Using KNITRO for AMPL

Robert Fourer

AMPL Optimization LLC
Ziena Optimization LLC

Industrial Engineering & Management Sciences
Northwestern University

INFORMS Conference on Business Analytics & Operations Research
Chicago — April 10-12, 2011 — Ziena Workshop
Example: Traffic Network

Given

\( N \) 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

and

\( b_{ij} \) Base travel time for each road link \((i,j)\) \(\in A\)
\( c_{ij} \) Capacity for each road link \((i,j)\) \(\in A\)
\( s_{ij} \) Traffic sensitivity for each road link \((i,j)\) \(\in A\)
\( T \) Desired throughput from \(e\) to \(f\)
Example: Traffic Network

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

\[ \sum_{(i,j) \in A} t_{ij} x_{ij} / T \]

Average travel time from \( e \) to \( f \)
Example: Traffic Network

Subject to

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

Travel times increase as flow approaches capacity

\[ \sum_{(i,j) \in A} x_{ij} = \sum_{(j,i) \in A} x_{ji} \text{ for all } i \in N \]

Flow out equals flow in at any intersection

\[ \sum_{(e,j) \in A} x_{ej} = T \]

Flow into the entrance equals the specified throughput
AMPL Traffic Network

Traffic network: *symbolic data*

```AMPL
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 cap {ROADS} > 0;   # capacities
param sens {ROADS} > 0;  # traffic sensitivities
param through > 0;      # throughput
```

Robert Fourer, Using KNITRO for AMPL
INFORMS Business Analytics Conference—April 10-12, 2011 — Ziena Workshop
AMPL Traffic Network

Algebraic modeling language: *symbolic model*

\[
\begin{align*}
\text{var } & \text{Flow } \{(i,j) \text{ in ROADS}\} \geq 0, \leq 0.9999 \times \text{cap}[i,j]; \\
\text{var } & \text{Time } \{\text{ROADS}\} \geq 0; \\
\text{minimize } & \text{Avg\_Time:} \\
& \left(\sum \{ (i,j) \text{ in ROADS}\} \text{Time}[i,j] \times \text{Flow}[i,j]\right) / \text{through}; \\
\text{subject to } & \text{Travel\_Time } \{(i,j) \text{ in ROADS}\}: \\
& \text{Time}[i,j] = \text{base}[i,j] + (\text{sens}[i,j] \times \text{Flow}[i,j]) / (1 - \text{Flow}[i,j] / \text{cap}[i,j]); \\
\text{subject to } & \text{Balance\_Node } \{i \text{ in INTERS}\}: \\
& \sum \{ (i,j) \text{ in ROADS}\} \text{Flow}[i,j] = \sum \{ (j,i) \text{ in ROADS}\} \text{Flow}[j,i]; \\
\text{subject to } & \text{Balance\_Enter:} \\
& \sum \{ (EN,j) \text{ in ROADS}\} \text{Flow}[EN,j] = \text{through};
\end{align*}
\]
AMPL Traffic Network

Explicit data independent of symbolic model

```
set INTERS := b c ;
param EN := a;
param EX := d;
param: ROADS: base cap sens :=
    a b  5  10 .1
    a c  1  30 .9
    c b  2  10 .9
    b d  1  30 .9
    c d  5  10 .1 ;
param through := 4;
```
AMPL Traffic Network

Model + data = problem to solve

```ampl
ampl: model traffic_c.mod;
ampl: data traffic_c.dat;
ampl: option solver knitroampl;
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective 8.178571439; feasibility error 2.23e-12
7 iterations; 8 function evaluations

ampl: display Flow, Time;
: Flow Time :=
 a b  2             5.25
 a c  2             2.92857
 b d  2             2.92857
 c b 2.79497e-08   2
 c d  2             5.25
;
```
Example: Portfolio Management

**Standard Markowitz quadratic model**

- General fractional shares
- Discrete fractional shares
- Discrete investment rules
  - min/max share
  - diversification
AMPL Portfolio Management

Symbolic model

```
set A;             # asset categories
set T := {1973..1994};    # years
param R {T,A};         # returns on asset categories
param mu default 2;    # weight on variance

param mean {j in A} = (sum {i in T} R[i,j]) / card(T);
param Rtilde {i in T, j in A} = R[i,j] - mean[j];

var Frac {A} >=0;
var Mean = sum {j in A} mean[j] * Frac[j];
var Variance =
    sum {i in T} (sum {j in A} Rtilde[i,j]*Frac[j])^2 / card{T};

minimize RiskReward:  mu * Variance - Mean;
subject to TotalOne:  sum {j in A} Frac[j] = 1;
```
AMPL Portfolio Management

Example of data

set A :=
    US_3-MONTH_T-BILLS US_GOVN_LONG_BONDS SP_500 WILSHIRE_5000
    NASDAQ_COMPOSITE CORPORATE_BONDS_INDEX EAFE GOLD;

param R:
    US_3-MONTH_T-BILLS US_GOVN_LONG_BONDS SP_500 WILSHIRE_5000
    NASDAQ_COMPOSITE CORPORATE_BONDS_INDEX EAFE GOLD :=

    1973  1.075  0.942  0.852  0.815  0.698  1.023  0.851  1.677
    1974  1.084  1.020  0.735  0.716  0.662  1.002  0.768  1.722
    1975  1.061  1.056  1.371  1.385  1.318  1.123  1.354  0.760
    1976  1.052  1.175  1.236  1.266  1.280  1.156  1.025  0.960
    1977  1.055  1.002  0.926  0.974  1.093  1.030  1.181  1.200
    1978  1.077  0.982  1.064  1.093  1.146  1.012  1.326  1.295
    1979  1.109  0.978  1.184  1.256  1.307  1.023  1.048  2.212
    1980  1.127  0.947  1.323  1.337  1.367  1.031  1.226  1.296
    1981  1.156  1.003  0.949  0.963  0.990  1.073  0.977  0.688
    1982  1.117  1.465  1.215  1.187  1.213  1.311  0.981  1.084
    1983  1.092  0.985  1.224  1.235  1.217  1.080  1.237  0.872
    1984  1.103  1.159  1.061  1.030  0.903  1.150  1.074  0.825 ...
AMPL Portfolio Management

Solving with KNITRO

```ampl
ampl: model markowitz.mod;
ampl: data markowitz.dat;
ampl: option solver knitroampl;
ampl: option knitro_options 'opttol 1e-12';
ampl: solve;
8 variables, all nonlinear
1 constraint, all linear; 8 nonzeros
1 nonlinear objective; 8 nonzeros.
KNITRO 7.0.0: Locally optimal solution.
objective -1.098362476; feasibility error 1.11e-16
10 iterations; 11 function evaluations
ampl:
```
AMPL Portfolio Management

Optimal portfolio

```ampl
ampl: option omit_zero_rows 1;
ampl: option display_eps .000001;
ampl: display Frac;
CORPORATE_BONDS_INDEX 0.397056
    EAFE 0.216083
    GOLD 0.185066
    WILSHIRE_5000 0.201795;
ampl: display Mean, Variance;
Mean = 1.11577
Variance = 0.00870377
ampl:
```
AMPL Portfolio Management

Solving with KNITRO: *Discrete fractions*

```ampl
var Share {A} integer >= 0, <= 20;
var Frac {j in A} = Share[j] / 20;
```

```ampl
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective -1.098266447; integrality gap -6.13e-13
13 nodes; 13 subproblem solves
```

```ampl
ampl: display Frac;

CORPORATE_BONDS_INDEX  0.4
                      EAFE   0.2
                      GOLD   0.2
                     WILSHIRE_5000  0.2
```

AMPL Portfolio Management

Solving with KNITRO: Investment rules

```plaintext
param leastUse = 5;
param leastFrac = .10;
param mostFrac = .35;

var Use {A} binary;

subject to UseDefn {j in A}:
    Frac[j] <= mostFrac * Use[j];  # upper limit on fraction of asset

subject to LeastFrac {j in A}:
    Frac[j] >= leastFrac * Use[j];  # lower limit on fraction of asset
    # if the asset is used at all

subject to LeastUse:
    sum {j in A} Use[j] >= leastUse;  # lower limit on number of assets
```
AMPL Portfolio Management

Solving with KNITRO: Investment rules (cont’d)

```ampl
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective -1.098228448; integrality gap -1.98e-14
5 nodes; 5 subproblem solves

ampl: display Frac;

CORPORATE_BONDS_INDEX  0.35
    EAFE       0.2
    GOLD       0.2
    SP_500     0.1
    WILSHIRE_5000  0.15
```
How KNITRO Interacts with AMPL

*User types...*

    option solver knitroampl;
    option knitro_options 'opttol 1e-12';
    solve;

*AMPL...*

    Writes at13151.nl
    Executes knitroampl at13151 -AMPL

*KNITRO “driver”...*

    Reads at13151.nl
    Gets environment variable knitro_options
    Calls KNITRO routines to solve the problem
    Writes at13151.sol

*AMPL...*

    Reads at13151.sol
What the KNITRO Driver Does

**Reads .nl problem file**

- Loads everything into ASL data structure
- Copies linear coefficients, bounds, etc. to solver’s arrays
- Sets directives indicated by _options string

**Runs algorithm**

- Uses ASL data structure to compute nonlinear expression values, 1st & 2nd derivatives

**Writes .sol solution file**

- Generates result message
- Writes values of variables
AMPL’s .nl File Format

File contents

Numbers of variables, constraints, integer variables, nonlinear constraints, etc.

Coefficient lists for linear part

Expression tree for nonlinear part plus sparsity pattern of derivatives

Expression tree nodes

Variables, constants
Binary, unary operators
Summations
Functions
Variables
Constants
Example of .nl File

Header

g3  0  1  0  # problem sum-of-norms3
14 11  1  0  9  # vars, constraints, objectives, ranges, eqns
 2  0  # nonlinear constraints, objectives
 0  0  # network constraints: nonlinear, linear
11  0  0  # nonlinear vars in constraints, objectives, both
 0  0  0  1  # linear network variables; functions; arith, flags
 0  0  0  0  0  # discrete variables: binary, integer, nonlinear (b,c,o)
 41  2  # nonzeros in Jacobian, gradients
 0  0  # max name lengths: constraints, variables
 0  0  0  0  0  # common exprs: b,c,o,c1,o1
Example of .nl File

Expression trees for nonlinear constraints

```
C0  #MaxDefinition[1]
o5  #sumlist
  6
  o5  #^    v2  #Fxplusg[1,1]
n2
  o5  #^    v3  #Fxplusg[1,2]
n2
  o5  #^    v4  #Fxplusg[1,3]
n2
  o5  #^    v5  #Fxplusg[1,4]
n2
  o5  #^    v6  #Fxplusg[1,5]
n2
  o16 #-    o5  #^    v0  #Max[1]
n2
C1  #MaxDefinition[2] ... 
```

```
subj to MaxDefinition {i in 1..p}:
  sum {k in 1..m[i]} Fxplusg[i,k]^2 <= Max[i]^2;
```
How AMPL Computes Derivatives

“Backward” Automatic Differentiation

- Forward sweep: compute $f(x)$, save info on $\frac{\partial f(x)}{\partial o}$ for each operation $o$
- Backward sweep: recur to compute $\nabla f(x)$

Complexity

- Small multiple of time for $f(x)$ alone
- Potentially large multiple of space

Advantages

- More accurate, efficient than finite differencing
- $O(n)$ vs. $O(n^2)$ for symbolic differentiation or forward AD
2nd Derivative (Hessian) Options

**Hessian-vector products: \( \nabla^2 f(x) \nu \)**

- Apply backward AD to compute gradients of \( \nu^T \nabla f(x) \)
- Equivalently, compute \( \nabla_x (df(x + \tau \nu) / d\tau \mid _{\tau = 0}) \)

**General case**

- \( \nabla^2 f(x) e_j \) for each \( j = 1, \ldots, n \)

**Partially separable case**

- \( f(x) = \sum_{t=1}^q f_t(U_t x) \) where \( U_t \) is \( m_t \times n \), \( m_t \gg n \)
- \( \nabla f(x) = \sum_{t=1}^q U_t^T \nabla f_t(U_t x) \)
- \( \nabla^2 f(x) = \sum_{t=1}^q U_t^T \nabla^2 f_t(U_t x) U_t \), a sum of outer products
How AMPL Computes Hessians

Detect partially separable structure
- Walk expression tree
- Use a hashing scheme to spot common subexpressions
  ... sometimes useful in itself

Compute derivative information
- General or partially separable computations
- Dense or sparse
- Full or lower triangle
  ... using general and/or partially separable approach

Further complications
- Hessian of Lagrangian
- Defined variables
KNITRO Derivative Options

**Gradient (1\textsuperscript{st} derivatives) — gradopt**
- 1: use exact gradients
- 2: compute forward finite-difference approximations
- 3: compute centered finite-difference approximations

**Hessian (2\textsuperscript{nd} derivatives) — hessopt**
- 1: use exact Hessian derivatives
- 2: use dense quasi-Newton BFGS Hessian approximation
- 3: use dense quasi-Newton SR1 Hessian approximation
- 4: compute Hessian-vector products by finite differences
- 5: compute exact Hessian-vector products
- 6: use limited-memory BFGS Hessian approximation

**Tradeoffs**
- More information: Reduction in iterations
- Less information: Reduction in work per iteration
Multiple Solutions

Best $n$ binary solutions
  ✔ AMPL scripting
  ✔ Modeling flexibility

Multiple starts for nonconvex problems
  ✔ Set starts in AMPL
  ✔ Generate starts automatically in KNITRO
AMPL Portfolio Management

Solving with KNITRO: 10 best portfolios

```AMPL
param nSols default 0;
param maxSols = 10;
set U {1..nSols} within A;
subject to exclude {k in 1..nSols}:
    sum {j in U[k]} (1-Use[j]) + sum {j in A diff U[k]} Use[j] >= 1;
repeat {
    solve;
    display Frac;
    let nSols := nSols + 1;
    let U[nSols] := {j in A: Use[j] > .5};
} until nSols = maxSols;
```
AMPL Portfolio Management

Solving with KNITRO: 10 best portfolios (cont’d)

```
ampl: include portfolios.run;

KNITRO 7.0.0: Locally optimal solution.
objective -1.098228448; integrality gap -1.98e-14
5 nodes; 5 subproblem solves
CORPORATE_BONDS_INDEX 0.35
     EAFE 0.2
     GOLD 0.2
     SP_500 0.1
     WILSHIRE_5000 0.15 ;

KNITRO 7.0.0: Locally optimal solution.
objective -1.097809549; integrality gap -1.88e-12
13 nodes; 13 subproblem solves
CORPORATE_BONDS_INDEX 0.35
     EAFE 0.2
     GOLD 0.15
     US_3-MONTH_T-BILLS 0.1
     WILSHIRE_5000 0.2 ;
```
AMPL Portfolio Management

Solving with KNITRO: 10 best portfolios (cont’d)

KNITRO 7.0.0: Locally optimal solution.
objective \(-1.097743771\); integrality gap \(3.19e-07\)
23 nodes; 23 subproblem solves

```
CORPORATE_BONDS_INDEX 0.25
   EAFE 0.2
   GOLD 0.2
   SP_500 0.1
   US_3-MONTH_T-BILLS 0.1
   WILSHIRE_5000 0.15;
```

KNITRO 7.0.0: Locally optimal solution.
objective \(-1.097696799\); integrality gap \(-2.37e-13\)
45 nodes; 45 subproblem solves

```
CORPORATE_BONDS_INDEX 0.25
   EAFE 0.2
   GOLD 0.2
   SP_500 0.25
   US_3-MONTH_T-BILLS 0.1;
```
Multiple Starting Points

Nonconvex transportation problem

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

param supply {ORIG} >= 0;      # amounts available at origins
param demand {DEST} >= 0;      # amounts required at destinations
param rate {ORIG,DEST} >= 0;   # base shipment costs per unit
param limit {ORIG,DEST} > 0;   # limit on units shipped

var Trans {i in ORIG, j in DEST} >= 1e-10, <= .9999 * limit[i,j];        
    # actual units to be shipped

minimize Total_Cost:
    sum {i in ORIG, j in DEST}
        rate[i,j] * Trans[i,j]^0.8 / (1 - Trans[i,j]/limit[i,j]);

subject to Supply {i in ORIG}: sum {j in DEST} Trans[i,j] = supply[i];
subject to Demand {j in DEST}: sum {i in ORIG} Trans[i,j] = demand[j];
```
Multiple Starting Points

Set starts in AMPL

```AMPL
for {init in 1..9} {
    let {i in ORIG, j in DEST} Trans[i,j] := Uniform01() * limit[i,j];
    solve;
}
```

KNITRO 7.0.0: Locally optimal solution.
objective 379000.7333; feasibility error 6.82e-13
26 iterations; 27 function evaluations

KNITRO 7.0.0: Locally optimal solution.
objective 370807.9426; feasibility error 0
47 iterations; 73 function evaluations

KNITRO 7.0.0: Locally optimal solution.
objective 356531.4679; feasibility error 0
79 iterations; 103 function evaluations

.......

Multiple Starting Points

Generate starts automatically in KNITRO

```ampl
ampl: option knitro_options 'alg=3 ms_enable=1 ms_maxsolves=25';
ampl: solve;

KNITRO 7.0.0: Locally optimal solution.
objective 354276.7169; feasibility error 2.27e-13
1013 iterations; 1448 function evaluations
```