and data setup
  - Move to interactive environment for troubleshooting
- Avoid repeating expensive setups
  - Set up once and record a snapshot
  - Create many containers and restore the snapshot in each

Snapshot Command Listing

```
###snapshot-version: 0.1.1
###model-start
set PRODUCTS;
set NODES;
set ARCS within {NODES, NODES};
param capacity{ARCS}  >= 0;
param inflow{PRODUCTS, NODES};
param cost{PRODUCTS, ARCS}  >= 0;
var Flow{PRODUCTS, ARCS}  >= 0;
minimize TotalCost: sum{p in PRODUCTS, (i,j) in ARCS} cost[p,i,j
subject to Capacity{(i,j) in ARCS} : sum{p in PRODUCTS}
  Flow[p,I,j] <= capacity[i,j];
subject to Conservation{p in PRODUCTS, j in NODES} : sum{(i,j) i
  Flow[p,i,j] + inflow[p,j] == sum{(j,i) in ARCS} Flow[p,j,i];
###model-end

###options-start
option AMPLFUNC 'C:\Users\AMPL\Desktop\Solvers\ampltabl_64.dll';
option ampl_Include 'C:\Users\AMPL\Desktop\Analytics\Netflow';
option ampl_include 'C:\Users\AMPL\Desktop\Analytics\Netflow';
###options-end

###data-start
data;
set PRODUCTS :=
     'Pencils'
     'Pens';
set NODES :=
     'Detroit'
     'Denver'
     'Boston'
     'New York'
     'Seattle';
set ARCS :=
     ('Detroit','Boston')
     ('Detroit','New York')
     ('Detroit','Seattle')
     ('Denver','Boston')
     ('Denver','New York')
     ('Denver','Seattle');
param capacity :=
     ['Detroit','Boston'] 100
```

```
     ['Detroit','New York'] 80
     ['Detroit','Seattle'] 120
     ['Denver','Boston'] 120
     ['Denver','New York'] 120
     ['Denver','Seattle'] 120;
param cost :=
     ['Pencils','Detroit','Boston'] 10
     ['Pencils','Detroit','New York'] 20
     ['Pencils','Detroit','Seattle'] 60
     ['Pencils','Denver','Boston'] 40
     ['Pencils','Denver','New York'] 40
     ['Pencils','Denver','Seattle'] 30
     ['Pens','Detroit','Boston'] 20
     ['Pens','Detroit','New York'] 20
     ['Pens','Detroit','Seattle'] 80
     ['Pens','Denver','Boston'] 60
     ['Pens','Denver','New York'] 70
     ['Pens','Denver','Seattle'] 30;

model;
###data-end

###current-problem-start
problem Initial;
environ Initial;
###current-problem-end

###objectives-start
objective TotalCost;
###objectives-end

###fixes-start
unfix Flow;
###fixes-end

###drop-restore-start
restore Capacity;
restore Conservation;
###drop-restore-end

###solution-start
###solution-end
```

Snapshot Usage

For use with an API

```
snapshot = ampl.get_output('snapshot;')
```

```
ampl2.eval(snapshot)
```

For use in interactive debugging

```
snapshot = ampl.get_output('snapshot;')
print(snapshot, file=open('snapshot.run', 'w'))
```

```
include "snapshot.run";
```

API Solver Integration

New library works more directly with solvers

- Access all of a solver's capabilities,
  using its lowest level (usually C) API functionalities
- Work with a generic solver interface
  that encapsulates the most common solver functionalities
  - switch between solvers without re-programming

Handle solver callbacks

- Customize or enhance MIP solver processing
- Employ solver-specific or generic callback interfaces

User's guide

- *https://ampls.readthedocs.io/en/latest/general/introduction.html*

Deploying Optimization in Virtual Environments

Dynamic licensing

- ❖ Not a machine-locked license scheme
  - ∗ Doesn't rely on fixed machine characteristics
- ❖ Not a traditional floating license scheme
  - ∗ Doesn't rely on a customer's machine for validation

Container support

- ❖ Use any base image
- ❖ Load needed AMPL and solver modules
- ❖ Initialize dynamic licensing

Dynamic Licensing

Not locked to computer hardware

- ❖ License based on usage limits
- ❖ Install anywhere: local or cloud, short-term or permanent

Usage monitored in real time

- ❖ Dashboard viewable by user and by us
- ❖ Short-term use may exceed limits
- ❖ If use exceeds limits in longer term, customer is contacted

How is it used?

- ❖ Flexible floating/departmental licensing of local machines
- ❖ Licensing of cloud or cluster machines
  that don't have *static* hardware signatures
- ❖ Licensing for containers
- ❖ New *free* community edition

Operation

Short-term leases

- Last 5 minutes
- Automatically renewed
- Internet connection required

Renewal procedure

- Sends current license & usage data to our REST API endpoints
  - Hosted across multiple cloud providers (AWS, Azure, . . .)
- User details and usage data are logged
- New license file is returned

. . . lease remains active during renewal process

Tracking

Information reported in renewal request

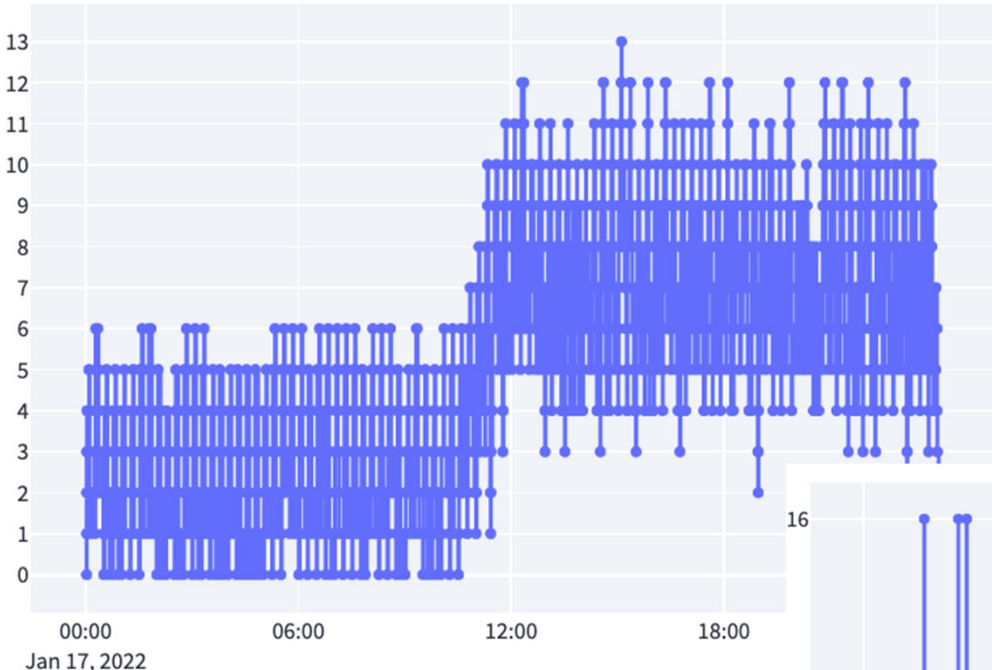
- CPU cores in the underlying machine
- CPU threads available

Information tracked

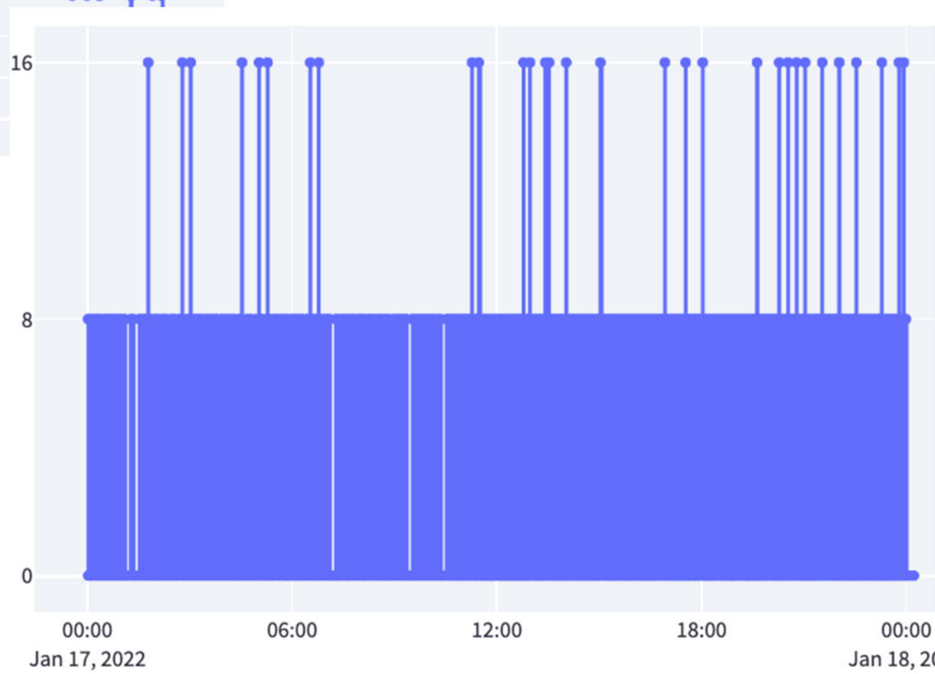
- Total concurrent active leases
- Total CPU threads available across all active leases
- Total different underlying machines with active leases
- Total CPU cores across all machines with active leases

Dynamic Licensing

Dashboard

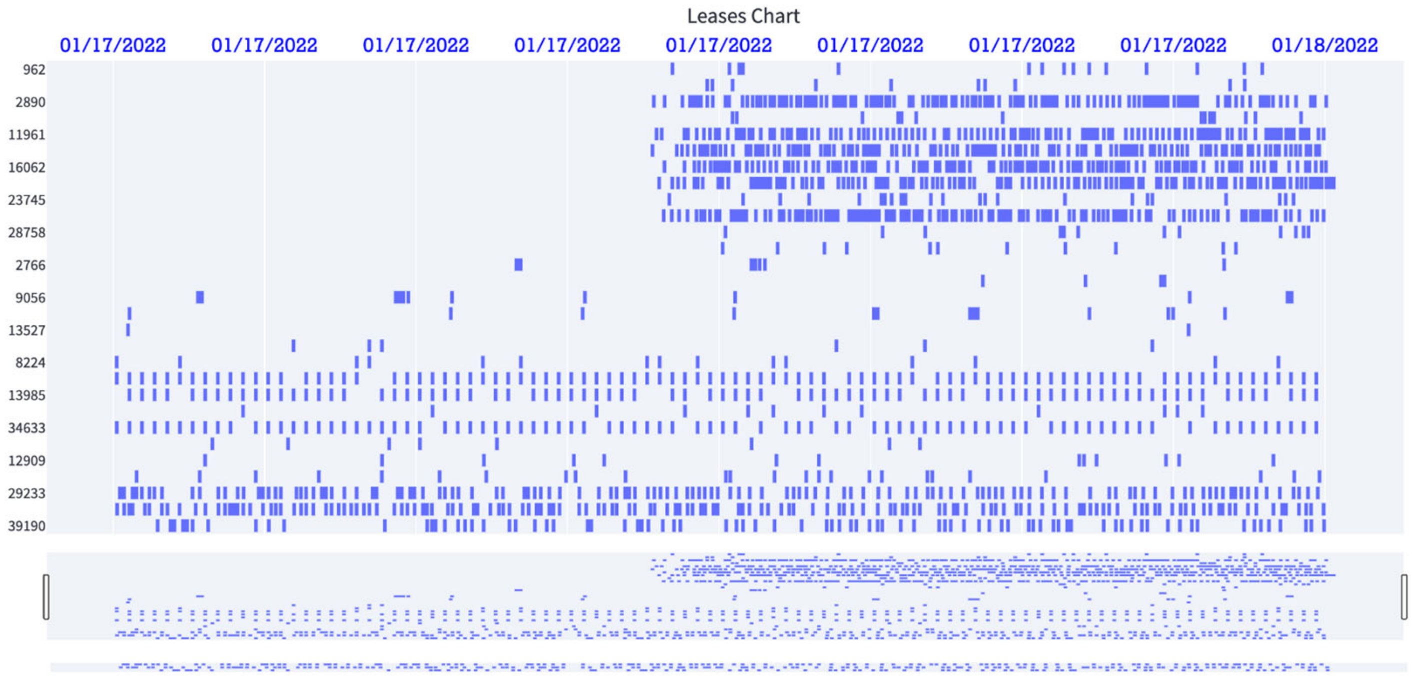


Total concurrent active leases



Total CPU cores across all machines with concurrent active leases

Dashboard



Leases aggregated by node ID

Container Setup *(in a dockerfile)*

Install curl, build arguments, get AMPL module

```
# Use any image as base image
FROM python:3.9-slim-buster

# Install curl in order to download the modules necessary
RUN apt-get update && apt-get install -y curl

# Build arguments
ARG LICENSE_UUID=f9758f88-b0a3-11eb-9e10-c75c7742e3ae
ARG MODULES_URL=https://ampl.com/dl/modules

# Download ampl-module.linux64.tgz
RUN cd /opt/ && curl -O ${MODULES_URL}/ampl-module.linux64.tgz && \
    tar xzvf ampl-module.linux64.tgz && rm ampl-module.linux64.tgz
```

Container Setup *(cont'd)*

Load Gurobi, COIN-OR solvers, license, amplpy API

```
# Download Gurobi solver
RUN cd /opt/ && curl -O ${MODULES_URL}/gurobi-module.linux64.tgz && \
    tar xzvf gurobi-module.linux64.tgz && rm gurobi-module.linux64.tgz

# Download COIN-OR solvers
RUN cd /opt/ && curl -O ${MODULES_URL}/coin-module.linux64.tgz && \
    tar xzvf coin-module.linux64.tgz && rm coin-module.linux64.tgz

# Download initial license file
RUN cd /opt/ampl.linux-intel64/ && curl -O \
    https://portal.ampl.com/download/license/${LICENSE_UUID}/ampl.lic

# Add installation directory to the environment variable PATH
ENV PATH="/opt/ampl.linux-intel64/:${PATH}"

# Install amplpy API
RUN pip3 install amplpy
```

New Ways to Try AMPL Now

Model colaboratory

NEOS Server Client (Kestrel)

Community Edition

AMPL Model Colaboratory

Run sample models free in Jupyter notebooks

- ❖ Run a short Python cell for setup
- ❖ Run AMPL cells for model, data, scripts

. . . execute in Google Colab, Kaggle, etc.

Great for getting started with AMPL

- ❖ No local downloads or installation needed
- ❖ Try out examples
  - ∗ From the AMPL book
  - ∗ From previous talks and papers
- ❖ Copy and modify an example

Index to examples

- ❖ *https://colab.ampl.com/*

NEOS Server Client

Use solvers on the free NEOS Server

- New *gokestrel* client works in any AMPL session
- Chosen solver is run by NEOS, returns results to AMPL

```
ampl: model multmip3.mod; data multmip3.dat;

ampl: option solver kestrel;
ampl: option kestrel_options 'solver=cplex';
ampl: option email '4er@ampl.com';
ampl: solve;

Connecting to: neos-server.org:3333
Job 12346358 submitted to NEOS, password='zBEfpQxD'
...
Executing on prod-exec-7.neos-server.org

CPLEX 20.1.0.0: threads=4
CPLEX 20.1.0.0: optimal integer solution; objective 235625
98 MIP simplex iterations
0 branch-and-bound nodes

ampl:
```

AMPL Community Edition

Open-ended trial

- Free AMPL and open-source solvers
  - no size or time limitations
- 30-day full-featured trials of commercial solvers

Dynamic licensing

- Runs on any computer with an internet connection

Immediate setup

- *https://ampl.com/ce*