

Python and AMPL :

Build Prescriptive Analytics applications quickly
with Pandas, Colab, Streamlit, and amplpy

Filipe Brandão, Robert Fourer

{filipe,4er@ampl.com}

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

INFORMS 2023, Phoenix, Arizona — October 15-18, 2023, CC-North 231B,
Technology Tutorial, Tuesday, October 17, 2:55 – 3:30 pm

Outline

Part I:

- Quick introduction to AMPL and new modeling functionalities

Part II (live demos):

- Quick introduction to amplpy (our Python API)
- AMPL on Google Colab
 - AMPL Model Colaboratory (<https://colab.ampl.com>)
 - New book: **Hands-On Optimization with AMPL in Python** (<https://mo-book.ampl.com>)
- AMPL and solvers as python packages
- AMPL on Streamlit Cloud
- How to deploy large-scale optimization applications with AMPL

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

Typical User Complaint

— — —
Thank you so much for replying.

Let me show my "if-then" constraint in a more clear way as follows:

```
set veh := {1..16 by 1};
```

```
param veh_ind {veh};  
param theory_time {veh};  
param UP := 400000;
```

```
var in_lane_veh {veh} integer >=1, <=2;  
var in_in_time {veh} >=0, <=UP;
```

*Note that "in_lane_veh {veh}" are integer variables which equal 1 or 2,
and "in_in_time {veh}" are continuous variables.*

```
subject to IfConstr {i in 1..card(veh)-1, j in i+1..card(veh):  
    veh_ind[i] = veh_ind[j] and theory_time[i] <= theory_time[j]}:
```

```
    in_lane_veh[i] = in_lane_veh[j] ==> in_in_time[j] >= in_in_time[i] + l_veh/V;
```

When I run my program, there appears the following statement:

CPLEX 20.1.0.0: logical constraint _slogcon[1] is not an indicator constraint.

Typical Reply

— — —

To reformulate this model in a way that your MIP solver would accept, you could define some more binary variables,

```
var in_lane_same {veh,veh} binary;
```

with the idea that $\text{in_lane_same}[i,j]$ should be 1 if and only if $\text{in_lane_veh}[i] = \text{in_lane_veh}[j]$.

Then the desired relation could be written as two constraints:

```
in_lane_veh[i] = in_lane_veh[j] ==> in_lane_same[i,j] = 1  
in_lane_same[i,j] = 1 ==> in_in_time[j] >= in_in_time[i] + l_veh/V;
```

The second one is an indicator constraint, but you would just need to replace the first one by equivalent linear constraints.

Given that in_lan_veh can only be either 1 or 2, those constraints could be

```
in_lane_same[i,j] >= 3 - in_lane_veh[i] - in_lane_veh[j]  
in_lane_same[i,j] >= in_lane_veh[i] + in_lane_veh[j] - 3
```

New Solver Interface Library (MP)

Design

- C++ library for building efficient, configurable solver drivers
- Support for features of current C interface library
- *Extensive toolset for problem recognition and transformation*

Motivation . . .





- AMPL has logical and “not linear” expressions
for *writing models the way you think of them*
- Old interfaces have very limited support for these
- New interfaces, built with MP,
allow these expressions to be used and combined freely

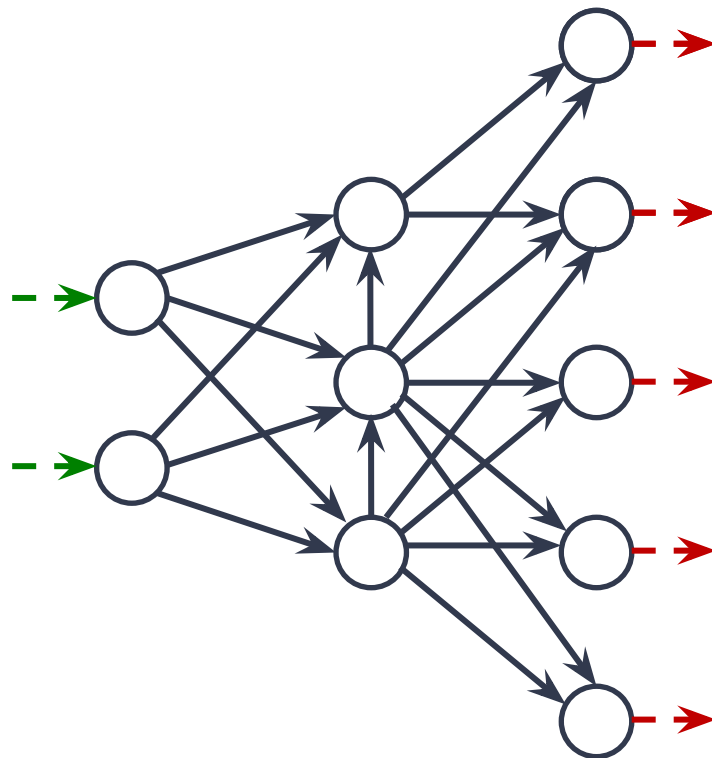
Example: Multi-Product Network Flow

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



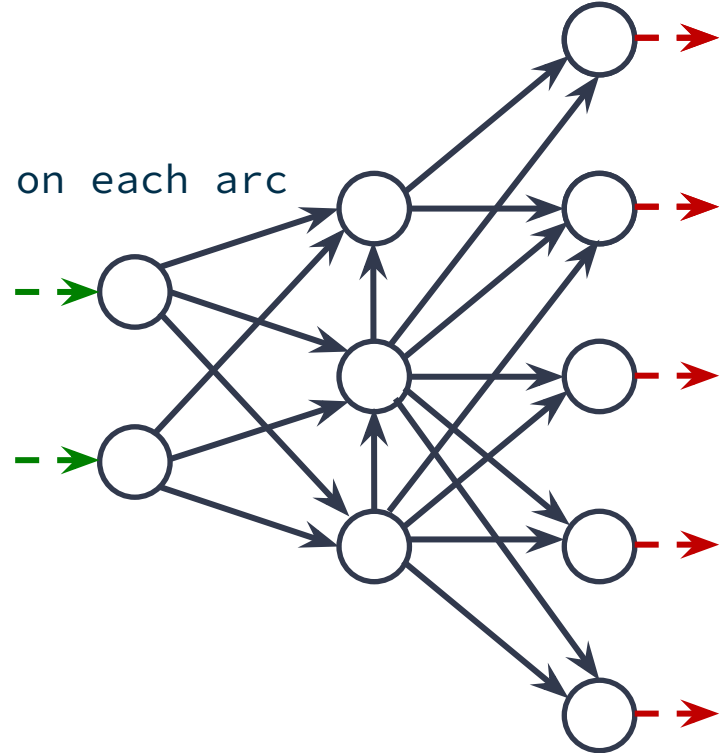
Example: Multi-Product Network Flow

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



AMPL Model for Multi-Product Network Flow

— — —

```
set PRODUCTS;
set NODES;
param net_inflow {PRODUCTS,NODES};

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

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

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

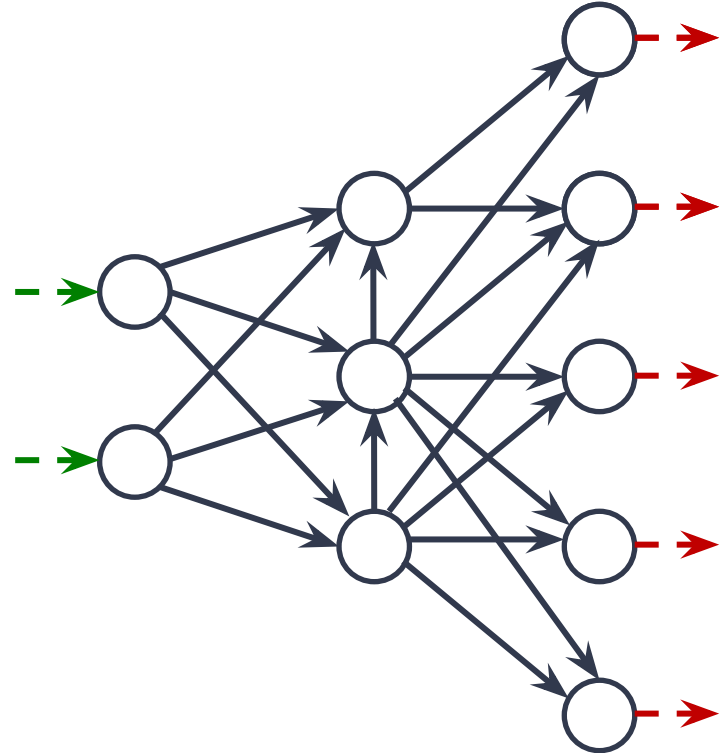
Example with conditions: Multi-Product Network Flow

Decide also

- whether to use each arc

So that

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



Positive Shipments Incur Fixed Costs

Linearization

```
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];
```

How you think about it

```
param fix_cost {ARCS} >= 0;  
  
minimize TotalCost:  
    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];
```

Shipments Can't Be Too Small

Linearization

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

How you think about it

```
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];
```

Can't Use Too Many Arcs

Linearization

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

How you think about it

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

Linearization is Usually Not That Easy!

— — —

```
subject to IfConstr {i in 1..card(veh)-1, j in i+1..card(veh):  
    veh_ind[i] = veh_ind[j] and theory_time[i] <= theory_time[j]}:  
    in_lane_veh[i] = in_lane_veh[j]  
    ==> in_in_time[j] >= in_in_time[i] + l_veh/V;
```

```
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);
```

```
subject 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;
```

Supported Extensions

— — —

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**

Expressions and constraints

- High-order polynomials
- Second-order and exponential cones (MOSEK driver!)

Supported Solvers

Solvers

- **Gurobi**, **Xpress**, **COPT**, **MOSEK**
- **HiGHS**, **CBC**, **SCIP**, **GCG**
- **CPLEX** soon

Modeling guide

- <https://mp.ampl.com/model-guide.html>

Examples using MP features

- <https://colab.ampl.com>
- <https://ampl.com/mo-book>

Quick introduction to amplpy! 

What do you need to know to use amplpy?

- Basic Python features (lists, dictionaries, etc.)
- Data manipulation with Pandas dataframes
- How to model in AMPL (or how to ask Chat GPT to write AMPL models for you!)

Example: Christmas model (<https://colab.ampl.com>)

The screenshot shows a Google Colaboratory notebook interface. The browser address bar displays `colab.research.google.com/github/ampl/amplcolab/blob/master/authors/fdabrandao/chatgpt/christmas.ipynb`. The notebook title is `christmas.ipynb`. The main content area is titled **Christmas model by ChatGPT** and contains the following text:

Problem description, mathematical model, AMPL model and data copied directly from [ChatGPT](#):

Can you generate a Christmas model in AMPL?

Certainly! Here is an example of a Christmas optimization model in AMPL:

```
# define the set of people
set PEOPLE;

# define the set of gifts
set GIFTS;

# define the cost of each gift
param cost{GIFTS};

# define the happiness level for each person receiving each gift
param happiness{PEOPLE, GIFTS};

# define the budget for gift-giving
param budget;

# define the decision variables:
# x[p,g] is 1 if person p receives gift g, 0 otherwise
var x{PEOPLE, GIFTS} binary;

# the objective is to maximize
```

Can you generate the description in mathjax to use in a colab notebook?

Certainly! Here is the description of the Christmas optimization model, formatted in MathJax for use in a Colab notebook:

In this optimization problem, we are trying to choose the best gift for each person, given a set of people 'PEOPLE' and a set of gifts 'GIFTS', in order to maximize the total happiness. Each gift has a certain cost, and we have a budget that we cannot exceed. Each person can only receive one gift. The decision variables 'x[p,g]' represent whether each person 'p' receives each gift 'g', and must be binary (either 0 or 1).

The mathematical formulation of the problem is as follows:

$$\begin{aligned} & \text{maximize} \quad \sum_{p \in \text{PEOPLE}} \sum_{g \in \text{GIFTS}} \text{happiness}_{\{p,g\}} \cdot x_{\{p,g\}} \\ & \text{subject to} \quad \sum_{g \in \text{GIFTS}} \text{cost}_{\{g\}} \cdot x_{\{p,g\}} \leq \text{budget} \quad \forall p \in \text{PEOPLE} \\ & \quad \sum_{p \in \text{PEOPLE}} x_{\{p,g\}} = 1 \quad \forall g \in \text{GIFTS} \\ & \quad x_{\{p,g\}} \in \{0,1\} \quad \forall p \in \text{PEOPLE}, g \in \text{GIFTS} \end{aligned}$$

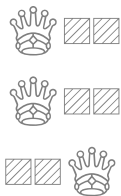
Example: N-Queens (<https://colab.ampl.com>)

— — —

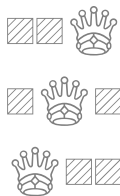
How can n queens be placed on an $n \times n$ chessboard so that no two of them attack each other?

Constraint **alldiff** enforces a set of integer variables to take distinct values. Using **alldiff**, we can model N-Queens as follows:

```
param n integer > 0; # N-queens
var Row {1..n} integer >= 1 <= n;
s.t. row_attacks: alldiff ({j in 1..n} Row[j]);
s.t. diag_attacks: alldiff ({j in 1..n} Row[j]+j);
s.t. rdiag_attacks: alldiff ({j in 1..n} Row[j]-j);
```



Row[1] == Row[2]
1 == 1



Row[1]+1 == Row[2]+2
3+1 == 2+2



Row[1]-1 == Row[2]-2
1-1 == 2-2

Example: N-Queens (<https://colab.ampl.com>)

The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL `colab.research.google.com/github/ampl/amplcolab/blob/master/authors/glebbelow/miscellaneous/nqueens.ipynb#scrollTo=jYDiBDD_7Kba`. The notebook title is `nqueens.ipynb`. The menu bar includes `File`, `Edit`, `View`, `Insert`, `Runtime`, `Tools`, and `Help`, with a status message `Cannot save changes`. The left sidebar shows a search bar and a file explorer with the notebook `nqueens.ipynb`. The main content area displays the notebook's metadata:

- N-Queens**
- Buttons: `github`, `Open in Colab`, `Open in Kaggle`, `Run on Gradient`, `Open`, `Studio Lab`, `hits 3 / 454`
- Description:** How can N queens be placed on an NxN chessboard so that no two of them attack each other?
- Tags:** `amplpy`, `constraint-programming`, `highlights`
- Notebook author:** Gleb Belov <gleb@ampl.com>

The notebook contains two code cells:

```
[1] # Install dependencies
!pip install -q amplpy
```

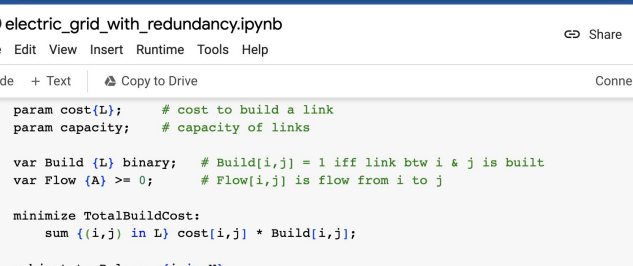
Progress bar: 5.6/5.6 MB 12.9 MB/s eta 0:00:00

```
# Google Colab & Kaggle integration
from amplpy import AMPL, tools
ampl = tools.ampl_notebook(
    modules=["highs"], # modules to install
    license_uuid="default") # license to use
```

A message at the bottom states: "Using default Community Edition License for Colab. Get yours at: <https://ampl.com/ce>. Licensed to AMPL Community Edition License for the AMPL Model Colaboratory (<https://colab.ampl.com>)."

The status bar at the bottom indicates the notebook is "2s completed at 11:53 AM".

Example: Network design with redundancy (<https://colab.ampl.com>)



```
[ ]
param cost{L};          # cost to build a link
param capacity;         # capacity of links

var Build {L} binary;    # Build[i,j] = 1 iff link btw i & j is built
var Flow {A} >= 0;       # Flow[i,j] is flow from i to j

minimize TotalBuildCost:
    sum {(i,j) in L} cost[i,j] * Build[i,j];

subject to Balance {i in N}:
    sum {(j,i) in A} Flow[j,i] - sum {(i,j) in A} Flow[i,j] >= demsup[i];

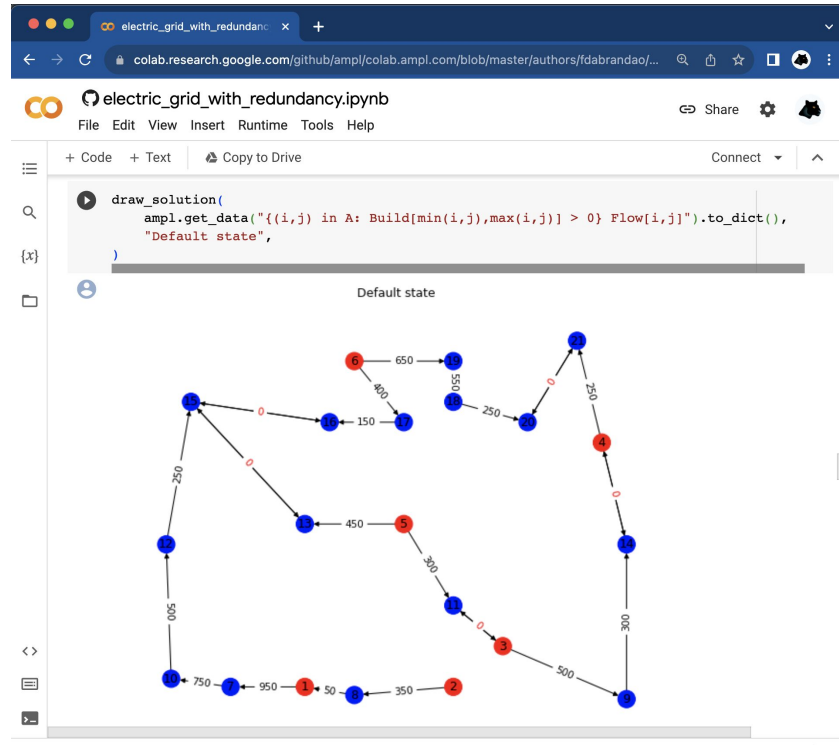
subject to ArcExists1 {(i,j) in L}:
    Flow[i,j] <= Build[i,j] * capacity;
subject to ArcExists2 {(i,j) in L}:
    Flow[j,i] <= Build[i,j] * capacity;

var FlRm {A,N} >= 0;     # FlRm[i,j,rn] is flow from i to j when node rn is removed

subject to RemoveNode {rm in N}:
    sum {(i,rm) in A} FlRm[i,rm,rm] + sum {(rm,j) in A} FlRm[rm,j,rm] = 0;

subject to BalanceRm {i in N, rm in N: i != rm}:
    sum {(j,i) in A} FlRm[j,i,rm] - sum {(i,j) in A} FlRm[i,j,rm] >= demsup[i];

subject to ArcExistsRm1 {(i,j) in L, rm in N}:
    FlRm[i,j,rm] <= Build[i,j] * capacity;
subject to ArcExistsRm2 {(i,j) in L, rm in N}:
    FlRm[j,i,rm] <= Build[i,j] * capacity;
```



Wait a minute. How are AMPL & solvers running on
Google Colab integrated with Python 🐍 ?

AMPL and all Solvers are now available as Python Packages

— — —

AMPL and all solvers are now available as python packages for **Windows**, **Linux (X86_64, aarch64, ppc64le)**, and **macOS**.

```
# Install Python API for AMPL
$ python -m pip install amplpy --upgrade

# Install solver modules (e.g., HiGHS, CBC, Gurobi)
$ python -m amplpy.modules install highs cbc gurobi

# Activate your license (e.g., free https://ampl.com/ce license)
$ python -m amplpy.modules activate <license-uuid>

# Import in Python
$ python
>>> from amplpy import AMPL
>>> ampl = AMPL() # instantiate AMPL object
```

> <https://ampl.com/python/>

AMPL is Free on Google Colab

— — —

> <https://dev.ampl.com/ampl/python/colab.html>

> <https://try.ampl.com> (quickly access to AMPL on Colab)

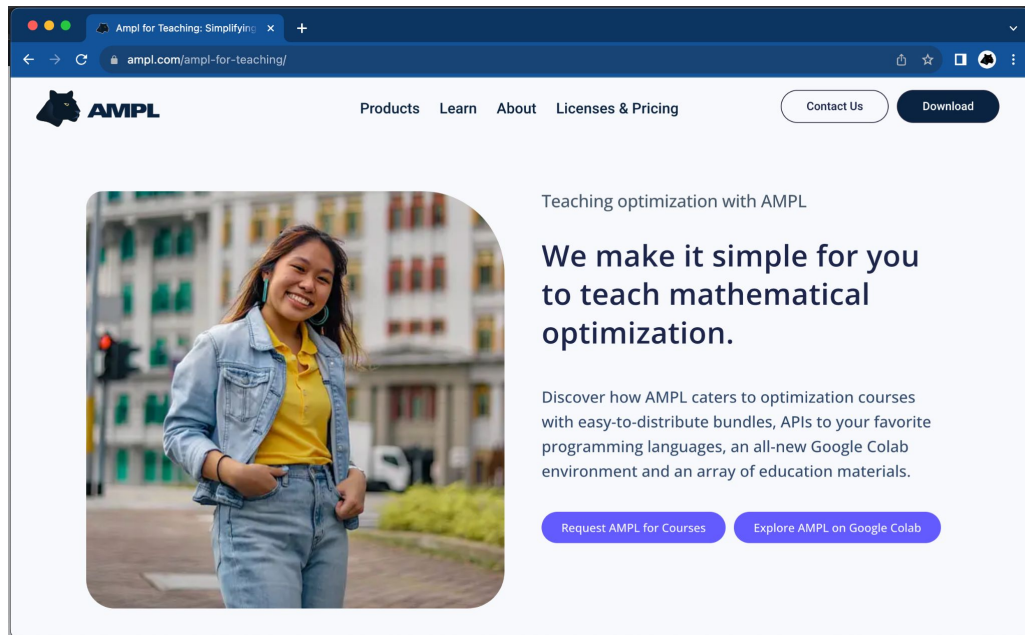
You can install AMPL on Google Colab (where it is free by default) as follows:


```
# Install dependencies  
%pip install -q amplpy
```

```
# Google Colab & Kaggle integration  
from amplpy import AMPL, tools  
ampl = tools.ampl_notebook(  
    modules=["gurobi", "coin", "highs", "gokestrel"], # modules to install  
    license_uid="default") # license to use
```

Free licenses to use on Google Colab (and locally!)

- — —
- **ampl.com/ce**
 - For personal use
 - Immediate access without approvals required
 - No size-limits
 - Includes access to:
 - Open-source solvers
 - Commercial solver trials
- **ampl.com/courses**
 - For teaching
 - No size-limits
 - **Full access to all solvers!**
 - All students can use the license during the course.



The Python-first  approach
to learn and model with AMPL!

AMPL Model Colaboratory (<https://colab.ampl.com>)

> Many examples: <https://colab.ampl.com> (live demo)

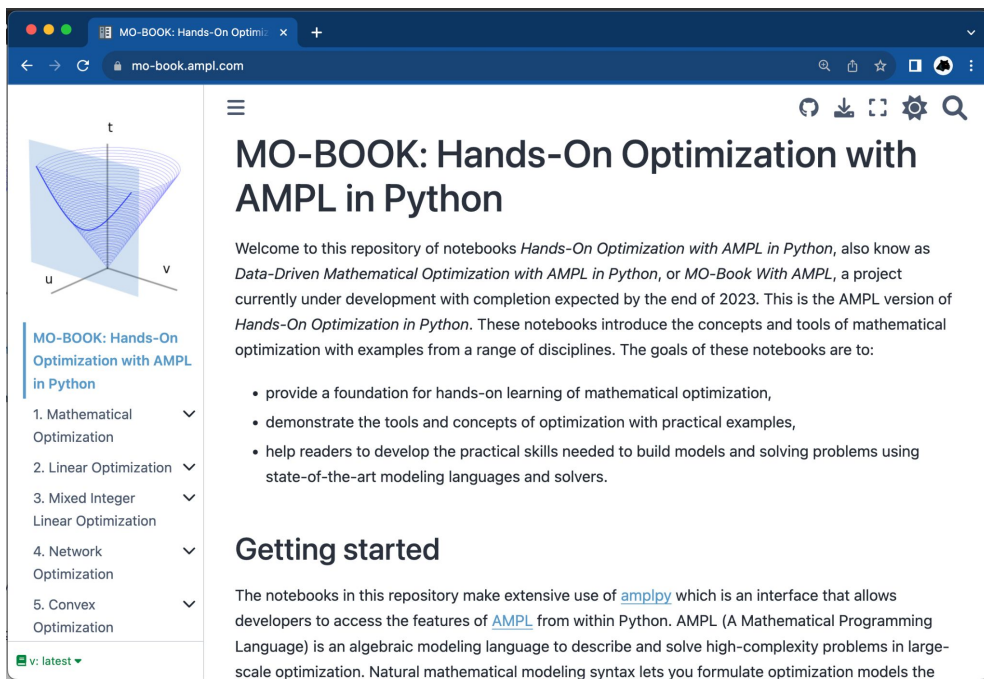
The screenshot displays the AMPL Model Colaboratory website interface. The top navigation bar includes links for Getting Started, Modules, Highlights, Lectures, Authors, Tags, and Notebooks. The main content area features a search bar and a section titled "AMPL Model Colaboratory" with a description: "AMPL Model Colaboratory is a collection of AMPL models in Jupyter Notebooks that run on platforms such as Google Colab, Kaggle, Gradient, and AWS SageMaker. [See our Highlights]". Below this, a Jupyter Notebook titled "nqueens.ipynb" is shown, displaying the following code:

```
1 # The only 3 lines you need to install and use AMPL
2 !pip install -q amplpy
3 from amplpy import tools
4 ampl = tools.ampl_notebook(modules=['highs', 'gurobi'])
```

The notebook interface also shows a "Cannot save changes" message and a "Copy to Drive" button. The left sidebar contains links to "AMPL's All-New Python Ecosystem", "New Python-AMPL Resource Hands-On Optimization with AMPL in Python", and "AMPL Community Edition". The bottom of the page mentions "You can use the Christmas notebook written..."

Data-Driven Mathematical Optimization with AMPL in Python

> New Book: <https://ampl.com/mo-book> (live demo)



The screenshot shows the homepage of the MO-BOOK repository. On the left, there is a 3D plot with axes labeled t, u, and v. Below it, the title "MO-BOOK: Hands-On Optimization with AMPL in Python" is displayed. A sidebar on the left lists the contents: 1. Mathematical Optimization, 2. Linear Optimization, 3. Mixed Integer Linear Optimization, 4. Network Optimization, and 5. Convex Optimization. The main text area welcomes visitors to the repository of notebooks, mentioning the project's completion by the end of 2023. It lists the goals of the notebooks: providing a foundation for learning, demonstrating optimization tools with practical examples, and helping readers develop practical skills for building models and solving problems using state-of-the-art modeling languages and solvers. A "Getting started" section is also visible, explaining that the notebooks use the `amplpy` interface to access the features of `AMPL` from within Python.

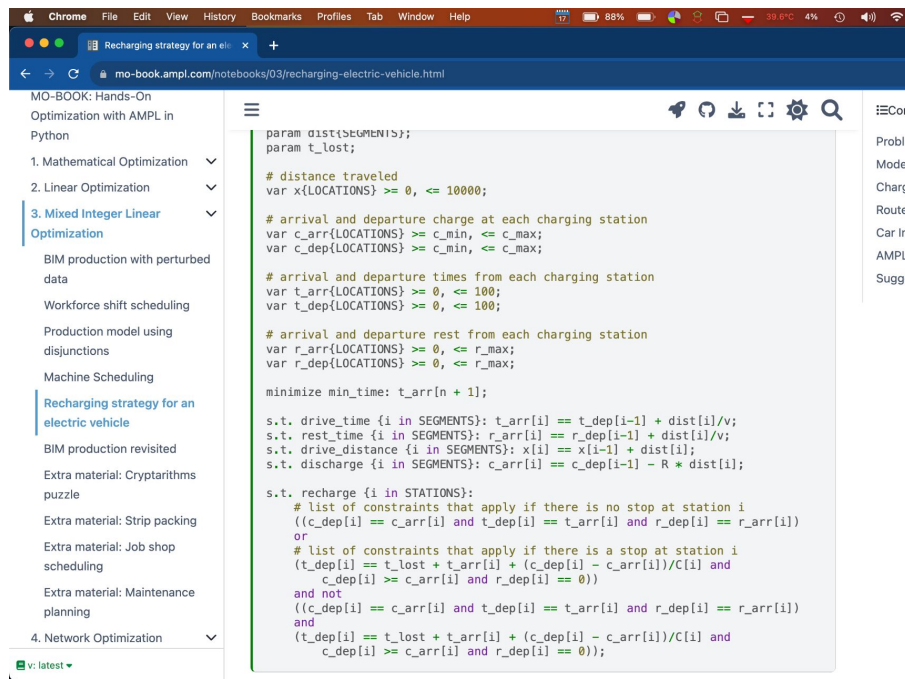
MO-BOOK: Hands-On Optimization with AMPL in Python

Welcome to this repository of notebooks *Hands-On Optimization with AMPL in Python*, also known as *Data-Driven Mathematical Optimization with AMPL in Python*, or *MO-Book With AMPL*, a project currently under development with completion expected by the end of 2023. This is the AMPL version of *Hands-On Optimization in Python*. These notebooks introduce the concepts and tools of mathematical optimization with examples from a range of disciplines. The goals of these notebooks are to:

- provide a foundation for hands-on learning of mathematical optimization,
- demonstrate the tools and concepts of optimization with practical examples,
- help readers to develop the practical skills needed to build models and solving problems using state-of-the-art modeling languages and solvers.

Getting started

The notebooks in this repository make extensive use of `amplpy` which is an interface that allows developers to access the features of `AMPL` from within Python. `AMPL` (A Mathematical Programming Language) is an algebraic modeling language to describe and solve high-complexity problems in large-scale optimization. Natural mathematical modeling syntax lets you formulate optimization models the



The screenshot shows a specific notebook titled "Recharging strategy for an electric vehicle". The notebook content is displayed in a code editor. It defines parameters for distance, arrival and departure charges, and arrival and departure times. It then sets up a minimization problem for the total time, subject to constraints on driving time, rest time, and discharge. The constraints are defined for each segment and station.

```
param dist{SEGMENTS};
param t_lost;

# distance traveled
var x{LOCATIONS} >= 0, <= 10000;

# arrival and departure charge at each charging station
var c_arr{LOCATIONS} >= c_min, <= c_max;
var c_dep{LOCATIONS} >= c_min, <= c_max;

# arrival and departure times from each charging station
var t_arr{LOCATIONS} >= 0, <= 100;
var t_dep{LOCATIONS} >= 0, <= 100;

# arrival and departure rest from each charging station
var r_arr{LOCATIONS} >= 0, <= r_max;
var r_dep{LOCATIONS} >= 0, <= r_max;

minimize min_time: t_arr[n + 1];

s.t. drive_time {i in SEGMENTS}: t_arr[i] == t_dep[i-1] + dist[i]/v;
s.t. rest_time {i in SEGMENTS}: r_arr[i] == r_dep[i-1] + dist[i]/v;
s.t. drive_distance {i in SEGMENTS}: x[i] == x[i-1] + dist[i];
s.t. discharge {i in SEGMENTS}: c_arr[i] == c_dep[i-1] - R * dist[i];

s.t. recharge {i in STATIONS}:
    # list of constraints that apply if there is no stop at station i
    ((c_dep[i] == c_arr[i] and t_dep[i] == t_arr[i] and r_dep[i] == r_arr[i])
    or
    # list of constraints that apply if there is a stop at station i
    (t_dep[i] == t_lost + t_arr[i] + (c_dep[i] - c_arr[i])/C[i] and
    c_dep[i] >= c_arr[i] and r_dep[i] == 0))
    and not
    ((c_dep[i] == c_arr[i] and t_dep[i] == t_arr[i] and r_dep[i] == r_arr[i])
    and
    (t_dep[i] == t_lost + t_arr[i] + (c_dep[i] - c_arr[i])/C[i] and
    c_dep[i] >= c_arr[i] and r_dep[i] == 0));
```

Deploying optimization applications quickly and easily using AMPL with Python

AMPL on Streamlit

> <https://ampl.com/streamlit> (live demo)

The screenshot shows the AMPL Streamlit app interface. The left sidebar contains a 'streamlit app' section with links to 'Modeling Tips', 'N-Queens', and 'Risk Return'. The main content area features the AMPL logo and the text 'AMPL's All-New Python Ecosystem' with links to 'ampl.com/python' and 'colab.ampl.com'. Below this is the 'N-Queens' section, which includes a description of the problem: 'How can n queens be placed on an $n \times n$ chessboard so that no two of them attack each other?'. It also provides a constraint definition using AMPL syntax:
`param n integer > 0; # N-queens
var Row {1..n} integer = 1..n;
s.t. row_attacks: alldiff {i in 1..n} Row[i];
s.t. diag_attacks: alldiff {i in 1..n} Row[i]+i;
s.t. rdiag_attacks: alldiff {i in 1..n} Row[i]-i;`
At the bottom, there is a slider for 'How many queens?' with a value of 8 and a range from 2 to 25.

The screenshot shows the AMPL Streamlit app interface for the 'Risk Return' model. The left sidebar contains a 'streamlit app' section with links to 'Modeling Tips', 'N-Queens', and 'Risk Return'. The main content area features the AMPL logo and the text 'AMPL's All-New Python Ecosystem' with links to 'ampl.com/python' and 'colab.ampl.com'. Below this is the 'Risk Return' section, which includes a description of the problem: 'How can n queens be placed on an $n \times n$ chessboard so that no two of them attack each other?'. It also provides a constraint definition using AMPL syntax:
`param n integer > 0; # N-queens
var Row {1..n} integer = 1..n;
s.t. row_attacks: alldiff {i in 1..n} Row[i];
s.t. diag_attacks: alldiff {i in 1..n} Row[i]+i;
s.t. rdiag_attacks: alldiff {i in 1..n} Row[i]-i;`
At the bottom, there is a slider for 'How many queens?' with a value of 8 and a range from 2 to 25.

Average return: -0.2%

Note: It is hard to predict expected returns!

Let's Optimize!

Pick the model

Minimize volatility

Pick the risk method

exp_cov

Pick the return method

capm_return

Pick the solver

gurobi

Minimize Volatility

Deploy anywhere with Docker

— — —

> <https://dev.ampl.com/ampl/docker/>

AMPL can be easily used on Docker containers and deployed anywhere.

```
# Use any image as base image with python installed
```

```
FROM python:3.9-slim-bullseye
```

```
# Install amplpy and all necessary amplpy.modules:
```

```
RUN python -m pip install amplpy --no-cache-dir # Install amplpy
```

```
RUN python -m amplpy.modules install highs gurobi --no-cache-dir # Install modules
```


Example project showing how to deploy applications

— — —

> <https://amplpyfinance.ampl.com/>

- How to use AMPL with Docker Containers:
 - A basic Docker Compose template for orchestrating a **Flask** application & a **Celery** queue with **Redis**.
 - <https://github.com/ampl/amplpyfinance/tree/master/deployment/docker>
- The same Docker images can be deployed to **Kubernetes Clusters**
- How to use AMPL in Continuous Integration Systems
 - This project uses **Azure Pipelines** and **GitHub Actions** for **CI/CD**
 - <https://dev.ampl.com/ampl/cicd/>

Continuous Integration Systems

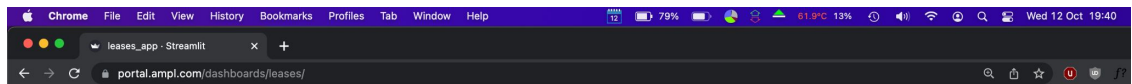
— — —

- How to use AMPL in Continuous Integration Systems
 - This project uses **Azure Pipelines** and **GitHub Actions** for **CI/CD**
 - <https://dev.ampl.com/ampl/cicd/>

```
jobs:
  Test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: ["3.10"]
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python ${ matrix.python-version }
        uses: actions/setup-python@v4
        with:
          python-version: ${ matrix.python-version }
      - name: Install dependencies
        run: |
          set -ex
          python -m pip install -r requirements.txt
          python -m pip install amplpy
          python -m amplpy.modules install <solver1> <solver2>
          python -m amplpy.activate <license-uuid>
      - name: Install package
        run: |
          python -m pip install .
      - name: Test package
        run: |
          python -m <package-name>.tests
```

What about licenses for AMPL and Commercial Solvers?

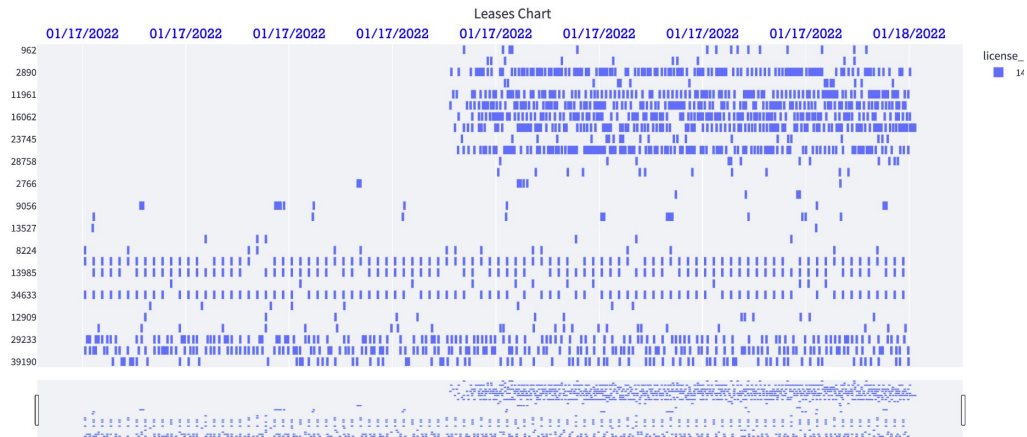
Dynamic Licensing System



Leases Chart

Aggregate by:

Node ID



Free licenses to use on Google Colab (and locally!)

- — —
- **ampl.com/ce**
 - For personal use
 - **Immediate access without approvals required!**
 - No size-limits
 - Includes access to:
 - Open-source solvers
 - Commercial solver trials
- **ampl.com/courses**
 - For teaching
 - No size-limits
 - **Full access to all solvers!**
 - All students can use the license during the course.

