Alternatives for Scripting in the AMPL Modeling Language

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

Industrial Engineering & Management Sciences,
Northwestern University

INFORMS International
Beijing, China — 24-27 June 2012
Session TB20, Software Tutorials
Alternatives for Scripting in Conjunction with an Algebraic Modeling Language for Optimization

Modeling languages are essentially declarative, yet successful optimization languages also offer ways to write interpreted scripts that resemble executable programs. What can scripting in a modeling language offer in comparison to modeling in a general-purpose scripting language? Answers will be suggested through varied examples of problem analyses and iterative schemes.
Alternatives for

Programming
in conjunction with an
Algebraic Modeling Language
for
Optimization

Robert Fourer
Department of Industrial Engineering
and Management Sciences
Northwestern University
Evanston, Illinois 60208-3119
4er@iems.nwu.edu

David M. Gay
AT&T Bell Laboratories
Murray Hill, New Jersey 07974-0636
dmg@research.att.com

INFORMS National Meeting
New Orleans, October 30, 1995
Topics: Introduction to AMPL

The optimization modeling cycle

Optimization modeling languages

Example: multicommodity transportation

- Mathematical formulation
- AMPL formulation
- AMPL solution
Topics: Scripting in AMPL

1: *Parametric analysis*

2: *Solution generation*
   a: *via cuts*
   b: *via solver*

3: *Heuristic optimization*

4: *Pattern generation*

5: *Decomposition*

*Scripts in practice . . .*

*Prospective improvements . . .*
The Optimization Modeling Cycle

Steps
- Communicate with problem owner
- Build model
- Prepare data
- Generate optimization problem
- Submit problem to solver
  - CPLEX, Gurobi, 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.”
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
# AMPL with Gurobi

## Features

- Detection of all supported problem types
- Access to all algorithm & display options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggfill</td>
<td>amount of fill allowed during aggregation during Gurobi's presolve (default 10)</td>
</tr>
<tr>
<td>aggregate</td>
<td>whether to use aggregation during Gurobi presolve: 0 = no (sometimes reduces numerical errors), 1 = yes (default)</td>
</tr>
<tr>
<td>ams_eps</td>
<td>relative tolerance for reporting alternate MIP solutions (default = no limit)</td>
</tr>
<tr>
<td>ams_epabs</td>
<td>absolute tolerance for reporting alternate MIP solutions (default = no limit)</td>
</tr>
<tr>
<td>ams_limit</td>
<td>limit on number of alternate MIP solutions written (default = number of available alternate solutions)</td>
</tr>
<tr>
<td>ams_stub</td>
<td>stub for alternate MIP solutions. The number of alternative MIP solution files written is determined by three keywords: ams_limit gives the maximum number of files written; ams_eps gives a relative tolerance on the objective values of alternative solutions; ams_epabs gives an absolute tolerance on how much worse the objectives can be.</td>
</tr>
</tbody>
</table>
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
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

```AMPL
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
```

Robert Fourer, Alternatives for Scripting in the AMPL Modeling Language
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}:

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

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

AMPL Formulation

Symbolic model: constraints

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

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

subject to Multi \{i in \text{ORIG, j in \text{DEST}}\}:
    \[ \text{sum} \{p in \text{PROD}\} \ Trans[i,j,p] <= \text{limit}[i,j] * \text{Use}[i,j]; \]

subject to Min_Ship \{i in \text{ORIG, j in \text{DEST}}\}:
    \[ \text{sum} \{p in \text{PROD}\} \ Trans[i,j,p] >= \text{minload} * \text{Use}[i,j]; \]

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

AMPL Formulation

Explicit data independent of symbolic model

```
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)

```AMPL
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  8
    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 ;
```

Robert Fourer, Alternatives for Scripting in the AMPL Modeling Language
INFORMS Int'l Beijing — 24-27 June, 2012 — TB20, Software Tutorials
**Multicommodity Transportation**

**AMPL Solution**

*Model + data = problem instance to *be* solved*

```
AMPL: model multmipG.mod;
AMPL: data multmipG.dat;
AMPL: option solver gurobi;
AMPL: solve;
Gurobi 5.0.0: optimal solution; objective 235625
394 simplex iterations
46 branch-and-cut nodes
AMPL: display Use;
Use [*,*]
  :  DET FRA FRE LAF LAN STL WIN  :=
CLEV  1 1 1 0 1 1 1 0
GARY  0 0 0 1 0 1 1 1
PITT  1 1 1 1 0 1 1 0
;```

Robert Fourer, Alternatives for Scripting in the AMPL Modeling Language
INFORMS Int'l Beijing — 24-27 June, 2012 — TB20, Software Tutorials
Multicommodity Transportation

AMPL Solution

Solver choice independent of model and data

```ampl
ampl: model multipG.mod;
ampl: data multipG.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.4.0.0: optimal integer solution; objective 235625
394 MIP simplex iterations
41 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
;
```

Robert Fourer, Alternatives for Scripting in the AMPL Modeling Language
INFORMS Int'l Beijing — 24-27 June, 2012 — TB20, Software Tutorials
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
```
Multicommodity Transportation

AMPL “Sparse” Network

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 “Sparse” Network

All constraints

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

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

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

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

subject to Max_Serve {i in ORIG}:
    \[ \text{sum \{(i,j) in LINK\} Use[i,j] \leq maxserve}; \]
### Multicommodity Transportation

#### AMPL “Sparse” Network

**1st dataset: shipments allowed**

```plaintext
set SHIP :=

(*,*,bands):  FRA DET LAN WIN STL FRE LAF :=
   GARY   +   +   +   +   +   -   +
   CLEV   +   -   +   -   +   +   +
   PITT   -   +   +   +   +   +   +

(*,*,coils):  FRA DET LAN WIN STL FRE LAF :=
   GARY   +   +   +   +   +   +   -
   CLEV   +   +   -   +   +   +   +
   PITT   +   +   +   +   +   +   +

(*,*,plate):  FRA DET LAN WIN STL FRE LAF :=
   GARY   +   +   -   +   +   -   +
   CLEV   +   +   +   +   +   +   +
   PITT   -   +   +   -   +   +   +
```

Multicommodity Transportation

AMPL “Sparse” Network

2nd dataset: shipments allowed

```
set SHIP :=
    (*,*,bands):  FRA DET LAN WIN STL FRE LAF :=
        GARY    +    +    +    +    +    -    -
        CLEV    -    +    +    -    +    +    +
        PITT    +    -    +    +    +    +    +
    (*,*,coils): FRA DET LAN WIN STL FRE LAF :=
        GARY    +    +    +    +    +    +    +
        CLEV    +    +    -    +    +    +    +
        PITT    +    +    +    +    +    +    +
    (*,*,plate): FRA DET LAN WIN STL FRE LAF :=
        GARY    -    +    +    +    +    -    +
        CLEV    +    +    +    +    +    +    +
        PITT    +    +    -    -    +    +    +
```

Multicommodity Transportation

AMPL “Sparse” Network

2nd dataset: shipments allowed

```
set SHIP :=
    (*,*,bands):  FRA DET LAN WIN STL FRE LAF :=
        GARY    +    +    +    +    +    -    -
        CLEV    -    +    +    -    +    +    +
        PITT    +    -    +    +    +    +    +
    (*,*,coils): FRA DET LAN WIN STL FRE LAF :=
        GARY    +    +    +    +    +    +    +
        CLEV    +    +    -    +    +    +    +
        PITT    +    +    +    +    +    +    +
    (*,*,plate): FRA DET LAN WIN STL FRE LAF :=
        GARY    -    +    +    +    +    -    +
        CLEV    +    +    +    +    +    +    +
        PITT    +    +    -    -    +    +    +
```
Multicommodity Transportation

AMPL “Sparse” Network

Same model, different data

```
ampl: model multmipT.mod;
ampl: data multmipT1.dat;
ampl: solve;
Gurobi 4.6.0: optimal solution; objective 247725
108 simplex iterations
13 branch-and-cut nodes
ampl: reset data;
ampl: data multmipT2.dat;
ampl: solve;
Gurobi 4.6.0: optimal solution; objective 237775
79 simplex iterations
ampl:
```
1: Parametric Analysis

Try different limits on destinations served

- Reduce parameter maxserve and re-solve
  - until there is no feasible solution
- Display results
  - parameter value
  - numbers of destinations actually served

Try different supplies of plate at Gary

- Increase parameter supply['GARY','plate'] and re-solve
  - until dual is zero (constraint is slack)
- Record results
  - distinct dual values
  - corresponding objective values

... display results at the end
Parametric Analysis on limits

Script to test sensitivity to serve limit

```AMPL
model multimipG.mod;
data multimipG.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;
}
```
Parametric Analysis on limits

Run showing sensitivity to serve limit

```
ampl: include multmipServ.run;

Gurobi 4.6.0: optimal solution; objective 233150
maxserve = 7
CLEV 5   GARY 3   PITT 6

Gurobi 4.6.0: optimal solution; objective 233150
maxserve = 6
CLEV 5   GARY 3   PITT 6

Gurobi 4.6.0: optimal solution; objective 235625
maxserve = 5
CLEV 5   GARY 3   PITT 5

Gurobi 4.6.0: infeasible
```
Parametric Analysis on supplies

Script to test sensitivity to plate supply at GARY

set SUPPLY default {};  
param sup_obj {SUPPLY};  
param sup_dual {SUPPLY};  

let supply['GARY','plate'] := 200;  
param sup_step = 10;  
param previous_dual default -Infinity;  

repeat while previous_dual < 0 {  
  solve;  
  if Supply['GARY','plate'].dual > previous_dual then {  
    let SUPPLY := SUPPLY union {supply['GARY','plate']};  
    let sup_obj[supply['GARY','plate']] := Total_Cost;  
    let sup_dual[supply['GARY','plate']] := Supply['GARY','plate'].dual;  
    let previous_dual := Supply['GARY','plate'].dual;  
  }  
  let supply['GARY','plate'] := supply['GARY','plate'] + supply_step;  
}
Parametric Analysis on supplies

Run showing sensitivity to plate supply at GARY

```ampl
ampl: include multipSupply.run;

ampl: display sup_obj, sup_dual;

:     sup_obj    sup_dual     :=
200 223504   -13
380 221171   -11.52
460 220260   -10.52
510 219754   -8.52
560 219413    0

;```

Parametric: Observations

Results of solve can be tested
  ▶ Check whether problem is no longer feasible
    * if solve_result = 'infeasible' then break;

Parameters are true objects
  ▶ Assign new value to param supply
    * let supply['GARY','plate'] :=
      supply['GARY','plate'] + supply_step;
  ▶ Problem instance changes accordingly

Sets are true data
  ▶ Assign new value to set SUPPLY
    * let SUPPLY := SUPPLY union {supply['GARY','plate']};
  ▶ All indexed entities change accordingly
2a: Solution Generation via Cuts

Same multicommodity transportation model

Generate $n$ best solutions using different routes

- Display routes used by each solution
### Solutions via Cuts

#### Script

```plaintext
param nSols default 0;
param maxSols = 3;
model multimipG.mod;
data multimipG.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;
repeat {
    solve;
    display Use;
    let nSols := nSols + 1;
    let USED[nSols] := {i in ORIG, j in DEST: Use[i,j] > .5};
} until nSols = maxSols;
```
AMPL Scripting

Run showing 3 best solutions

```
ampl: include multmipBestA.run;

Gurobi 4.6.0: 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  0  1  0  0;

Gurobi 4.6.0: 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  0  0;

Gurobi 4.6.0: 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  0  1  0  0;
```
Solutions via Cuts: Observations

Same expressions describe sets and indexing

- Index a summation
  - \( \cdots \text{sum} \{(i,j) \text{ in } \{\text{ORIG,DEST}\} \text{ diff USED}[k]\} \text{ Use}[i,j] >= 1; \)
- Assign a value to a set
  - \( \text{let} \text{ USED}[n\text{Sols}] := \{i \text{ in ORIG, } j \text{ in DEST: Use}[i,j] > .5\}; \)

New cuts defined automatically

- Index cuts over a set
  - \( \text{subject to exclude } \{k \text{ in } 1..n\text{Sols}\}: \cdots \)
- Add a cut by expanding the set
  - \( \text{let} \text{ nSols} := n\text{Sols} + 1; \)
2b: Solution Generation via Solver

Same model

Ask solver to return multiple solutions

- Set options
- Get all results from one “solve”
- Retrieve and display each solution
Solutions via Solver

Script

```AMPL
option solver cplex;
option cplex_options "poolstub=multmip poolcapacity=3 \ 
    populate=1 poolintensity=4 poolreplace=1";
solve;
for {i in 1..Current.npool} {
    solution ("multmip" & i & ".sol");
    display Use;
}
```
Solutions via Solver

Results

ampl: include multmipBestB.run;

CPLEX 12.4.0.0: poolstub=multmip
poolcapacity=3
populate=1
poolintensity=4
poolreplace=1

CPLEX 12.4.0.0: optimal integer solution; objective 235625
439 MIP simplex iterations
40 branch-and-bound nodes

Wrote 3 solutions in solution pool
to files multmip1.sol ... multmip3.sol.

Suffix npool OUT;
Solutions via Solver

Results (continued)

Solution pool member 1 (of 3); 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;

Solution pool member 2 (of 3); 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;

Solution pool member 3 (of 3); 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 1 0 1 0;
Solutions via Solver: Observations

Filenames can be formed dynamically

- Write a (string expression)
- Numbers are automatically converted
  
  * solution ("multmip" & i & ".sol");


3: Heuristic Optimization

Workforce planning

- Cover demands for workers
  - Each “shift” requires a certain number of employees
  - Each employee works a certain “schedule” of shifts

- Satisfy scheduling rules
  - Only “valid” schedules from given list may be used
  - Each schedule that is used at all must be worked by at least ?? employees

- Minimize total workers needed
  - Which schedules should be used?
  - How many employees should work each schedule?

Difficult instances

- Set ?? to a “hard” value
- Get a very good solution quickly
Heuristic

Model (sets, parameters)

```plaintext
set SHIFTS;               # shifts
param Nsched;             # number of schedules;
set SCHEDS = 1..Nsched;   # set of schedules
set SHIFT_LIST {SCHEDS} within SHIFTS;
param rate {SCHEDS} >= 0; # pay rates
param required {SHIFTS} >= 0; # staffing requirements
param least_assign >= 0;   # min workers on any schedule used
```
Heuristic

Model (variables, objective, constraints)

```plaintext
var Work {SCHEDS} >= 0 integer;
var Use {SCHEDS} >= 0 binary;

minimize Total_Cost:
    sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
    sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use1 {j in SCHEDS}:
    least_assign * Use[j] <= Work[j];

subject to Least_Use2 {j in SCHEDS}:
    Work[j] <= (max {i in SHIFT_LIST[j]} required[i]) * Use[j];
```
Heuristic

Data

set SHIFTS := Mon1 Tue1 Wed1 Thu1 Fri1 Sat1
  Mon2 Tue2 Wed2 Thu2 Fri2 Sat2
  Mon3 Tue3 Wed3 Thu3 Fri3 ;

param Nsched := 126 ;

set SHIFT_LIST[1] := Mon1 Tue1 Wed1 Thu1 Fri1 ;
set SHIFT_LIST[2] := Mon1 Tue1 Wed1 Thu1 Fri2 ;
set SHIFT_LIST[3] := Mon1 Tue1 Wed1 Thu1 Fri3 ;
set SHIFT_LIST[4] := Mon1 Tue1 Wed1 Thu1 Sat1 ;
set SHIFT_LIST[5] := Mon1 Tue1 Wed1 Thu1 Sat2 ;

param required :=
  Mon1 100  Mon2 78  Mon3 52
  Tue1 100  Tue2 78  Tue3 52
  Wed1 100  Wed2 78  Wed3 52
  Thu1 100  Thu2 78  Thu3 52
  Fri1 100  Fri2 78  Fri3 52
  Sat1 100  Sat2 78 ;
Heuristic

Hard case:  \texttt{least\_assign = 19}

```
ampl: model sched1.mod;
ampl: data sched.dat;
ampl: let least_assign := 19;
ampl: option solver cplex;
ampl: solve;

CPLEX 12.2.0.2: optimal integer solution; objective 269
635574195 MIP simplex iterations
86400919 branch-and-bound nodes

ampl: option omit_zero_rows 1, display_1col 0;
ampl: display Work;
Work [*] :=
  4 22   16 39   55 39   78 39   101 39   106 52   122 39
;

... 94.8 minutes
```
Heuristic

Alternative, indirect approach

- Step 1: Relax integrality of Work variables
  Solve for zero-one Use variables
- Step 2: Fix Use variables
  Solve for integer Work variables

... not necessarily optimal, but ...
Heuristic

Script

```AMPL
model sched1.mod;
data sched.dat;
let least_assign := 19;

let {j in SCHEDS} Work[j].relax := 1;
solve;

fix {j in SCHEDS} Use[j];
let {j in SCHEDS} Work[j].relax := 0;
solve;
```
Heuristic

Results

ampl: include sched1-fix.run;

CPLEX 12.2.0.2: optimal integer solution; objective 268.5
32630436 MIP simplex iterations
2199508 branch-and-bound nodes

Work [*] :=
  1 24  32 19  80 19.5  107 33  126 19.5
  3 19  66 19  90 19.5  109 19
  10 19  72 19.5 105 19.5  121 19 ;

CPLEX 12.2.0.2: optimal integer solution; objective 269
2 MIP simplex iterations
0 branch-and-bound nodes

Work [*] :=
  1 24  10 19  66 19  80 19  105 20  109 19  126 20
  3 19  32 19  72 19  90 20  107 33  121 19 ;

... 2.85 minutes
Heuristic: Observations

Models can be changed dynamically

- Adapt modeling expressions
- Execute model-related commands
  * fix {j in SCHEDS} Use[j];
- Assign values to properties of model components
  * let {j in SCHEDS} Work[j].relax := 1;
4: Pattern Generation

Roll cutting

- Min rolls cut (or material wasted)
- Decide number of each pattern to cut
- Meet demands for each ordered width

Generate cutting patterns

- Read general model
- Read data: demands, raw width
- Compute data: all usable patterns
- Solve problem instance
Pattern Generation

Model

```
param roll_width > 0;
set WIDTHS ordered by reversed Reals;
param orders {WIDTHS} > 0;

param maxPAT integer >= 0;
param nPAT integer >= 0, <= maxPAT;
param nbr {WIDTHS,1..maxPAT} integer >= 0;

var Cut {1..nPAT} integer >= 0;

minimize Number:
    sum {j in 1..nPAT} Cut[j];

subj to Fulfill {i in WIDTHS}:
    sum {j in 1..nPAT} nbr[i,j] * Cut[j] >= orders[i];
```
Pattern Generation

Data

\[
\text{param roll\_width := 90 ;}
\]

\[
\text{param: WIDTHS: orders :=}
\]

\[
\begin{array}{cc}
60 & 3 \\
30 & 21 \\
25.5 & 94 \\
20 & 50 \\
17.25 & 288 \\
15 & 178 \\
12.75 & 112 \\
10 & 144 \\
\end{array}
\]
Pattern Generation

Script (initialize)

```AMPL
model cutPAT.mod;
data ChvatalD.dat;
model;
param curr_sum >= 0;
param curr_width > 0;
param pattern {WIDTHS} integer >= 0;
let maxPAT := 100000000;
let nPAT := 0;
let curr_sum := 0;
let curr_width := first(WIDTHS);
let {w in WIDTHS} pattern[w] := 0;
```
Pattern Generation

Script (loop)

```plaintext
repeat {
    if curr_sum + curr_width <= roll_width then {
        let pattern[curr_width] := floor((roll_width-curr_sum)/curr_width);
        let curr_sum := curr_sum + pattern[curr_width] * curr_width;
    }
    if curr_width != last(WIDTHS) then
        let curr_width := next(curr_width,WIDTHS);
    else {
        let nPAT := nPAT + 1;
        let {w in WIDTHS} nbr[w,nPAT] := pattern[w];
        let curr_sum := curr_sum - pattern[last(WIDTHS)] * last(WIDTHS);
        let pattern[last(WIDTHS)] := 0;
        let curr_width := min {w in WIDTHS: pattern[w] > 0} w;
        if curr_width < Infinity then {
            let curr_sum := curr_sum - curr_width;
            let pattern[curr_width] := pattern[curr_width] - 1;
            let curr_width := next(curr_width,WIDTHS);
        } else break;
    }
}
```
Pattern Generation

Script (solve, report)

```plaintext
option solver gurobi;
solve;
printf "\n%5i patterns, %3i rolls", nPAT, sum {j in 1..nPAT} Cut[j];
printf "\n\n Cut ";
printf {j in 1..nPAT: Cut[j] > 0}: "%3i", Cut[j];
printf "\n\n";
for {i in WIDTHS} {
    printf "%7.2f ", i;
    printf {j in 1..nPAT: Cut[j] > 0}: "%3i", nbr[i,j];
    printf "\n";
}
printf "\nWASTE = %5.2f\n\n",
    100 * (1 - (sum {i in WIDTHS} i * orders[i]) / (roll_width * Number));
```
Pattern Generation

Results

ampl: include cutPatEnum.run

Gurobi 4.6.1: optimal solution; objective 164
15 simplex iterations

290 patterns, 164 rolls

<table>
<thead>
<tr>
<th>Cut</th>
<th>3 7 50 44 17 25 2 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>60.00</td>
<td>1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>30.00</td>
<td>0 3 0 0 0 0 0 0</td>
</tr>
<tr>
<td>25.50</td>
<td>0 0 1 1 0 0 0 0</td>
</tr>
<tr>
<td>20.00</td>
<td>0 0 0 0 3 0 0 0</td>
</tr>
<tr>
<td>17.25</td>
<td>0 0 3 2 0 2 0 0</td>
</tr>
<tr>
<td>15.00</td>
<td>2 0 0 2 2 2 0 0</td>
</tr>
<tr>
<td>12.75</td>
<td>0 0 1 0 0 2 7 0</td>
</tr>
<tr>
<td>10.00</td>
<td>0 0 0 0 0 0 0 9</td>
</tr>
</tbody>
</table>

WASTE = 0.32%
Pattern Generation

Data 2

```
param roll_width := 349 ;
param: WIDTHS: orders :=
   28.75    7
   33.75    23
   34.75    23
   37.75    31
   38.75    10
   39.75    39
   40.75    58
   41.75    47
   42.25    19
   44.75    13
   45.75    26 ;
```
## Pattern Generation

### Results 2

```plaintext
ampl: include cutPatEnum.run

Gurobi 4.6.1: optimal solution; objective 34
291 simplex iterations

54508 patterns, 34 rolls

<table>
<thead>
<tr>
<th>Cut</th>
<th>8</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>3</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>45.75</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>44.75</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42.25</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41.75</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40.75</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>39.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>38.75</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>37.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>34.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>33.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28.75</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

WASTE = 0.69%
```
Pattern Generation

Data 3

param roll_width := 172;
param: WIDTHS: orders :=
  25.000  5
  24.750  73
  18.000  14
  17.500  4
  15.500  23
  15.375  5
  13.875  29
  12.500  87
  12.250  9
  12.000  31
  10.250  6
  10.125  14
  10.000  43
  8.750  15
  8.500  21
  7.750  5;
### Pattern Generation

**Results 3 (using a subset of patterns)**

ampl: include cutPatEnum.run

Gurobi 4.6.1: optimal solution; objective 33  
722 simplex iterations  
40 branch-and-cut nodes  
273380 patterns, 33 rolls

<table>
<thead>
<tr>
<th>Cut</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.00</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24.75</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>17.50</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>.......</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>10.12</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8.75</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8.50</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7.75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

WASTE = 0.62%
Pattern Generation: Observations

Parameters can serve as script variables

- Declare as in model
  * param pattern {WIDTHS} integer >= 0;
- Use in algorithm
  * let pattern[curr_width] := pattern[curr_width] - 1;
- Assign to model parameters
  * let {w in WIDTHS} nbr[w,nPAT] := pattern[w];

Scripts are easy to modify

- Store only every 100th pattern found
  * if nPAT mod 100 = 0 then
    let {w in WIDTHS} nbr[w,nPAT/100] := pattern[w];
5: Decomposition

**Stochastic nonlinear location-transportation**

- Min expected total cost
  - Nonlinear construction costs at origins
  - Linear transportation costs from origins to destinations
- Stochastic demands with recourse
  - Decide what to build
  - Observe demands and decide what to ship

**Solve by Benders decomposition**

- Nonlinear master problem
- Linear subproblem for each scenario
Decomposition

Original model (sets, parameters, variables)

```
set WHSE;  # shipment origins (warehouses)
set STOR;  # shipment destinations (stores)

param build_cost {WHSE} > 0;  # costs per unit to build warehouse
param build_limit {WHSE} > 0;  # limits on units shipped

var Build {i in WHSE} >= 0, <= .9999 * build_limit[i];
    # capacities of warehouses to build

set SCEN;  # demand scenarios

param prob {SCEN} >= 0, <= 1;  # probabilities of scenarios
param demand {STOR,SCEN} >= 0;  # amounts required at stores

param ship_cost {WHSE,STOR} >= 0;  # shipment costs per unit

var Ship {WHSE,STOR,SCEN} >= 0;  # amounts to be shipped
```
Decomposition

Original model (objective, constraints)

minimize Total_Cost:
  sum {i in WHSE}
    build_cost[i] * Build[i] / (1 - Build[i]/build_limit[i]) +
  sum {s in SCEN} prob[s] *
    sum {i in WHSE, j in STOR} ship_cost[i,j] * Ship[i,j,s];

subj to Supply {i in WHSE, s in SCEN}:
  sum {j in STOR} Ship[i,j,s] <= Build[i];

subj to Demand {j in STOR, s in SCEN}:
  sum {i in WHSE} Ship[i,j,s] = demand[j,s];
Decomposition

Sub model (sets, parameters, variables)

```
set WHSE;    # shipment origins (warehouses)
set STOR;    # shipment destinations (stores)

param build {i in WHSE} >= 0, <= .9999 * build_limit[i];
    # capacities of warehouses built

set SCEN;    # demand scenarios

param prob {SCEN} >= 0, <= 1;   # probabilities of scenarios
param demand {STOR,SCEN} >= 0;  # amounts required at stores

param ship_cost {WHSE,STOR} >= 0;  # shipment costs per unit

var Ship {WHSE,STOR,SCEN} >= 0;   # amounts to be shipped
```
Decomposition

Sub model (objective, constraints)

```ampl
param S symbolic in SCEN;

minimize Scen_Ship_Cost:
    prob[S] * sum {i in WHSE, j in STOR} ship_cost[i,j] * Ship[i,j];

subj to Supply {i in WHSE}:
    sum {j in STOR} Ship[i,j] <= build[i];

subj to Demand {j in STOR}:
    sum {i in WHSE} Ship[i,j] = demand[j,S];
```
Decomposition

Master model (sets, parameters, variables)

```AMPL
param build_cost {WHSE} > 0; # costs per unit to build warehouse
param build_limit {WHSE} > 0; # limits on units shipped

var Build {i in WHSE} >= 0, <= .9999 * build_limit[i]; # capacities of warehouses to build

param nCUT >= 0 integer;
param cut_type {SCEN,1..nCUT} symbolic
  within {"feas","infeas","none"};

param supply_price {WHSE,SCEN,1..nCUT} <= 0.000001;
param demand_price {STOR,SCEN,1..nCUT};

var Max_Exp_Ship_Cost {SCEN} >= 0;
```
Decomposition

Master model (objective, constraints)

```
minimize Expected_Total_Cost:
    sum {i in WHSE}
        build_cost[i] * Build[i] / (1 - Build[i]/build_limit[i]) +
    sum {s in SCEN} Max_Exp_Ship_Cost[s];

subj to Cut_Defn {s in SCEN, k in 1..nCUT: cut_type[s,k] != "none"}:
    if cut_type[s,k] = "feas" then Max_Exp_Ship_Cost[s] else 0 >=
    sum {i in WHSE} supply_price[i,s,k] * Build[i] +
    sum {j in STOR} demand_price[j,s,k] * demand[j,s];
```
Decomposition

Script (initialization)

```AMPL
model stbenders.mod;
data stnltrnloc.dat;
suffix dnbdd;
option presolve 0;

  option solver cplex;
  option cplex_options 'primal presolve 0';
  option solver minos;

let nCUT := 0;
param GAP default Infinity;
param RELGAP default Infinity;
param Exp.Ship.Cost;
```
Decomposition

Script (iteration)

repeat {
    solve Master;
    let {i in WHSE} build[i] := Build[i];
    let Exp_Ship_Cost := 0;
    let nCUT := nCUT + 1;
    for {s in SCEN} {
        let S := s;
        solve Sub;
        ...
        generate a cut ...
    }
    if forall {s in SCEN} cut_type[s,nCUT] != "infeas" then {
        let GAP := min (GAP,
                        Exp_Ship_Cost - sum {s in SCEN} Max_Exp_Ship_Cost[s]);
        let RELGAP := 100 * GAP / Expected_Total_Cost;
    }
} until RELGAP <= .000001;
Decomposition

Script (cut generation)

```AMPL
for {s in SCEN} {
    let S := s;
solve Sub;

    if Sub.result = "solved" then {
        let Exp_Ship_Cost := Exp_Ship_Cost + Scen_Ship_Cost;
        if Scen_Ship_Cost > Max_Exp_Ship_Cost[s] + 0.00001 then {
            let cut_type[s,nCUT] := "feas";
            let {i in WHSE} supply_price[i,s,nCUT] := Supply[i].dual;
            let {j in STOR} demand_price[j,s,nCUT] := Demand[j].dual;
        }
        else let cut_type[s,nCUT] := "none";
    }
    else if Sub.result = "infeasible" then {
        let cut_type[s,nCUT] := "infeas";
        let {i in WHSE} supply_price[i,s,nCUT] := Supply[i].dunbdd;
        let {j in STOR} demand_price[j,s,nCUT] := Demand[j].dunbdd;
    }
}
```
## Decomposition

### Results

```ampl
ampl: include stbenders.run;

MASTER PROBLEM 1: 0.000000

SUB-PROBLEM 1 low: infeasible
SUB-PROBLEM 1 mid: infeasible
SUB-PROBLEM 1 high: infeasible

MASTER PROBLEM 2: 267806.267806

SUB-PROBLEM 2 low: 1235839.514234
SUB-PROBLEM 2 mid: 1030969.048921
SUB-PROBLEM 2 high: infeasible

MASTER PROBLEM 3: 718918.236014

SUB-PROBLEM 3 low: 1019699.661119
SUB-PROBLEM 3 mid: 802846.293052
SUB-PROBLEM 3 high: 695402.974379

GAP = 2517948.928551, RELGAP = 350.241349%
```
## Decomposition

### Results (continued)

<table>
<thead>
<tr>
<th>Master Problem</th>
<th>Objective Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2606868.719958</td>
</tr>
<tr>
<td>5</td>
<td>2685773.838398</td>
</tr>
<tr>
<td>6</td>
<td>2743483.001029</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub-Problem</th>
<th>Objective Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, Low</td>
<td>1044931.784272</td>
</tr>
<tr>
<td>4, Mid</td>
<td>885980.640150</td>
</tr>
<tr>
<td>4, High</td>
<td>944581.118758</td>
</tr>
<tr>
<td>5, Low</td>
<td>1028785.052062</td>
</tr>
<tr>
<td>5, Mid</td>
<td>815428.531237</td>
</tr>
<tr>
<td>5, High</td>
<td>753627.189086</td>
</tr>
<tr>
<td>6, Low</td>
<td>1000336.408156</td>
</tr>
<tr>
<td>6, Mid</td>
<td>785602.983289</td>
</tr>
<tr>
<td>6, High</td>
<td>725635.817601</td>
</tr>
</tbody>
</table>

GAP = 749765.716399, RELGAP = 28.761161%

GAP = 394642.837091, RELGAP = 14.693822%

GAP = 222288.965560, RELGAP = 8.102436%
Decomposition

Results (continued)

<table>
<thead>
<tr>
<th>MASTER PROBLEM 7: 2776187.713412</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-PROBLEM 7 low: 986337.500000</td>
</tr>
<tr>
<td>SUB-PROBLEM 7 mid: 777708.466300</td>
</tr>
<tr>
<td>SUB-PROBLEM 7 high: 693342.659287</td>
</tr>
<tr>
<td>GAP = 59240.084058, RELGAP = 2.133864%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASTER PROBLEM 8: 2799319.395374</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-PROBLEM 8 low: 991426.284976</td>
</tr>
<tr>
<td>SUB-PROBLEM 8 mid: 777146.351060</td>
</tr>
<tr>
<td>SUB-PROBLEM 8 high: 704353.854398</td>
</tr>
<tr>
<td>GAP = 38198.286498, RELGAP = 1.364556%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASTER PROBLEM 9: 2814772.778136</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-PROBLEM 9 low: 987556.309573</td>
</tr>
<tr>
<td>SUB-PROBLEM 9 mid: 772147.258329</td>
</tr>
<tr>
<td>SUB-PROBLEM 9 high: 696060.666966</td>
</tr>
<tr>
<td>GAP = 17658.226624, RELGAP = 0.627341%</td>
</tr>
</tbody>
</table>
Decomposition

Results (continued)

<table>
<thead>
<tr>
<th>MASTER PROBLEM 10: 2818991.649514</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-PROBLEM 10 mid: 771853.500000</td>
</tr>
<tr>
<td>SUB-PROBLEM 10 high: 689709.131427</td>
</tr>
<tr>
<td>GAP = 2361.940101, RELGAP = 0.083787%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASTER PROBLEM 11: 2819338.502316</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-PROBLEM 11 high: 692406.351318</td>
</tr>
<tr>
<td>GAP = 2361.940101, RELGAP = 0.083776%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASTER PROBLEM 12: 2819524.204253</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-PROBLEM 12 high: 690478.286312</td>
</tr>
<tr>
<td>GAP = 541.528304, RELGAP = 0.019206%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASTER PROBLEM 13: 2819736.994159</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAP = -0.000000, RELGAP = -0.000000%</td>
</tr>
</tbody>
</table>

OPTIMAL SOLUTION FOUND

Expected Cost = 2819736.994159
Decomposition: Observations

Loops can iterate over sets

- Solve a subproblem for each scenario
  - for {s in SCEN} { ... }

One model can represent all subproblems

- Assign loop index s to set S, then solve
  - let S := s;
  - solve Sub;

Related solution values can be returned

- Use dual ray to generate infeasibility cuts
  - if Sub.result = "infeasible" then { ... 
    let {i in WHSE}
    supply_price[i,s,nCUT] := Supply[i].dunbdc;
    let {j in STOR}
    demand_price[j,s,nCUT] := Demand[j].dunbdc; 
  }
Concluding Observations

Scripts in practice

- Large and complicated
  - Multiple files
  - Hundreds of statements
  - Millions of statements executed
- Run within broader applications

Prospective improvements

- Faster loops
- True script functions
  - Arguments and return values
  - Local sets & parameters
- More database connections
- IDE for debugging
- APIs for popular languages (C++, Java, C#, VB, Python)