Optimization in Your Toolchain: How AMPL is Making it Faster and Easier

Robert Fourer
4er@ampl.com
AMPL Optimization Inc.
www.ampl.com — +1 773-336-AMPL

INFORMS Business Analytics Conference
Houston, Texas — 3-5 April 2022
Technology Tutorials Track
Optimization has been fundamental to 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 enterprise systems. In this presentation, you’ll learn how AMPL’s modeling framework has been enhanced to support optimization in today’s challenging applications. Topics include:

- Expressing constraint logic more directly and understandably
- Exchanging data and results directly and efficiently with spreadsheets and database systems
- Interfacing to business systems through APIs for popular programming languages
- Deploying optimization in cloud environments and containers

To round out the presentation, the theme of adding optimization to applications will be illustrated by several varied case studies.
Outline

**Optimization**
- Optimization in practice
- Model-based optimization
- Model-based optimization in AMPL

**Developments in 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
Optimization in Practice

*Given a recurring need to make many interrelated decisions*
  - Purchases, production and shipment amounts, assignments, . . .

*Consistently make highly desirable choices*

*By applying ideas from mathematical optimization*
  - Ways of describing problems (*models*)
  - Ways of solving problems (*algorithms*)
Optimization in Practice

Large numbers of decision variables
- Thousands to millions

An objective function
- Minimize or maximize

Various constraint types
- 10-20 distinct types
- Thousands to millions of each type
- Few variables involved in each constraint

Solved many times with different data
- Simple rules can’t capture all possibilities in advance

Solvable only by a series of computational steps
- Number of steps not predictable
**Example:**
Multi-Product Network Flow

**Motivation**

- Ship products efficiently to meet demands

**Context**

- a transportation network
  - nodes \(\bigcirc\) representing cities
  - arcs \(\rightarrow\) representing roads
- supplies \(\rightarrow\) at nodes
- demands \(\rightarrow\) 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 \textit{with complications}:
Multi-Product Network Flow

\textit{Decide also}
\begin{itemize}
\item whether to use each arc
\end{itemize}

\textit{So that}
\begin{itemize}
\item variable plus fixed shipping costs are kept low
\item shipments are not too small
\item not too many arcs are used
\end{itemize}
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
**Multi-Product Flow**

**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

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

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

\[ \sum_{p \in P} X_{p ij} \geq m Y_{ij}, \quad \text{for all } (i,j) \in A \]

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

\[ \sum_{(i,j) \in A} X_{p ij} + s_{pj} = \sum_{(j,i) \in A} X_{p ji}, \quad \text{for all } p \in P, j \in N \]

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

```AMPL
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;

\[
\sum_{p \in P} X_{pij} \leq u_{ij} Y_{ij}, \text{ for all } (i,j) \in A
\]
Multi-Product Flow

Data Instance in AMPL Text Format

Limits

```AMPL
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**

```AMPL
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

\[
\text{sum } \{(i,j) \text{ in ARCS}\} \text{ fix\_cost}[i,j] \ast \text{Use}[i,j];
\]

Natural formulation

\[
\text{sum } \{(i,j) \text{ in ARCS}\}
\quad \text{if exists } \{p \text{ in PRODUCTS}\} \text{ Flow}[p,i,j] > 0 \text{ then fix\_cost}[i,j]
\]
Formulating

Shipments Can’t Be Too Small

Linearized formulation

\[
\begin{align*}
\sum_{p \in \text{PRODUCTS}} \text{Flow}[p,i,j] & \geq \text{min}_\text{ship} \times \text{Use}[i,j]; \\
\sum_{p \in \text{PRODUCTS}} \text{Flow}[p,i,j] & \leq \text{capacity}[i,j] \times \text{Use}[i,j];
\end{align*}
\]

Natural formulation

\[
\begin{align*}
\sum_{p \in \text{PRODUCTS}} \text{Flow}[p,i,j] & = 0 \text{ or } \\
\text{min}_\text{ship} & \leq \sum_{p \in \text{PRODUCTS}} \text{Flow}[p,i,j] \leq \text{capacity}[i,j]
\end{align*}
\]
Can’t Use Too Many Arcs

**Linearized formulation**

\[
\text{sum } \{(i,j) \text{ in ARCS}\} \text{ } \text{Use}[i,j] \leq \text{max_arcs};
\]

**Natural formulation**

\[
\text{count } \{(i,j) \text{ in ARCS}\} \left( \text{sum } \{p \text{ in PRODUCTS}\} \text{ Flow}[p,i,j] > 0 \right) \leq \text{max_arcs};
\]
Formulating Optimization by Same MIP Solver \((x\text{-}gurobi)\)
Formulating

Supported Model Expressions

Arithmetic operators

- + – * / ^
- iterated sum, prod

Logical operators

- if-then, if-then-else; ==> (implies)
- or, and, not
- iterated exists, forall, count

Almost-linear functions

- << . . . ; . . . >> (piecewise-linear)
- min, max, abs

Univariate nonlinear functions

- sin, cos, tan; sinh, cosh, tanh; asin, acos, . . .
- log, log10
Formulating

Supported Model Expressions (cont’d)

Relational constraints

- =, >=, <=
- double >=, <=

Logical constraints

- <, >, !=
- atleast, atmost, exactly
- alldiff
- complements

Variable domains

- >=, <=
- integer, binary
- in set
Formulating

Interface to MIP Solvers

Read objectives & constraints from AMPL

- Store initially as expression trees
  - (except for linear ones)
- Analyze to determine which are linearizable

Generate linearizations

- Walk trees to build linearizations (flatten)
- Define auxiliary variables (often 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
Apply our linearization (count)

- Use Gurobi’s linear API

Have Gurobi do simple linearizations (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)\text{ in ARCS}\}:

\[
\sum_{p \text{ in PRODUCTS}} \text{Flow}[p,i,j] = 0 \text{ or } \min\text{ship} \leq \sum_{p \text{ in PRODUCTS}} \text{Flow}[p,i,j] \leq \text{capacity}[i,j];
\]

Is there a simplified formulation?

- Yes for min-max, no for max-min

\[
\begin{align*}
\text{minimize Max_Cost:} \\
& \max_{i \text{ in PEOPLE}} \sum_{j \text{ in PROJECTS}} \text{cost}[i,j] \times \text{Assign}[i,j];
\end{align*}
\]

\[
\begin{align*}
\text{maximize Max_Value:} \\
& \sum_{t \text{ in T}} \max_{n \text{ in N}} \text{weight}[t,n] \times \text{Value}[n];
\end{align*}
\]
Formulating

Implementation Issues (cont’d)

Does an exact linearization exist?

- Yes if constraint set is “closed”
- No if constraint set is “open”

```plaintext
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?**

- Yes if constraint set is “closed”
- No if constraint set is “open”

```plaintext
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;
```

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ITEMS</td>
<td>FROM</td>
<td>TO</td>
<td>capacity</td>
<td>ITEMS</td>
<td>FROM</td>
<td>TO</td>
<td>cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Bands</td>
<td>Detroit</td>
<td>Boston</td>
<td>100</td>
<td>Bands</td>
<td>Detroit</td>
<td>Boston</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Coils</td>
<td>Detroit</td>
<td>New York</td>
<td>80</td>
<td>Bands</td>
<td>Detroit</td>
<td>New York</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Detroit</td>
<td>Seattle</td>
<td>120</td>
<td>Bands</td>
<td>Detroit</td>
<td>Seattle</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Denver</td>
<td>Boston</td>
<td>120</td>
<td>Bands</td>
<td>Denver</td>
<td>Boston</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>New York</td>
<td>Denver</td>
<td>New York</td>
<td>120</td>
<td>Bands</td>
<td>Denver</td>
<td>New York</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Detroit</td>
<td>Denver</td>
<td>Seattle</td>
<td>120</td>
<td>Bands</td>
<td>Denver</td>
<td>Seattle</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Denver</td>
<td>Coils</td>
<td>Detroit</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Boston</td>
<td>Coils</td>
<td>Detroit</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>New York</td>
<td>Coils</td>
<td>Detroit</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Seattle</td>
<td>Bands</td>
<td>Detroit</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Bands</td>
<td>Denver</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Bands</td>
<td>Boston</td>
<td>-50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Bands</td>
<td>New York</td>
<td>-50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Bands</td>
<td>Seattle</td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Coils</td>
<td>Detroit</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Coils</td>
<td>Denver</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Coils</td>
<td>Boston</td>
<td>-40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Coils</td>
<td>New York</td>
<td>-30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Coils</td>
<td>Seattle</td>
<td>-30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Direct Spreadsheet Interface

“2D” spreadsheet ranges

![Excel Spreadsheet Example](image)
Direct spreadsheet interface

Data Handling

Script (input)

```plaintext
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;
```
Data Handling

Script (input)

```plaintext
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)

```plaintext
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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>shipments</td>
<td>ITEMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Detroit</td>
<td>Boston</td>
<td>50</td>
<td>30</td>
<td>Detroit</td>
<td>Boston</td>
<td>80</td>
<td>80.0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Detroit</td>
<td>New York</td>
<td>0</td>
<td>30</td>
<td>Detroit</td>
<td>New York</td>
<td>30</td>
<td>37.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Detroit</td>
<td>Seattle</td>
<td>0</td>
<td>0</td>
<td>Detroit</td>
<td>Seattle</td>
<td>0</td>
<td>0.0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Denver</td>
<td>Boston</td>
<td>0</td>
<td>10</td>
<td>Denver</td>
<td>Boston</td>
<td>10</td>
<td>8.3%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Denver</td>
<td>New York</td>
<td>50</td>
<td>0</td>
<td>Denver</td>
<td>New York</td>
<td>50</td>
<td>41.7%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Denver</td>
<td>Seattle</td>
<td>10</td>
<td>30</td>
<td>Denver</td>
<td>Seattle</td>
<td>40</td>
<td>33.3%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Excel spreadsheet image]
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

![Graph showing reading efficiency](image)
Writing Efficiency

![Graph showing the relationship between write time and number of rows for different tools: eodbc, mplcsv, and mplxl. The x-axis represents the number of rows, and the y-axis represents time in seconds, with a log scale.]
Updating Efficiency

![Graph showing time (seconds) vs. number of rows with different lines representing eodbc, amplcsv, and amplxl]
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*
- Snapshots
- AMPL Colaboratory
Snapshots

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

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

```plaintext
###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) in ARCS} 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\amplib64_m.dll';
option ampl_Include 'C:\Users\AMPL\Desktop\Analytics\Netflow';
###options-end

###data-start
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
```

Robert Fourer, Optimization in Your Toolchain
INFORMS Analytics, Houston — 3-5 April 2022 — Technology Tutorials
Snapshot Usage

For use with an API

```python
snapshot = ampl.get_output('snapshot;')
ampl2.eval(snapshot)
```

For use in interactive debugging

```python
snapshot = ampl.get_output('snapshot;')
print(snapshot, file=open('snapshot.run', 'w'))
include "snapshot.run";
```
AMPL Colaboratory

Run sample models free in Jupyter notebooks
  - Run a short Python cell for setup
  - Run AMPL cells for model, data, scripts
    \[\ldots\text{execute in Google Colab, Kaggle, etc.}\]

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
Deploying optimization in containers and in the cloud

Virtual 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
- For clouds • containers • clusters ... even physical machines

Container setup

- Use any base image
- Load needed AMPL and solver modules
- Initialize virtual licensing
**Virtual Licensing**

*Short-term leases*
- Last 5 minutes
- Automatically renewed

*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*
Virtual Licensing

Flexible license limits

- No automatic enforcement
- Short-term use in excess of limits is expected
- If use exceeds limits in longer term, customer is contacted

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
Container Setup

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

```bash
# 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
```
Learn More

https://dev.ampl.com
  new AMPL development projects

https://github.com/ampl/
  all AMPL open-source projects

https://colab.ampl.com/
  AMPL Colaboratory links