New and Forthcoming Developments in the AMPL Modeling Language & System

Robert Fourer
AMPL Optimization
www.ampl.com — 773-336-AMPL

INFORMS Conference on Business Analytics and Operations Research
San Antonio, Texas — 7-9 April 2013
Track 11, Software Tutorials
Outline

Essentials

- Why AMPL?
- AMPL’s users

Enhancements: Building & maintaining models

- More natural formulations
  - Logical conditions
  - Quadratic constraints
- Integrated development environment (AMPL IDE)

Enhancements: Deploying models

- Application programming interfaces (AMPL API)
  - C++, Java, .NET, Python
  - MATLAB, R
The Optimization Modeling Cycle

**Steps**
- Communicate with problem owner
- Build model
- Prepare data
- Generate optimization problem
- Submit problem to solver
  - CPLEX, Gurobi, Xpress, KNITRO, CONOPT, MINOS, ...
- Report & analyze results
- **Repeat!**

**Goals**
- Do this quickly and reliably
- Get results before client loses interest
- Deploy for application
What Makes This Hard?

“We do not feel that the linear programming user’s most pressing need over the next few years is for a new optimizer that runs twice as fast on a machine that costs half as much (although this will probably happen). Cost of optimization is just not the dominant barrier to LP model implementation.

“The process required to manage the data, formulate and build the model, report on and analyze the results costs far more, and is much more of a barrier to effective use of LP, than the cost/performance of the optimizer.”

Krabek, Sjoquist, Sommer,
“The APEX Systems: Past and Future.”
*SIGMAP Bulletin* 29 (April 1980) 3-23.
Optimization Modeling Languages

Two forms of an optimization problem

- Modeler’s form
  - Mathematical description, easy for people to work with
- Algorithm’s form
  - Explicit data structure, easy for solvers to compute with

Idea of a modeling language

- A computer-readable modeler’s form
  - You write optimization problems in a modeling language
  - Computers translate to algorithm’s form for solution

Advantages of a modeling language

- Faster modeling cycles
- More reliable modeling and maintenance
Algebraic Modeling Languages

Formulation concept

- Define data in terms of sets & parameters
  - Analogous to database keys & records
- Define decision variables
- Minimize or maximize a function of decision variables
- Subject to equations or inequalities that constrain the values of the variables

Advantages

- Familiar
- Powerful
- Implemented
The AMPL Modeling Language

Features
- Algebraic modeling language
- Variety of data sources
- Connections to all solver features
- Interactive and scripted control

Advantages
- Powerful, general expressions
- Natural, easy-to-learn design
- Efficient processing scales well with problem size
Introductory Example

Multicommodity transportation . . .

- Products available at factories
- Products needed at stores
- Plan shipments at lowest cost

. . . with practical restrictions

- Cost has fixed and variables parts
- Shipments cannot be too small
- Factories cannot serve too many stores
Multicommodity Transportation

*Given*

- $O$ Set of origins (factories)
- $D$ Set of destinations (stores)
- $P$ Set of products

*and*

- $a_{ip}$ Amount available, for each $i \in O$ and $p \in P$
- $b_{jp}$ Amount required, for each $j \in D$ and $p \in P$
- $l_{ij}$ Limit on total shipments, for each $i \in O$ and $j \in D$
- $c_{ijp}$ Shipping cost per unit, for each $i \in O$, $j \in D$, $p \in P$
- $d_{ij}$ Fixed cost for shipping any amount from $i \in O$ to $j \in D$
- $s$ Minimum total size of any shipment
- $n$ Maximum number of destinations served by any origin
**Multicommodity Transportation**

**Mathematical Formulation**

**Determine**

- $X_{ijp}$ Amount of each $p \in P$ to be shipped from $i \in O$ to $j \in D$
- $Y_{ij}$ 1 if any product is shipped from $i \in O$ to $j \in D$
  0 otherwise

**to minimize**

\[
\sum_{i \in O} \sum_{j \in D} \sum_{p \in P} c_{ijp} X_{ijp} + \sum_{i \in O} \sum_{j \in D} d_{ij} Y_{ij}
\]

Total variable cost plus total fixed cost
Multicommodity Transportation

Mathematical Formulation

Subject to

\[ \sum_{j \in D} X_{ijp} \leq a_{ip} \quad \text{for all } i \in O, p \in P \]

Total shipments of product \( p \) out of origin \( i \)
must not exceed availability

\[ \sum_{i \in O} X_{ijp} = b_{jp} \quad \text{for all } j \in D, p \in P \]

Total shipments of product \( p \) into destination \( j \)
must satisfy requirements
Multicommodity Transportation

Mathematical Formulation

Subject to

\[ \sum_{p \in P} X_{ijp} \leq l_{ij} Y_{ij} \quad \text{for all } i \in O, j \in D \]

When there are shipments from origin \( i \) to destination \( j \),
the total may not exceed the limit, and \( Y_{ij} \) must be 1

\[ \sum_{p \in P} X_{ijp} \geq s Y_{ij} \quad \text{for all } i \in O, j \in D \]

When there are shipments from origin \( i \) to destination \( j \),
the total amount of shipments must be at least \( s \)

\[ \sum_{j \in D} Y_{ij} \leq n \quad \text{for all } i \in O \]

Number of destinations served by origin \( i \)
must be as most \( n \)
Multicommodity Transportation

AMPL Formulation

Symbolic data

```
set ORIG;   # origins
set DEST;   # destinations
set PROD;   # products

param supply {ORIG,PROD} >= 0;  # availabilities at origins
param demand {DEST,PROD} >= 0;  # requirements at destinations
param limit {ORIG,DEST} >= 0;   # capacities of links
param vcost {ORIG,DEST,PROD} >= 0; # variable shipment cost
param fcost {ORIG,DEST} > 0;     # fixed usage cost
param minload >= 0;             # minimum shipment size
param maxserve integer > 0;     # maximum destinations served
```
Multicommodity Transportation

AMPL Formulation

Symbolic model: variables and objective

```
var Trans {ORIG, DEST, PROD} >= 0;   # actual units to be shipped
var Use {ORIG, DEST} binary;       # 1 if link used, 0 otherwise

minimize Total_Cost:
  sum {i in ORIG, j in DEST, p in PROD} vcost[i,j,p] * Trans[i,j,p]
  + sum {i in ORIG, j in DEST} fcost[i,j] * Use[i,j];
```

\[ \sum_{i \in O} \sum_{j \in D} \sum_{p \in P} c_{ijp} X_{ijp} + \sum_{i \in O} \sum_{j \in D} d_{ij} Y_{ij} \]
Multicommodity Transportation

AMPL Formulation

Symbolic model: constraint

subject to Supply \{i in ORIG, p in PROD\}:

\[
\sum_{j in DEST} Trans[i,j,p] \leq supply[i,p];
\]

\[
\sum_{j\in D} X_{ijp} \leq a_{ip}, \text{ for all } i \in O, p \in P
\]
Multicommodity Transportation

AMPL Formulation

Symbolic model: constraints

subject to Supply \{i in ORIG, p in PROD\}:
    \[ \sum \{j in DEST\} Trans[i,j,p] \leq supply[i,p]; \]

subject to Demand \{j in DEST, p in PROD\}:
    \[ \sum \{i in ORIG\} Trans[i,j,p] = demand[j,p]; \]

subject to Multi \{i in ORIG, j in DEST\}:
    \[ \sum \{p in PROD\} Trans[i,j,p] \leq limit[i,j] \times Use[i,j]; \]

subject to Min_Ship \{i in ORIG, j in DEST\}:
    \[ \sum \{p in PROD\} Trans[i,j,p] \geq minload \times Use[i,j]; \]

subject to Max_Serve \{i in ORIG\}:
    \[ \sum \{j in DEST\} Use[i,j] \leq maxserve; \]
Multicommodity Transportation

AMPL Formulation

Explicit data independent of symbolic model

```AMPL
set ORIG := GARY CLEV PITT ;
set DEST := FRA DET LAN WIN STL FRE LAF ;
set PROD := bands coils plate ;
param supply (tr):  GARY CLEV PITT :=
    bands   400    700    800
    coils   800   1600   1800
    plate   200    300    300 ;
param demand (tr):
    FRA DET LAN WIN STL FRE LAF :=
    bands   300   300   100    75   650   225   250
    coils   500   750   400   250   950   850   500
    plate   100   100    0    50   200   100   250 ;
param limit default 625 ;
param minload := 375 ;
param maxserve := 5 ;
```
Multicommodity Transportation

AMPL Formulation

Explicit data (continued)

```plaintext
param vcost :=
  [*,*,bands]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
    GARY   30   10    8   10   11   71    6
    CLEV   22    7   10    7   21   82   13
    PITT   19   11   12   10   25   83   15
  [*,*,coils]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
    GARY   39   14   11   14   16   82    8
    CLEV   27    9   12    9   26   95   17
    PITT   24   14   17   13   28   99   20
  [*,*,plate]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
    GARY   41   15   12   16   17   86   18
    CLEV   29    9   13    9   28   99   18
    PITT   26   14   17   13   31  104   20 ;

param fcost:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
  GARY  3000 1200 1200 1200 2500 3500 2500
  CLEV  2000 1000 1500 1200 2500 3000 2200
  PITT  2000 1200 1500 1500 2500 3500 2200 ;
```
AMPL Solution

Model + data = problem instance to be solved

```
ampl: model multimipG.mod;
ampl: data multimipG.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.5.0: optimal solution; objective 235625
289 simplex iterations
28 branch-and-cut nodes
ampl: display Use;
Use [*,*]
  :    DET  FRA  FRE  LAF  LAN  STL  WIN    :=
CLEV  1   1   1   0   1   1   0
GARY  0   0   0   1   0   1   1
PITT  1   1   1   1   0   1   0
;
```

Multicommodity Transportation
Multicommodity Transportation

AMPL Solution

Solver choice independent of model and data

```
AMPL: model multimipG.mod;
AMPL: data multimipG.dat;
AMPL: option solver cplex;
AMPL: solve;
CPLEX 12.5.0.1: optimal integer solution; objective 235625
112 MIP simplex iterations
0 branch-and-bound nodes
AMPL: display Use;
Use [*,*]
:   DET FRA FRE LAF LAN STL WIN   :=
  CLEV  1 1 1 0 1 1 0
  GARY  0 0 0 1 0 1 1
  PITT  1 1 1 1 0 1 0
;  
```


**Multicommodity Transportation**

**AMPL Solution**

*Examine results*

```ampl
ampl: display {i in ORIG, j in DEST}

ampl? sum {p in PROD} Trans[i,j,p] / limit[i,j];

:   DET  FRA  FRE  LAF  LAN  STL  WIN  :=
CLEV  1    0.6  0.88 0    0.8  0.88 0
GARY  0    0    0    0.64 0    1    0.6
PITT  0.84 0.84 1    0.96 0    1    0

;  

ampl: display Max_Serve.body;

CLEV  5
GARY  3
PITT  5

;  

ampl: display TotalCost,

ampl? sum {i in ORIG, j in DEST} fcost[i,j] * Use[i,j];

TotalCost = 235625
sum {i in ORIG, j in DEST} fcost[i,j]*Use[i,j] = 27600
```
Indexed over sets of pairs and triples

set ORIG;   # origins
set DEST;   # destinations
set PROD;   # products
set SHIP within {ORIG,DEST,PROD};
    # (i,j,p) in SHIP ==> can ship p from i to j
set LINK = setof {(i,j,p) in SHIP} (i,j);
    # (i,j) in LINK ==> can ship some products from i to j

var Trans {SHIP} >= 0;   # actual units to be shipped
var Use {LINK} binary;   # 1 if link used, 0 otherwise

minimize Total_Cost:
    sum {(i,j,p) in SHIP} vcost[i,j,p] * Trans[i,j,p]
+ sum {(i,j) in LINK} fcost[i,j] * Use[i,j];
Multicommodity Transportation

AMPL “Sparse” Network

Constraint for dense network

subject to Supply {i in ORIG, p in PROD}:
    sum {j in DEST} Trans[i,j,p] <= supply[i,p];

Constraint for sparse network

subject to Supply {i in ORIG, p in PROD}:
    sum {(i,j,p) in SHIP} Trans[i,j,p] <= supply[i,p];
Multicommodity Transportation

AMPL Scripting

Script to test sensitivity to serve limit

```AMPL
model multmipG.mod;
data multmipG.dat;
option solver gurobi;
for {m in 7..1 by -1} {
    let maxserve := m;
solve;
    if solve_result = 'infeasible' then break;
display maxserve, Max_Serve.body;
}
```
Multicommodity Transportation

AMPL Scripting

Run showing sensitivity to serve limit

```plaintext
ampl: include multimipServ.run;

Gurobi 5.5.0: optimal solution; objective 233150
maxserve = 7
CLEV 5   GARY 3   PITT 6

Gurobi 5.5.0: optimal solution; objective 233150
maxserve = 6
CLEV 5   GARY 3   PITT 6

Gurobi 5.5.0: optimal solution; objective 235625
maxserve = 5
CLEV 5   GARY 3   PITT 5

Gurobi 5.5.0: infeasible
```
Multicommodity Transportation

AMPL Scripting

Script to generate n best solutions

```AMPL
param nSols default 0;
param maxSols;
model multmipG.mod;
data multmipG.dat;
set USED {1..nSols} within {ORIG,DEST};
subject to exclude {k in 1..nSols}:
    sum {(i,j) in USED[k]} (1-Use[i,j]) +
    sum {(i,j) in {ORIG,DEST} diff USED[k]} Use[i,j] >= 1;
option solver gurobi;
repeat {
    solve;
    display Use;
    let nSols := nSols + 1;
    let USED[nSols] := {i in ORIG, j in DEST: Use[i,j] > .5};
} until nSols = maxSols;
```
Multicommodity Transportation

AMPL Scripting

Run showing 3 best solutions

```AMPL
ampl: include multmipBest.run;

Gurobi 5.5.0: optimal solution; objective 235625
:    DET FRA FRE LAF LAN STL WIN    :=
   CLEV   1   1   1   0   1   1   0
   GARY   0   0   0   1   0   1   1
   PITT   1   1   1   1   0   1   0 ;

Gurobi 5.5.0: optimal solution; objective 237125
:    DET FRA FRE LAF LAN STL WIN    :=
   CLEV   1   1   1   1   0   1   0
   GARY   0   0   0   1   0   1   1
   PITT   1   1   1   0   1   1   0 ;

Gurobi 5.5.0: optimal solution; objective 238225
:    DET FRA FRE LAF LAN STL WIN    :=
   CLEV   1   0   1   0   1   1   1
   GARY   0   1   0   1   0   1   0
   PITT   1   1   1   1   0   1   0 ;
```
AMPL’s Users

Business
- Customer relationships
- Customer areas
- Project examples

Government

Academic
AMPL's Users

Business Customer Relationships

Internal projects
  - We supply software & answer a few questions
  - Company’s employees build the models

Training and consulting
  - Available on request
AMPL's Users

Business Customer Areas

Transportation
- Air, rail, truck

Production
- Planning
  - steel
  - automotive
- Supply chain
  - consumer products

Finance
- Investment banking
- Insurance

Natural resources
- Electric power
- Gas distribution
- Mining

Information technology
- Telecommunications
- Internet services

Consulting practices
- Management
- Industrial engineering
AMPL's Users

Business Customer Examples

Three award-winning projects

- Norske Skog (paper manufacturer)
- ZARA (clothing retailer)
- Carlson Rezidor (hotel operator)

. . . finalists for Edelman Award for practice of Operations Research
**Optimization of production and distribution**

- Australasia
- Europe
  - 640 binary variables
  - 524,000 continuous variables
  - 33,000 constraints

**Optimization of shutdown decisions worldwide**

- Multiple scenarios
- Numerous sensitivity analyses
- **Key role of AMPL models**
  - Implemented in a few weeks
  - Modified to analyze alternatives
  - Run interactively at meetings
**Optimization of worldwide shipments**

- Legacy process
- New process

- Piecewise-linear AMPL model with integer variables
- Run once for each product each week
- Decides how much of each size to ship to each store
- Increases sales 3-4%

---

**ZARA**

**AMPLe's Users**

---

*Robert Fourer, New & Forthcoming Developments in AMPL*  
**INFORMS Conf on Analytics — 7-9 April 2013 — Track 11, Software Tutorials***
Optimization of hotel room rates

- Traditional process
  - Maximizes revenues, given . . .
    * Availability of prices at competing hotels
    * Objective measures of price elasticity of demand
  - Prototyped in AMPL with quadratic objective
  - Increases revenues 2-4%

- New process

**AMPL's Users**

Carlson Rezidor
AMPL’s Users

Government Customers

Financial agencies
- United States
- Canada
- Sweden

U.S. departments
- Census Bureau
- Army Corps of Engineers

U.S. research centers
- Argonne National Laboratory
- Sandia National Laboratories
- Lawrence Berkeley Laboratory
AMPL's Users

Academic Customers

Research
- Over 250 university installations worldwide
- Nearly 1000 citations in scientific papers
  - engineering, science, economics, management

Teaching
- Linear & nonlinear optimization
  - Graph optimization
  - Stochastic programming
- Operations Research
- Specialized courses
  - Supply chain modeling
  - Electric power system planning
  - Transportation logistics
  - Communication network design & algorithms

Over 70 courses so far in 2013
Enhancements

Building & maintaining models

- More natural formulations
  - Logical conditions
  - Quadratic constraints

- AMPL IDE (Integrated Development Environment)
  - Unified editor & command processor
  - Built on the Eclipse platform

Deploying models

- AMPL API (Application Programming Interfaces)
  - Programming languages: C++, Java, .NET, Python
  - Analytics languages: MATLAB, R
More Natural Modeling

Logical Conditions

Common “not linear” expressions

- Disjunctions (or), implications (==>)
- Counting expressions (count),
  Counting constraints (atleast, atmost)
- Aggregate constraints (alldiff, numberof)

Variety of solvers

- CPLEX mixed-integer solver
  * Applied directly
  * Applied after conversion to MIP
- Constraint solvers
  * IBM ILOG CP
  * Gecode
  * JaCoP (coming soon)
Example: Multi-Commodity *(revisited)*

Minimum-shipment constraints

- From each origin to each destination, *either* ship nothing *or* ship at least minload units

Conventional linear mixed-integer formulation

```plaintext
var Trans {ORIG, DEST, PROD} >= 0;
var Use {ORIG, DEST} binary;
....... subject to Multi {i in ORIG, j in DEST}:
   sum {p in PROD} Trans[i,j,p] <= limit[i,j] * Use[i,j];
subject to Min_Ship {i in ORIG, j in DEST}:
   sum {p in PROD} Trans[i,j,p] >= minload * Use[i,j];
```
Multi-Commodity

Zero-One Alternatives

Mixed-integer formulation using implications

subject to Multi_Min_Ship {i in ORIG, j in DEST}:
  Use[i,j] = 1  ==>  
  minload <= sum {p in PROD} Trans[i,j,p] <= limit[i,j] 
  else sum {p in PROD} Trans[i,j,p] = 0;

Solved directly by CPLEX

ampl: model multmipImpl.mod;
ampl: data multmipG.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.5.0.1: optimal integer solution; objective 235625
175 MIP simplex iterations
0 branch-and-bound nodes
Multi-Commodity

Non-Zero-One Alternatives

Disjunctive constraint

```
subject to Multi_Min_Ship {i in ORIG, j in DEST}:
    sum {p in PROD} Trans[i,j,p] = 0  or
    minload <= sum {p in PROD} Trans[i,j,p] <= limit[i,j];
```

Solved by CPLEX after automatic conversion

```
ampl: model multmipDisj.mod;
ampl: data multmipG.dat;
ampl: solve;
CPLEX 12.5.0.1: logical constraint not indicator constraint.
ampl: option solver ilogcp;
ampl: option ilogcp_options 'optimizer cplex';
ampl: solve;
ilogcp 12.4.0: optimal solution
  0 nodes, 175 iterations, objective 235625
```
Example: Optimal Arrangement

*Optimally line up a group of people*

- Given a set of adjacency preferences, maximize the number that are satisfied

**Decision variables**

- For each preference “i1 adjacent to i2”:
  \[ \text{Sat}[i_1, i_2] = 1 \text{ iff this is satisfied in the lineup} \]
- \( \text{Pos}[i] \) is the position of person \( i \) in the line

\[ \ldots \text{fewer variables, larger domains} \]
"CP-Style" Alternative

All-different constraint

```AMPL
param nPeople integer > 0;
set PREFS within {i1 in 1..nPeople, i2 in 1..nPeople: i1 <> i2};

var Sat {PREFS} binary;
var Pos {1..nPeople} integer >= 1, <= nPeople;

maximize NumSat: sum {(i1,i2) in PREFS} Sat[i1,i2];

subject to OnePersonPerPosition:
   alldiff {i in 1..nPeople} Pos[i];

subject to SatDefn {(i1,i2) in PREFS}:
   Sat[i1,i2] = 1 <==> Pos[i1]-Pos[i2] = 1 or Pos[i2]-Pos[i1] = 1;

subject to SymmBreaking:
   Pos[1] < Pos[2];
```
Arrangement

“CP-Style” Alternative (cont’d)

11 people, 20 preferences

```ampl
ampl: model photo.mod;
ampl: data photo11.dat;
ampl: option solver ilogcp;
ampl: solve;
ilogcp 12.5.0: optimizer cp
ilogcp 12.5.0: optimal solution
8837525 choice points, 8432821 fails, objective 12
ampl: option solver gecode;
ampl: solve;
gecode 3.7.3: optimal solution
589206448 nodes, 294603205 fails, objective 12
ampl:
```
AMPL IDE

Integrated Development Environment

- Unified editor & command processor
- Included in the AMPL distribution
  - Easy upgrade path
  - Command-line, batch versions remain available
- Built on the Eclipse platform

Planned availability . . .
**AMPL IDE**

Sample Screenshot
**AMPL IDE**

**Planned Availability**

**Rollout dates**

- Beta test
  - May-June 2013
  - *Seeking beta testers now*

- Release
  - Summer-Fall 2013
  - Available with all new downloads

**Development details**

- Partnership with OptiRisk Systems
- "AMPLDEV" advanced IDE to be marketed by OptiRisk
  - Offers full stochastic programming support
AMPL API

Application Programming Interface
- Programming languages: C++, Java, .NET, Python
- Analytics languages: MATLAB, R

Facilitates use of AMPL for
- Complex algorithmic schemes
- Embedding in other applications
- Deployment of models
**AMPL API**

**Deployment Alternatives**

**Stand-alone:** *Give (temporary) control to AMPL*

- Write needed files
- Invoke AMPL to run some scripts
- Read the files that AMPL leaves on exit

**API:** *Interact with AMPL*

- Execute AMPL statements individually
- Read model, data, script files when convenient
- Exchange data tables directly with AMPL
  * populate sets & parameters
  * invoke any available solver
  * extract values of variables & result expressions

  . . . *all embedded within your program’s logic*
**AMPL API**

**Example: Java**

**Efficient frontier: Initialize, read files**

```java
AMPL ampl = createAMPL();
int steps = 30;
try{
    ampl.interpretFile(Utils.getResFileName("qpmv.mod","qpmv",true),false);
    ampl.interpretFile(Utils.getResFileName("qpmv.dat","qpmv",true),true);
}
catch (IOException e){
    e.printStackTrace();
    return -1;
}
VariableMap portfolioReturn = ampl.getVariable('portret');
ParameterMap averageReturn = ampl.getParameter('averret');
ParameterMap targetReturn = ampl.getParameter('targetret');
ObjectiveMap deviation = ampl.getObjective('cst');
```
Example: Java (cont’d)

Efficient frontier: Solve, set up for loop

```java
ampl.interpret("option solver afortmp;");
ampl.interpret("let stockopall:={ }; let stockrun:=stockall;"酹); 
ampl.interpret("option relax_integrality 1;"酹);
ampl.solve()

double minret = portfolioReturn.get().value();
double maxret = findMax(averageReturn.getDouble());
double stepsize = (maxret-minret)/steps;
double[] returns = new double[steps];
double[] deviations = new double[steps];
```
**Example: Java (cont’d)**

**Efficient frontier: Loop over solves**

```java
for(int i=0; i<steps; i++)
{
    System.out.println(String.format
        ("Solving for return = %f", maxret - (i-1)*stepsize));
    targetReturn.let(maxret - (i-1)*stepsize);
    ampl.interpret("let stockopall:={ }; let stockrun:=stockall;");
    ampl.interpret("options relax_integrality 1;");
    ampl.solve();
    ampl.interpret("let stockrun2:={i in stockrun:weights[i]>0};");
    ampl.interpret(" let stockrun:=stockrun2;");
    ampl.interpret(" let stockpall:={i in stockrun:weights[i]>0.5};");
    ampl.interpret("options relax_integrality 0;");
    ampl.solve();
    returns[i] = maxret - (i-1)*stepsize;
    deviations[i] = deviation.get().value();
}
```
**AMPL API**

**Example: MATLAB**

*Efficient frontier: Initialize, read files*

```plaintext
ampl = initAMPL;
steps = 30;
ampl.interpretFile('qpmv.mod', false)
ampl.interpretFile('qpmv.dat', true)
portfolioReturn = ampl.getVariable('portret');
averageReturn = ampl.getParameter('averret');
targetReturn = ampl.getParameter('targetret');
deviation = ampl.getObjective('cst');
```
Example: MATLAB (cont’d)

Efficient frontier: Solve, set up for loop

```ampl
ampl.interpret('option solver afortmp;');
ampl.interpret('let stockopall:={}; let stockrun:=stockall;');
ampl.interpret('option relax_integrality 1;');
ampl.solve()
minret = portfolioReturn.getDouble();
maxret = max(averageReturn.getDouble());
stepsize = (maxret-minret)/steps;
returns = zeros(steps, 1);
deviations = zeros(steps, 1);
```
Example: MATLAB (cont’d)

Efficient frontier: Loop over solves

```matlab
for i=1:steps
    fprintf('Solving for return = %f\n', maxret - (i-1)*stepsize)
    targetReturn.let(maxret - (i-1)*stepsize);
    ampl.interpret('let stockopall:={}; let stockrun:=stockall;');
    ampl.interpret('option relax_integrality 1;');
    ampl.solve();
    ampl.interpret('let stockrun2:={i in stockrun:weights[i]>0};');
    ampl.interpret('let stockrun:=stockrun2;');
    ampl.interpret('let stockopall:={i in stockrun:weights[i]>0.5};');
    ampl.interpret('option relax_integrality 0;');
    ampl.solve();
    returns(i) = maxret - (i-1)*stepsize;
    deviations(i) = deviation.getDouble();
end
plot(returns, deviations)
```
**AMPL API**

**Planned Availability**

**Rollout dates**

- Beta test (Java, MATLAB, ...)
  - Summer 2013
  - *Seeking beta testers now*

- Release
  - Fall 2013
  - Available with all new downloads

**Development details**

- Partnership with OptiRisk Systems
- At least 6 languages to be provided
Readings *(AMPL)*


Readings *(Interfaces)*

