New Interface Developments in the AMPL Modeling Language & System

Robert Fourer

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

EURO-INFORMS Joint International Meeting

Rome — 1-4 July 2013
Track TB-56, *Optimization Modeling I*

Outline

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

Logical Conditions

Common "not linear" expressions

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

Variety of solvers

- IBM CPLEX mixed-integer solver
  - Applied directly
  - Applied after conversion to MIP
- Constraint solvers
  - IBM ILOG CP
  - Gecode
  - JaCoP

Example: Multi-Commodity

Minimum-shipment constraints

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

Conventional linear mixed-integer formulation

```
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`": `Sat[i1,i2] = 1` iff this is satisfied in the lineup
- `Pos[i]` is the position of person `i` in the line

. . . fewer variables, larger domains

"CP-Style" Alternative

All-different constraint

```
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];
```

"CP-Style" Alternative *(cont'd)*

11 people, 20 preferences

```
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:
```

Quadratic Constraints

Given a traffic network

N Set of nodes representing intersections

e Entrance to network

f Exit from network

$A \subseteq N \cup \{e\} \times N \cup \{f\}$

Set of arcs representing road links

with associated data

b_{ij} Base travel time for each road link $(i,j) \in A$

s_{ij} Traffic sensitivity for each road link $(i,j) \in A$

c_{ij} Capacity for each road link $(i,j) \in A$

T Desired throughput from e to f

Formulation

Determine

x_{ij} Traffic flow through road link $(i,j) \in A$

t_{ij} Actual travel time on road link $(i,j) \in A$

to minimize

$\Sigma_{(i,j)\in A}\, t_{ij} x_{ij} / T$

Average travel time from e to f

Formulation *(cont'd)*

Subject to

$$t_{ij} = b_{ij} + \frac{s_{ij}x_{ij}}{1 - x_{ij}/c_{ij}} \quad \text{for all } (i,j) \in A$$

Travel times increase as flow approaches capacity

$$\Sigma_{(i,j)\in A}\, x_{ij} = \Sigma_{(j,i)\in A}\, x_{ji} \quad \text{for all } i \in N$$

Flow out equals flow in at any intersection

$$\Sigma_{(e,j)\in A}\, x_{ej} = T$$

Flow into the entrance equals the specified throughput

AMPL Formulation

Symbolic data

```
set INTERS;           # intersections (network nodes)

param EN symbolic;    # entrance
param EX symbolic;    # exit

   check {EN,EX} not within INTERS;

set ROADS within {INTERS union {EN}} cross {INTERS union {EX}};

                          # road links (network arcs)

param base {ROADS} > 0;   # base travel times
param sens {ROADS} > 0;   # traffic sensitivities
param cap {ROADS} > 0;    # capacities

param through > 0;        # throughput
```

AMPL Formulation *(cont'd)*

Symbolic model

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Time {ROADS} >= 0;

minimize Avg_Time:
  (sum {(i,j) in ROADS} Time[i,j] * Flow[i,j]) / through;

subject to Travel_Time {(i,j) in ROADS}:
  Time[i,j] = base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j]);

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

Traffic Network

AMPL Data

Explicit data independent of symbolic model

```
set INTERS := b c ;

param EN := a ;
param EX := d ;

param: ROADS: base cap sens :=
         a b    4   10   .1
         a c    1   12   .7
         c b    2   20   .9
         b d    1   15   .5
         c d    6   10   .1 ;

param through := 20 ;
```

AMPL Solution

Model + data = problem to solve, using KNITRO

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

ampl: option solver knitro;
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective 61.04695019; feasibility error 3.55e-14
12 iterations; 25 function evaluations

ampl: display Flow, Time;

:        Flow        Time    :=
a b     9.55146    25.2948
a c    10.4485     57.5709
b d    11.0044     21.6558
c b     1.45291     3.41006
c d     8.99562    14.9564
;
```

AMPL Solution *(cont'd)*

Same with integer-valued variables

```
var Flow {(i,j) in ROADS} integer >= 0, <= .9999 * cap[i,j];
```

```
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective 76.26375; integrality gap 0
3 nodes; 5 subproblem solves

ampl: display Flow, Time;

:      Flow    Time   :=
a b     9      13
a c    11      93.4
b d    11      21.625
c b     2       4
c d     9      15
;
```

AMPL Solution *(cont'd)*

Model + data = problem to solve, using Gurobi?

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 5.5.0:
Gurobi can't handle nonquadratic nonlinear constraints.
```

AMPL Solution *(cont'd)*

Look at the model again . . .

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Time {ROADS} >= 0;

minimize Avg_Time:
  (sum {(i,j) in ROADS} Time[i,j] * Flow[i,j]) / through;

subject to Travel_Time {(i,j) in ROADS}:
  Time[i,j] = base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j]);

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Solution *(cont'd)*

Quadratically constrained reformulation

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Delay {ROADS} >= 0;

minimize Avg_Time:
  sum {(i,j) in ROADS} (base[i,j]*Flow[i,j] + Delay[i,j]) / through;

subject to Delay_Def {(i,j) in ROADS}:
  sens[i,j] * Flow[i,j]^2 <= (1 - Flow[i,j]/cap[i,j]) * Delay[i,j];

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Solution *(cont'd)*

Model + data = problem to solve, using Gurobi?

```
ampl: model trafficQUAD.mod;
ampl: data traffic.dat;

ampl: option solver gurobi;
ampl: solve;

Gurobi 5.5.0:
quadratic constraint is not positive definite
```

AMPL Solution *(cont'd)*

Simple conic quadratic reformulation

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Delay {ROADS} >= 0;
var Slack {ROADS} >= 0;

minimize Avg_Time:
  sum {(i,j) in ROADS} (base[i,j]*Flow[i,j] + Delay[i,j]) / through;

subject to Delay_Def {(i,j) in ROADS}:
  sens[i,j] * Flow[i,j]^2 <= Slack[i,j] * Delay[i,j];

subject to Slack_Def {(i,j) in ROADS}:
  Slack[i,j] = 1 - Flow[i,j]/cap[i,j];

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Solution *(cont'd)*

Model + data = problem to solve, using Gurobi!

```
ampl: model trafficSOC.mod;
ampl: data traffic.dat;

ampl: option solver gurobi;
ampl: solve;

Gurobi 5.5.0: optimal solution; objective 61.04696953
47 barrier iterations

ampl: display Flow;

Flow :=
a b    9.55146
a c   10.4485
b d   11.0031
c b    1.45167
c d    8.99687
;
```

AMPL Solution *(cont'd)*

Same with integer-valued variables

```
var Flow {(i,j) in ROADS} integer >= 0, <= .9999 * cap[i,j];
```

```
ampl: solve;

Gurobi 5.5.0: optimal solution; objective 76.26374998
32 simplex iterations

ampl: display Flow;

Flow :=
a b    9
a c   11
b d   11
c b    2
c d    9
;
```

Processing of Convex Quadratics

Problem types

- Elliptical: quadratic programs (QPs)
- Conic: second-order cone programs (SOCPs)

What AMPL may do

- Recognize quadratic objectives & constraints
- Multiply out products of linear terms
- Send linear & quadratic coefficient lists to solver

What the solver may do

- Detect elliptical forms numerically
- Detect conic forms by structural analysis

. . . analysis could be stronger if done by AMPL

. . . more in session WB-9 tomorrow, room O3-3

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

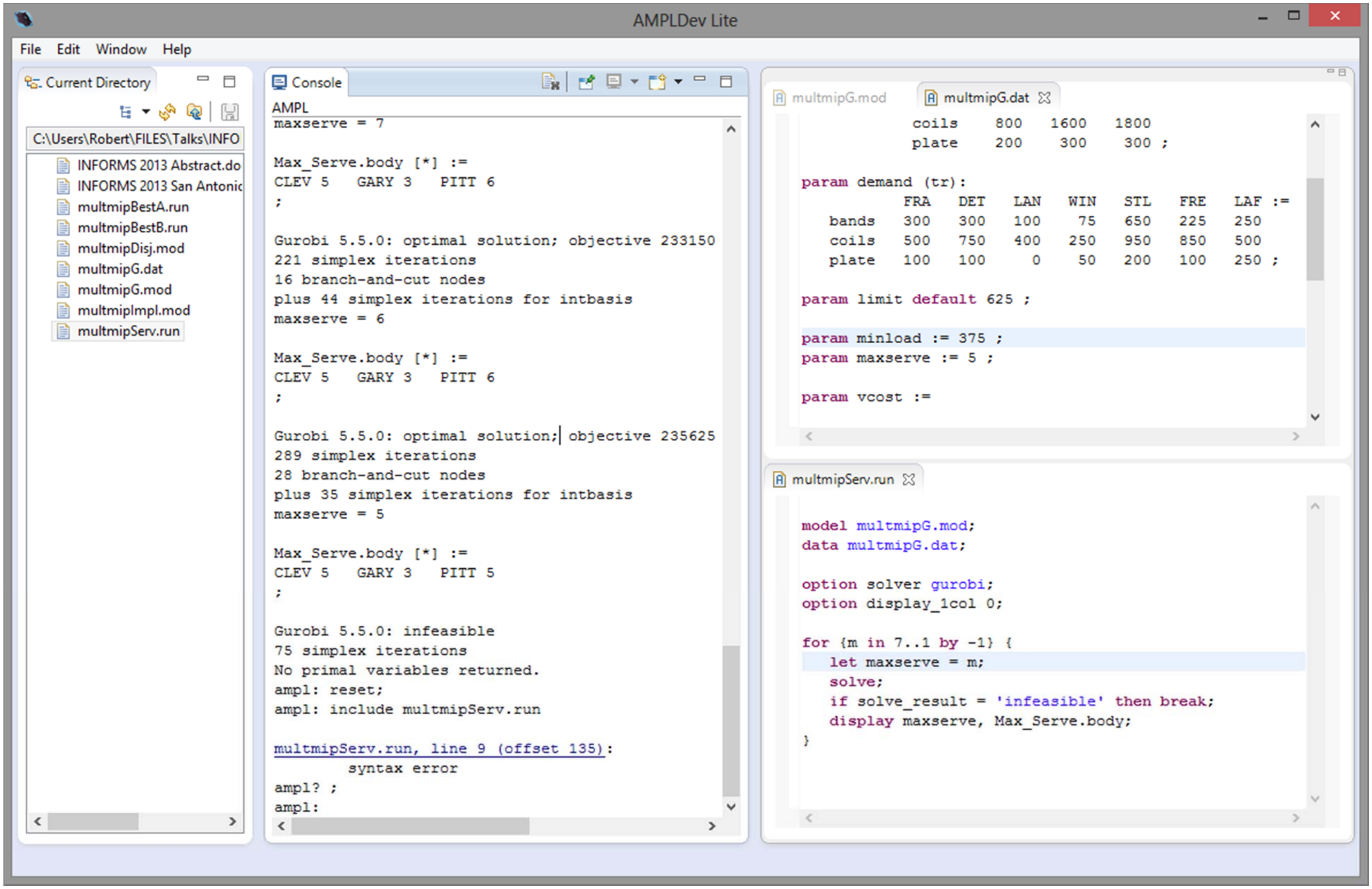
Initial release

- ❖ Simplified for easy transition
- ❖ Works with existing installations

Beta test version available very soon . . .

AMPL IDE

Sample Screenshot



Planned Availability

Rollout dates

- Beta test this summer
  - Actively seeking testers now
  - ***Instructions at www.ampl.com/IDE/beta.html***
- Release
  - Fall 2013
  - Included in all AMPL distributions

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

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

Example: Java

Efficient frontier: Initialize, read files

```
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

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

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

```
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 stockopall:={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();
}
```

Example: MATLAB

Efficient frontier: Initialize, read files

```
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.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

```
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)
```

Planned Availability

Rollout dates

- Beta test (Java, MATLAB, . . .)
  - End of summer 2013
  - *Seeking beta testers now*
- Release
  - End of 2013
  - Included in all AMPL distributions

Development details

- Partnership with OptiRisk Systems
- At least 6 languages to be provided

Readings *(AMPL)*

- R. Fourer, "Modeling Languages versus Matrix Generators for Linear Programming." *ACM Transactions on Mathematical Software* **9** (1983) 143–183.
- R. Fourer, D.M. Gay, B.W. Kernighan, "A Modeling Language for Mathematical Programming." *Management Science* **36** (1990) 519–554.
- R. Fourer, D.M. Gay, B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Belmont, CA (first edition 1993, second edition 2003).
- R. Fourer, "Algebraic Modeling Languages for Optimization." Forthcoming in Saul I. Gass and Michael C. Fu (eds.), *Encyclopedia of Operations Research and Management Science,* Springer (2012).
- Robert Fourer, On the Evolution of Optimization Modeling Systems. M. Groetschel (ed.), *Optimization Stories.* Documenta Mathematica (2012) 377-388.

Readings *(Interfaces)*

- G. Everett, A. Philpott, K. Vatn, R. Gjessing, "Norske Skog Improves Global Profitability Using Operations Research." *Interfaces* **40,** 1 (Jan–Feb 2010) 58–70.
- F. Caro, J. Gallien, M. Díaz, J. García, J.M. Corredoira, M. Montes, J.A. Ramos, J. Correa, "Zara Uses Operations Research to Reengineer Its Global Distribution Process." *Interfaces* **40,** 1 (Jan–Feb 2010) 71–84.
- P. Pekgün, R.P. Menich, S. Acharya, P.G. Finch, F. Deschamps, K. Mallery, J. Van Sistine, K. Christianson, J. Fuller, "Carlson Rezidor Hotel Group Maximizes Revenue Through Improved Demand Management and Price Optimization." *Interfaces* **43**, 1 (Jan–Feb 2013) 21–36.