Convexity Detection in Large-Scale Optimization

Robert Fourer

Industrial Engineering & Management Sciences
Northwestern University

AMPL Optimization

4er@northwestern.edu — 4er@ampl.com

Very Large-Scale Optimization
A Conference in Honor of Etienne Loute
Facultés Universitaires Saint-Louis, Brussels — 6 September 2011

Given an Optimization Model . . .

Does it have nice properties?

- Is it convex?
- Is it equivalent to a convex quadratic?

Does knowing that help to solve the problem?

- Are the results more believable?
- Are the computations more reliable?
- Are the computations more efficient?

Ways to Answer These Questions

Thought

- Theorems
- Equivalent formulations

Computation

- Detection algorithms
- Transformation algorithms
- Faster and more reliable
- Intractable in general
- Challenging in concept
- Challenging to implement

Example: Traffic Network

Nonlinear objective due to congestion effects

```
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;
```

Example: AC Power Network

Sines & cosines due to Kirchhoff's laws for AC

```
var G{(k,m) in YBUS} =
  if(k == m) then ...
else if(k != m) then
  sum {(l,k,m) in BRANCH}
    (- branch_g[l,k,m] * cos(branch_def[l,k,m])
     - branch_b[l,k,m] * sin(branch_def[l,k,m])) * branch_tap[l,k,m] +
  sum {(l,m,k) in BRANCH}
    (- branch_g[l,m,k] * cos(branch_def[l,m,k])
     + branch_b[l,m,k] * sin(branch_def[l,m,k])) * branch_tap[l,m,k];

minimize active_power :
  sum {k in BUS : bus_type[k] == 2 || bus_type[k] == 3}
   (bus_p_load[k]
    + sum {(k,m) in YBUS}
       bus_voltage[k] * bus_voltage[m]
         * (  G[k,m] * cos(bus_angle[k] - bus_angle[m])
            + B[k,m] * sin(bus_angle[k] - bus_angle[m])) )^2;
```

Outline

Recursive tree-walking algorithms

- Expression trees
- Detection algorithms
- Transformation algorithms

Convexity of general expressions

- Proof of convexity
- Disproof of convexity

Convexity of quadratic problems

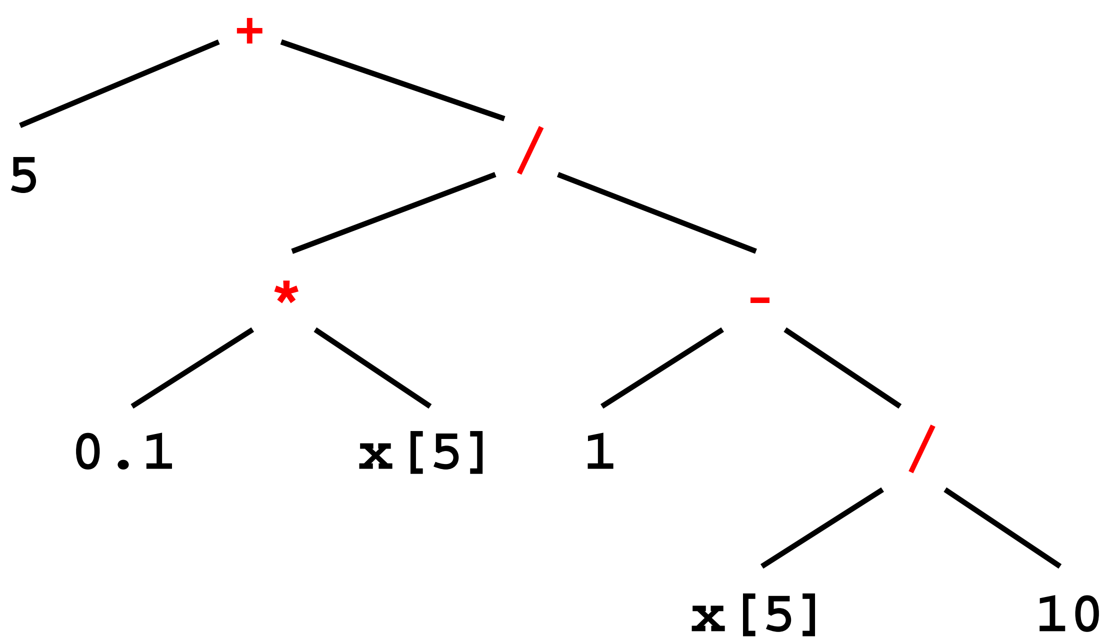
- Conic constraints
- Detection of equivalent problems
- Transformation of equivalent problems

Recursive Tree-Walking Algorithms

Expression

```
base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j])
```

Expression tree



. . . actually a DAG

Detection: `isLinear()`

```
boolean isLinear (Node);

case of Node {

 PLUS:
 MINUS: return( isLinear(Node.left) and isLinear(Node.right) );

 TIMES: return( isConst(Node.left) and isLinear(Node.right) or
                isLinear(Node.left) and isConst(Node.right) );

 DIV:   return( isLinear(Node.left) and isConst(Node.right) );

 VAR:   return( TRUE );

 CONST: return( TRUE );

}
```

. . . to detect, test `isLinear(root)`

Detection: `isQuadr()`

```
boolean isQuadr (Node);

case of Node {

 PLUS:
 MINUS: return( isQuadr(Node.left) and isQuadr(Node.right) );

 TIMES: return( isLinear(Node.left) and isLinear(Node.right) or
                isQuadr(Node.left) and isConst(Node.right)   or
                isConst(Node.left) and isQuadr(Node.right) );

 POWER: return( isLinear(Node.left) and
                isConst(Node.right) and value(Node.right) == 2 );

 VAR:   return( TRUE );

 CONST: return( TRUE );

}
```

Transformation: `buildLinear()`

```
(coeff,const) = buildLinear (Node);

if Node.L then (coefL,consL) = buildLinear(Node.L);
if Node.R then (coefR,consR) = buildLinear(Node.R);

case of Node {

 PLUS:  coeff = mergeLists( coefL, coefR );
        const = consL + consR;

 TIMES: ...

 DIV:   coeff = coefL / consR;
        const = consL / consR;

 VAR:   coeff = makeList( 1, Node.index );
        const = 0;

 CONST: coeff = makeList( );
        const = Node.value;

}
```

. . . to transform, call `buildLinear(root)`

Convexity of General Expressions

Analysis

- Proof of convexity
- Disproof of convexity

Larger context ("DrAMPL")

- Classify problems based on AMPL output
- Recommend solvers

. . . joint project with Dominique Orban,
École Polytechnique de Montréal

Significance of Convexity

Properties

- For an optimization problem of the form

$$\boxed{\begin{array}{ll} \text{Minimize} & f(x_1,\ldots,x_n) \\ \text{Subject to} & g_i(x_1,\ldots,x_n) \geq 0, \quad i=1,\ldots,r \\ & h_i(x_1,\ldots,x_n) = 0, \quad i=1,\ldots,s \end{array}}$$

  a local minimum is global provided

  - ⋆ f is convex
  - ⋆ each g_i is convex
  - ⋆ each h_i is linear

- Many physical problems are naturally convex if formulated properly

Analyses . . .

- Proof of convexity
- Disproof of convexity

Interest in Convexity Detection

Earlier approaches

- D.R. Stoutmeyer, "Automatic categorization of optimization problems: An application of computer symbolic mathematics." *Operations Research* **26** (1978) 773–788.
- I.P. Nenov, D.H. Fylstra and L.V. Kolev, "Convexity determination in the Microsoft Excel solver using automatic differentiation techniques." Fourth International Workshop on Automatic Differentiation (2004).
- M.C. Grant, S. Boyd and Y. Ye "Disciplined convex programming." In L. Liberti, N. Maculan, eds. *Global Optimization: From Theory to Implementation.* Springer, Nonconvex Optimization and Its Applications Series (2006) 155–210.

This work

- R. Fourer, C. Maheshwari, A. Neumaier, D. Orban and H. Schichl, "Convexity and Concavity Detection in Computational Graphs: Tree Walks for Convexity Assessment." dx.doi.org/10.1287/ijoc.1090.0321: *INFORMS Journal on Computing* **22** (2010) 26–43.

Proof of Convexity

Apply properties of functions

- $\|\mathbf{x}\|_p$ is convex, ≥ 0 everywhere
- x^α is convex for $\alpha \leq 0$, $\alpha \geq 1$; $-x^\alpha$ is convex for $0 \leq \alpha \leq 1$
- x^p for even $p > 0$ is convex everywhere,
  decreasing on $x \leq 0$, increasing on $x \geq 0$, *etc.*
- $-\log x$ and $x \log x$ are convex and increasing on $x > 0$
- $\sin x$ is concave on $0 \leq x \leq \pi$, convex on $\pi \leq x \leq 2\pi$,
  increasing on $0 \leq x \leq \pi/2$ and $3\pi/2 \leq x \leq 2\pi$, decreasing . . .
  ≥ -1 and ≤ 1 everywhere
- $e^{\alpha x}$ is convex, increasing everywhere for $\alpha > 0$, *etc.*
- $-(\Pi_i\, x_i)^{1/n}$ is convex where all $x_i > 0$

. . . ***etc., etc.***

Proof of Convexity *(cont'd)*

Apply properties of convexity

- Certain expressions are convex:
  - ⋆ $-f(\mathbf{x})$ for any concave f
  - ⋆ $\alpha f(\mathbf{x})$ for any convex f and $\alpha > 0$
  - ⋆ $f(\mathbf{x}) + g(\mathbf{x})$ for any convex f and g
  - ⋆ $f(\mathbf{Ax} + \mathbf{b})$ for any convex f
  - ⋆ $f(g(\mathbf{x}))$ for any convex nondecreasing f and convex g
  - ⋆ $f(g(\mathbf{x}))$ for any convex nonincreasing f and concave g
- Use these with function properties to assess convexity of node expressions on their domains

Apply properties of concavity, similarly

Proof of Convexity *(cont'd)*

Recursively apply `isConvex (lb,ub)`

- Return values
  - `+1: convex`
  - ` 0: can't tell`
  - `-1: concave`
- Bounds
  - `lb: lower bound`
  - `ub: upper bound`

Deduce status of each nonlinear expression

- Convex, concave, or indeterminate
- Lower and upper bounds

Disproof of Convexity

Find any counterexample

- Sample in feasible region
- Test any characterization of convex functions

Sampling along lines

- Look for $f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) > \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2)$
- See implementation in John Chinneck's MProbe (www.sce.carleton.ca/faculty/chinneck/mprobe.html)

Sampling at points

- Look for $\nabla^2 f(\mathbf{x})$ not positive semi-definite
- Implemented in DrAMPL . . .

Disproof of Convexity *(cont'd)*

Sampling

- Choose points $\mathbf{x}_0$
  such that $x_{01}, \ldots, x_{0n}$ are within inferred bounds

Testing

- Apply GLTR (galahad.rl.ac.uk/galahad-www/doc/gltr.pdf) to

$$\begin{array}{ll} \min_{\mathbf{d}} & \nabla f(\mathbf{x}_0)\mathbf{d} + \tfrac{1}{2}\mathbf{d}\nabla^2 f(\mathbf{x}_0)\mathbf{d} \\ \text{s.t.} & \|\mathbf{d}\|_2 \le \max\{10, \|\nabla f(x_0)\|/10\} \end{array}$$

- Declare ***nonconvex*** if GLTR's Lanczos method
  finds a direction of negative curvature
- Declare ***inconclusive*** if GLTR
  reaches the trust region boundary
  without finding a direction of negative curvature

Testing Convexity Analyzers

Principles

- Disprovers can establish nonconvexity, suggest convexity
- Provers can establish convexity, suggest nonconvexity

Test problems

- Established test sets:
  COPS (17), CUTE (734), Hock & Schittkowski (119),
  Netlib (40), Schittkowski (195), Vanderbei (29 groups)
- Submissions to NEOS Server

Design of experiments

- Run a prover and a disprover on each test problem
- Check results for consistency
- Collect and characterize problems found to be convex
- Inspect functions not proved or disproved convex,
  to suggest possible enhancements to analyzers

Example

Torsion model (parameters and variables)

```
param nx > 0, integer;    # grid points in 1st direction
param ny > 0, integer;    # grid points in 2nd direction

param c;                  # constant

param hx := 1/(nx+1);     # grid spacing
param hy := 1/(ny+1);     # grid spacing

param area := 0.5*hx*hy;  # area of triangle

param D {i in 0..nx+1,j in 0..ny+1} =

  min( min(i,nx-i+1)*hx, min(j,ny-j+1)*hy );

                          # distance to the boundary

var v {i in 0..nx+1, j in 0..ny+1};

                          # definition of the
                          # finite element approximation
```

Example *(cont'd)*

Torsion model (objective and constraints)

```
var linLower = sum {i in 0..nx, j in 0..ny}
   (v[i+1,j] + v[i,j] + v[i,j+1]);

var linUpper = sum {i in 1..nx+1, j in 1..ny+1}
   (v[i,j] + v[i-1,j] + v[i,j-1]);

var quadLower = sum {i in 0..nx,j in 0..ny} (
   ((v[i+1,j] - v[i,j])/hx)**2 + ((v[i,j+1] - v[i,j])/hy)**2 );

var quadUpper = sum {i in 1..nx+1, j in 1..ny+1} (
   ((v[i,j] - v[i-1,j])/hx)**2 + ((v[i,j] - v[i,j-1])/hy)**2 );

minimize Stress:
  area * ((quadLower+quadUpper)/2 - c*(linLower+linUpper)/3);

subject to distanceBound {i in 0..nx+1, j in 0..ny+1}:
  -D[i,j] <= v[i,j] <= D[i,j];
```

Example *(cont'd)*

Output from AMPL's presolver

```
Presolve eliminates 2704 constraints and 204 variables.
Substitution eliminates 4 variables.

Adjusted problem:
2500 variables, all nonlinear
0 constraints
1 nonlinear objective; 2500 nonzeros.
```

Example *(cont'd)*

Output from DrAMPL (analysis)

```
Problem type
-------------
 -Problem has bounded variables
 -Problem has no constraints

Analyzing problem using only objective
--------------------------------------
 -This objective is quadratic
 -Problem is a QP with bounds

 -0.833013 <= objective <= 0.8359

Problem convexity
------------------
 Nonlinear objective looks convex on its domain.

 Detected 0/0 nonlinear convex  constraints,
          0/0 nonlinear concave constraints.
```

Issues

Algorithmic requirements

- Convexity outside feasible region

Nonconvex cases missed

- Choice of starting point can be crucial

Convex cases missed

- Polynomials
  - $x^4 - 4x^3 + 6x^2 - 4x + 1$ is $(x-1)^4$
  - $x^4 - 4x^3 + 7x^2 - 2x + 2$ is $(x-1)^4 + (x+1)^2$
- ***Quadratics . . .***

Convexity of Quadratic Expressions

"Elliptic" quadratic programming

- Detection
- Solving

"Conic" quadratic programming

- Detection
- Solving
- **Conversion**

. . . Ph.D. project of Jared Erickson,
Northwestern University

"Elliptic" Quadratic Programming

Symbolic detection

- Objectives
  - Minimize $x_1^2 + \ldots + x_n^2$
  - Minimize $\sum_{i=1}^{n} a_i (\mathbf{f}_i \mathbf{x} + g_i)^2,\; a_i \geq 0$
- Constraints
  - $x_1^2 + \ldots + x_n^2 \leq r$
  - $\sum_{i=1}^{n} a_i (\mathbf{f}_i \mathbf{x} + g_i)^2 \leq r,\; a_i \geq 0$

Numerical detection

- Objectives
  - Minimize $\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q} \mathbf{x}$
- Constraints
  - $\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q} \mathbf{x} \leq r$
- . . . where **Q** is positive semidefinite

Solving

Representation

- Much like LP
  - Coefficient lists for linear terms
  - Coefficient lists for quadratic terms
- No expression trees

Optimization

- Much like LP
  - Generalizations of barrier methods
  - Generalizations of simplex methods
  - Extensions of mixed-integer branch-and-bound schemes
- Simple derivative computations
- Less overhead than general-purpose nonlinear solvers

. . . ***your speedup may vary***

Elliptic QP

Example

Portfolio optimization

```
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;
```

Elliptic QP

Example *(cont'd)*

Portfolio data

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

param R:
  US_3-MONTH_T-BILLS US_GOVN_LONG_BONDS SP_500 WILSHIRE_5000
  NASDAQ_COMPOSITE LEHMAN_BROTHERS_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 ...
```

Example *(cont'd)*

Solving with CPLEX

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

ampl: option solver cplexamp;

ampl: solve;

8 variables, all nonlinear
1 constraint, all linear; 8 nonzeros
1 nonlinear objective; 8 nonzeros.

CPLEX 12.2.0.0: optimal solution; objective -1.098362471
12 QP barrier iterations

ampl:
```

Example *(cont'd)*

Solving with CPLEX (simplex)

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

ampl: option solver cplexamp;
ampl: option cplex_options 'primalopt';

ampl: solve;

8 variables, all nonlinear
1 constraint, all linear; 8 nonzeros
1 nonlinear objective; 8 nonzeros.

CPLEX 12.2.0.0: primalopt
No QP presolve or aggregator reductions.

CPLEX 12.2.0.0: optimal solution; objective -1.098362476
5 QP simplex iterations (0 in phase I)

ampl:
```

Example *(cont'd)*

Optimal portfolio

```
ampl: option omit_zero_rows 1;

ampl: display Frac;

                                  EAFE  0.216083
                                  GOLD  0.185066
LEHMAN_BROTHERS_CORPORATE_BONDS_INDEX  0.397056
                          WILSHIRE_5000  0.201795 ;

ampl: display Mean, Variance;

Mean = 1.11577
Variance = 0.00870377

ampl:
```

Example *(cont'd)*

Optimal portfolio (discrete)

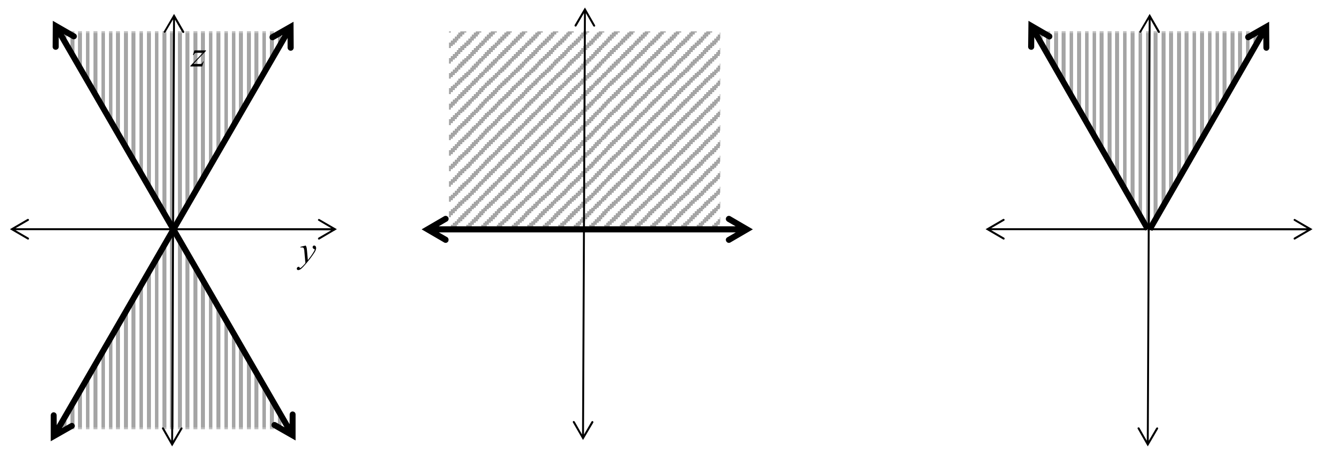
```
var Share {A} integer >= 0, <= 100;

var Frac {j in A} = Share[j] / 100;
```

```
ampl: solve;

CPLEX 12.2.0.0: optimal integer solution within mipgap or absmipgap;
   objective -1.098353751

10 MIP simplex iterations
0 branch-and-bound nodes

absmipgap = 8.72492e-06, relmipgap = 7.94364e-06

ampl: display Frac;

                              EAFE  0.22
                              GOLD  0.18
LEHMAN_BROTHERS_CORPORATE_BONDS_INDEX  0.4
                      WILSHIRE_5000  0.2 ;
```

Second-Order Cone Constraints

Standard cone

$x^2 + y^2 \leq z^2$ $\qquad$ $z \geq 0$ $\qquad$ $x^2 + y^2 \leq z^2,\ z \geq 0$



. . . boundary not smooth

Rotated cone

- $x^2 \leq yz,\ y \geq 0, z \geq 0,\ . . .$

"Conic" Quadratic Programming

Symbolic detection

- Constraints (standard)
  - $x_1^2 + \ldots + x_n^2 \le x_{n+1}^2,\ \ x_{n+1} \ge 0$
  - $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})^2,$
    $a_1, \ldots, a_{n+1} \ge 0,\ \ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \ge 0$
- Constraints (rotated)
  - $x_1^2 + \ldots + x_n^2 \le x_{n+1}\, x_{n+2},\ x_{n+1} \ge 0,\ x_{n+2} \ge 0$
  - $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})(\mathbf{f}_{n+2}\mathbf{x} + g_{n+2}),$
    $a_1, \ldots, a_{n+1} \ge 0,\ \ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \ge 0,\ \ \mathbf{f}_{n+2}\mathbf{x} + g_{n+2} \ge 0$

Numerical detection

- $\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}\mathbf{x} \le r$
- . . . where **Q** has one negative eigenvalue
  - see Ashutosh Mahajan and Todd Munson, "Exploiting Second-Order Cone Structure for Global Optimization"

Solving

Similarities

- ❖ Describe by lists of coefficients
- ❖ Solve by extensions of LP barrier methods
- ❖ Extend to mixed-integer branch-and-bound

Differences

- ❖ Quadratic part not positive semi-definite
- ❖ Nonnegativity is essential
- ❖ Boundary of feasible region is not differentiable
- ❖ ***Many convex problems can be reduced to these . . .***

Terminology

- ❖ Second-order cone programs, SOCPs
- ❖ Allow also elliptical quadratic & linear constraints

Conic QP

Example 1: Sum of Norms

```
param p integer > 0;
param m {1..p} integer > 0;
param n integer > 0;

param F {i in 1..p, 1..m[i], 1..n};
param g {i in 1..p, 1..m[i]};
```

```
param p := 2 ;
param m := 1 5  2 4 ;
param n := 3 ;

param g (tr): 1   2 :=
          1  12   2
          2   7  11
          3   7   1
          4   8   0
          5   4   . ;

param F := ...
```

Conic QP

Example 1: Original Formulation

```
var x {1..n};

minimize SumOfNorms:
   sum {i in 1..p} sqrt(
      sum {k in 1..m[i]} (sum {j in 1..n} F[i,k,j] * x[j] + g[i,k])^2 );
```

```
3 variables, all nonlinear
0 constraints
1 nonlinear objective; 3 nonzeros.

CPLEX 12.2.0.0: at12228.nl contains a nonlinear objective.
```

Conic QP

Example 1: Converted to Quadratic

```
var x {1..n};
var Max {1..p} >= 0;

minimize SumOfNorms: sum {i in 1..p} Max[i];

subj to MaxDefinition {i in 1..p}:
   sum {k in 1..m[i]} (sum {j in 1..n} F[i,k,j] * x[j] + g[i,k])^2
      <= Max[i]^2;
```

```
5 variables, all nonlinear
2 constraints, all nonlinear; 8 nonzeros
1 linear objective; 2 nonzeros.

CPLEX 12.2.0.0: QP Hessian is not positive semi-definite.
```

Example 1: Simpler Quadratic

```
var x {1..n};
var Max {1..p} >= 0;
var Fxplusg {i in 1..p, 1..m[i]};

minimize SumOfNorms: sum {i in 1..p} Max[i];

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

subj to FxplusgDefinition {i in 1..p, k in 1..m[i]}:
   Fxplusg[i,k] = sum {j in 1..n} F[i,k,j] * x[j] + g[i,k];
```

```
14 variables:
      11 nonlinear variables
      3 linear variables
11 constraints; 41 nonzeros
      2 nonlinear constraints
      9 linear constraints
1 linear objective; 2 nonzeros.

CPLEX 12.2.0.0: primal optimal; objective 11.03323293
11 barrier iterations
```

Example 1: Integer Quadratic

```
var xint {1..n} integer;

var x {j in 1..n} = xint[j] / 10;

.......
```

```
Substitution eliminates 3 variables.

14 variables:
      11 nonlinear variables
      3 integer variables
11 constraints; 41 nonzeros
      2 nonlinear constraints
      9 linear constraints
1 linear objective; 2 nonzeros.

CPLEX 12.2.0.0: optimal integer solution; objective 11.12932573
88 MIP simplex iterations
19 branch-and-bound nodes
```

Example 2: Traffic Network

```
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;
```

Conic QP

Example 2: Original Formulation

```
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;
```

Conic QP

Example 2: Rotated Cone Formulation

```
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}:
  1 - Flow[i,j]/cap[i,j] = Slack[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;
```

Example 2: Solution

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

ampl: solve;

CPLEX 12.3.0.0: Constraint _scon[1] is not convex quadratic
since it is an equality constraint.
```

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

ampl: solve;

CPLEX 12.3.0.0: primal optimal; objective 8.178571451
8 barrier iterations
```

SOCP-Solvable Forms

Quadratic

- Constraints (already seen)
- Objectives

SOC-representable

- Quadratic-linear ratios
- Generalized geometric means
- Generalized p-norms

Other objective functions

- Generalized product-of-powers
- Logarithmic Chebychev

SOCP-solvable

Quadratic

Standard cone constraints

- $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})^2,$
  $a_1, \ldots, a_{n+1} \ge 0,\ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \ge 0$

Rotated cone constraints

- $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})(\mathbf{f}_{n+2}\mathbf{x} + g_{n+2}),$
  $a_1, \ldots, a_{n+1} \ge 0,\ \mathbf{f}_{n+1}\mathbf{x} + g_{n+1} \ge 0,\ \mathbf{f}_{n+2}\mathbf{x} + g_{n+2} \ge 0$

Sum-of-squares objectives

- Minimize $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2$
  - Minimize v
    Subject to $\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x} + g_i)^2 \le v^2,\ v \ge 0$

SOCP-solvable

SOC-Representable

Definition

- Function $s(x)$ is SOC-representable *iff* . . .
- $s(x) \leq a_n(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})$ is equivalent to some combination of linear and quadratic cone constraints

Minimization property

- Minimize $s(x)$ is SOC-solvable
  - Minimize v_{n+1}
    Subject to $s(x) \leq v_{n+1}$

Combination properties

- $a \cdot s(x)$ is SOC-representable for any $a \geq 0$
- $\sum_{i=1}^{n} s_i(x)$ is SOC-representable
- $max_{i=1}^{n}\, s_i(x)$ is SOC-representable

. . . ***requires a recursive detection algorithm!***

SOCP-solvable

SOC-Representable (1)

Vector norm

- $\|\mathbf{a}\cdot(\mathbf{Fx}+\mathbf{g})\| = \sqrt{\sum_{i=1}^{n} a_i^2 (\mathbf{f}_i\mathbf{x}+g_i)^2} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$
  - ∗ Square both sides to get standard SOC
    $\sum_{i=1}^{n} a_i^2 (\mathbf{f}_i\mathbf{x}+g_i)^2 \le a_{n+1}^2(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})^2$

Quadratic-linear ratio

- $\dfrac{\sum_{i=1}^{n} a_i (\mathbf{f}_i\mathbf{x}+g_i)^2}{\mathbf{f}_{n+2}\mathbf{x}+g_{n+2}} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$
  - ∗ Multiply by denominator to get rotated SOC
    $\sum_{i=1}^{n} a_i\,(\mathbf{f}_i\mathbf{x}+g_i)^2 \le a_{n+1}\,(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})(\mathbf{f}_{n+2}\mathbf{x}+g_{n+2})$

SOCP-solvable

SOC-Representable (2)

Negative geometric mean

- $-\prod_{i=1}^{p}(\mathbf{f}_i\mathbf{x}+g_i)^{1/p} \le \mathbf{f}_{n+1}\mathbf{x}+g_{n+1},\ \ p\in\mathbb{Z}^+$
  - $-x_1^{1/4}x_2^{1/4}x_3^{1/4}x_4^{1/4} \le -x_5$ becomes rotated SOCs: $x_5^2 \le v_1v_2,\ v_1^2 \le x_1x_2,\ v_2^2 \le x_3x_4$
  - apply recursively $\lceil \log_2 p \rceil$ times

Generalizations

- $-\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{\alpha_i} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$: $\sum_{i=1}^{n}\alpha_i \le 1$, $\alpha_i\in\mathbb{Q}^+$
- $\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{-\alpha_i} \le\ a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1}),\ \ \alpha_i\in\mathbb{Q}^+$

SOCP-solvable

SOC-Representable (3)

p-norm

- $(\sum_{i=1}^{n} |\mathbf{f}_i \mathbf{x} + g_i|^p)^{1/p} \le \mathbf{f}_{n+1}\mathbf{x} + g_{n+1},\ p \in \mathbb{Q}^+,\ p \ge 1$
  - $(|x_1|^5 + |x_2|^5)^{1/5} \le x_3$ can be written
    $|x_1|^5/x_3^4 + |x_2|^5/x_3^4 \le x_3$ which becomes
    $v_1 + v_2 \le x_3$ with $-v_1^{1/5}\, x_3^{4/5} \le \pm x_1, -v_1^{1/5}\, x_3^{4/5} \le \pm x_2$
  - reduces to product of powers

Generalizations

- $(\sum_{i=1}^{n} |\mathbf{f}_i \mathbf{x} + g_i|^{\alpha_i})^{1/\alpha_0} \le \mathbf{f}_{n+1}\mathbf{x} + g_{n+1},\ \alpha_i \in \mathbb{Q}^+,\ \alpha_i \ge \alpha_0 \ge 1$
- $\sum_{i=1}^{n} |\mathbf{f}_i \mathbf{x} + g_i|^{\alpha_i} \le (\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})^{\alpha_0}$
- Minimize $\sum_{i=1}^{n} |\mathbf{f}_i \mathbf{x} + g_i|^{\alpha_i}$

. . . ***standard SOCP has*** $\boldsymbol{\alpha_i \equiv 2}$

SOCP-solvable

Other Objective Forms

Unrestricted product of powers

- Minimize $-\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{\alpha_i}$ for any $\alpha_i \in \mathbb{Q}^+$

Logarithmic Chebychev approximation

- Minimize $\max_{i=1}^{n}|\log(\mathbf{f}_i\mathbf{x})-\log(g_i)|$

Why no constraint versions?

- Not SOC-representable
- Transformation changes objective value (but not solution)

Example: Sum-of-Norms Objective

Given

- Minimize $\sum_{i=1}^{m} a_i \sqrt{\sum_{j=1}^{n} \left(\mathbf{f}_{ij}\mathbf{x} + g_{ij}\right)^2 + c_i}$

Transform to

- Minimize $\sum_{i=1}^{m} s_i$
- $\sum_{j=1}^{n} t_{ij}^2 + a_i^2 c_i \leq s_i^2,\ \ s_i \geq 0, i = 1, \ldots, m$
- $a_i\left(\mathbf{f}_{ij}\mathbf{x} + g_{ij}\right) = t_{ij}\ , \qquad\qquad j = 1, \ldots, n$

Sum of Norms

Detection

```
boolean isSumNorms (Node);

case of Node {
 PLUS:  return( isSumNorms(Node.left) and isSumNorms(Node.right) );
 TIMES: return( isSumNorms(Node.right) and
                isConst(Node.left) and value(Node.left) > 0 );
 SQRT:  return( isNormSquared(Node.child) );
}

boolean isSumSquares (Node);

case of Node {
 PLUS:  return( isSumSquares(Node.left) and isSumSquares(Node.right) );
 POWER: return( isLinear(Node.left) and
                isConst(Node.right) and value(Node.right) == 2 );
 CONST: return( value(Node) > 0 );
}
```

Transformation: Preliminaries

Functions

- o objective
- l_i linear inequality — q_i standard cone
- e_i linear equality — r_i rotated cone

Terminology

- m_c most recently used constraint of type c
- n most recently used variable index
- x vector of original variables
- v vector of all variables

Example

- $l_1(v) \coloneqq 3x_1 - 2,\ l_1(v) \coloneqq l_1(v) + v_3,\ \ l_1(v) \le 0$
  create $3x_1 + v_3 \le 2$

Sum of Norms

Transformation: Utilities

```
newVar ( b );

n++;
add variable v_n ;
if ( b ) then add v_n ≥ b ;

newFunc ( c );

m_c++;
add constraint of type c ;
c_{m_c}(v) := 0 ;
```

Transformation: Sum of Norms

```
newFunc( o );
o_{m_o}(v) := 0 ;
tranSumNorms( Root, o_{m_o}(v), 1 );
```

Transformation: Sum of Norms

```
tranSumNorms ( Node, o(v), c );

case of Node {

 PLUS:   tranSumNorms( Node.left, o(v), c );
         tranSumNorms( Node.right, o(v), c );

 MULT:   tranSumNorms( Node.right, o(v), c·value(Node.left) );

 SQRT:   newVar( 0 );
         o(v) := o(v) + v_n ;

         newFunc( q );
         q_{m_q}(v) := -v_n^2 ;

         tranSumSquares( Node.child, q_{m_q}(v), c );

}
```

Sum of Norms

Transformation: Sum of Squares

```
tranSumSquares ( Node, q(v), c );

case of Node {

 PLUS:  tranSumSquares( Node.left, q(v), c );
        tranSumSquares( Node.right, q(v), c );

 POWER: newvar(  );
        q(v) := q(v) + v_n^2;

        newfunc( e );
        e_{m_e}(v) := −v_n ;

        tranLinear( Node.left, e_{m_e}(v), c );

 CONST: q(v) := q(v) + c^2 · value(Node);
}
```

Issues

Which SOCP-solvable forms . . .

- ❖ are of practical use?
- ❖ are worth transforming?
  - ∗ for continuous problems?
  - ∗ for integer problems?