Strategies for Using Algebraic Modeling Languages to Formulate Second-Order Cone Programs

Robert Fourer, Jared Erickson

Industrial Engineering & Management Sciences
Northwestern University

4er@northwestern.edu, jarederickson2012@u.northwestern.edu

INFORMS Annual Meeting
Charlotte — 13-16 November 2011

SB03.1 Optimization Software

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 $(i,j) \in A$

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

Traffic Network

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

Traffic Network

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

Traffic Network

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 CPLEX?

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

ampl: option solver cplex;
ampl: solve;

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

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 CPLEX?

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

ampl: option solver cplex;
ampl: solve;

CPLEX 12.3.0.0:
QP Hessian is not positive semi-definite.
```

AMPL Solution *(cont'd)*

Quadratic reformulation #2

```
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 CPLEX!

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

ampl: option solver cplex;
ampl: solve;

CPLEX 12.3.0.0: primal optimal; objective 61.04693968
15 barrier iterations

ampl: display Flow;

Flow :=
a b    9.55175
a c   10.4482
b d   11.0044
c b    1.45264
c d    8.99561
;
```

AMPL Solution *(cont'd)*

Same with integer-valued variables

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

```
ampl: solve;

CPLEX 12.3.0.0: optimal integer solution within mipgap or absmipgap;
   objective 76.26375017
19 MIP barrier iterations
0 branch-and-bound nodes

ampl: display Flow;

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

Which Solver Is Preferable?

General nonlinear solver

- Fewer variables
- More natural formulation

MIP solver with convex quadratic option

- Mathematically simpler formulation
- No derivative evaluations
    - no problems with nondifferentiable points
- More powerful large-scale solver technologies

Outline

Convex quadratic programs

- “Elliptic” forms
- “Conic” forms
  - ***SOCPs*** or *second-order cone programs*

SOCP-solvable forms

- Quadratic
- SOC-representable
- Other objective functions

Detection & transformation of SOCPs

- General principles
- Example: Sum of norms
- Survey of nonlinear test problems

Convex Quadratic Programs

"Elliptic" quadratic programming

- Detection
- Solving

"Conic" quadratic programming

- Detection
- Solving

"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

Elliptic QP

Solving

Representation

- ❖ Much like LP
  - ∗ Coefficient lists for linear terms
  - ∗ Coefficient lists for quadratic terms
- ❖ A lot simpler than general NLP

Optimization

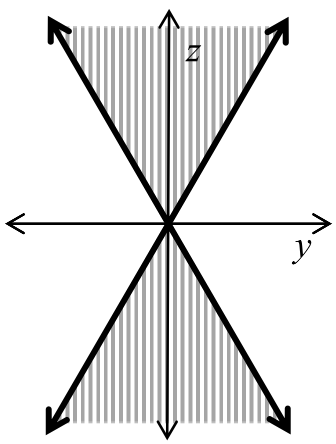
- ❖ Much like LP
  - ∗ Generalizations of barrier methods
  - ∗ Generalizations of simplex methods
  - ∗ Extensions of mixed-integer branch-and-bound schemes
- ❖ Simple derivative computations
- ❖ Less overhead than general-purpose nonlinear solvers

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

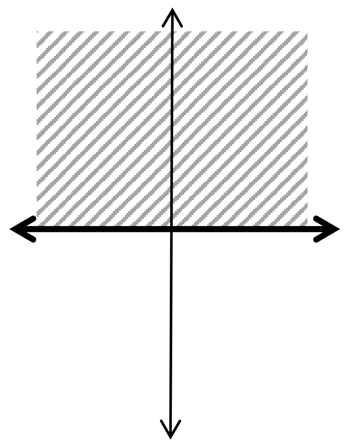
“Conic” Quadratic Programming

Standard cone

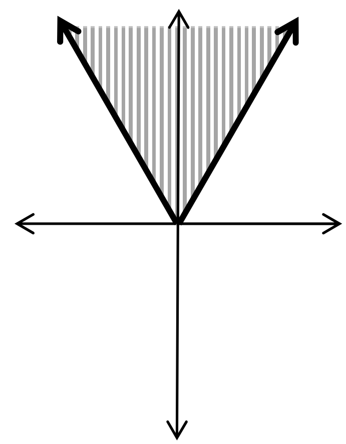
$x^2 + y^2 \leq z^2$



$z \geq 0$



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



. . . 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"

Conic QP

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

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

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

- $\|\mathbf{a}\cdot(\mathbf{F}\mathbf{x}+\mathbf{g})\| = \sqrt{\sum_{i=1}^{n} a_i^2(\mathbf{f}_i\mathbf{x}+g_i)^2} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$
  - 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

- $\dfrac{\sum_{i=1}^{n} a_i(\mathbf{f}_i\mathbf{x}+g_i)^2}{\mathbf{f}_{n+2}\mathbf{x}+g_{n+2}} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$
  - where $\mathbf{f}_{n+2}\mathbf{x}+g_{n+2} \ge 0$
  - multiply by denominator to get rotated SOC
    $\sum_{i=1}^{n} a_i\,(\mathbf{f}_i\mathbf{x}+g_i)^2 \le a_{n+1}\,(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})(\mathbf{f}_{n+2}\mathbf{x}+g_{n+2})$

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

- $-\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{\alpha_i} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1})$: $\sum_{i=1}^{n}\alpha_i \le 1,\; \alpha_i \in \mathbb{Q}^+$
- $\prod_{i=1}^{n}(\mathbf{f}_i\mathbf{x}+g_i)^{-\alpha_i} \le a_{n+1}(\mathbf{f}_{n+1}\mathbf{x}+g_{n+1}),\;\; \alpha_i \in \mathbb{Q}^+$
  - all require $\mathbf{f}_i\mathbf{x}+g_i \ge 0$

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 $\alpha_i \equiv 2$

Other Objective Functions

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)

Detection & Transformation of SOCPs

Principles

- Representation of expressions by trees
- Recursive tree-walk functions
  - `isLinear()`, `isQuadratic()`, `buildLinear()`

Example: Sum of norms

Survey of nonlinear test problems

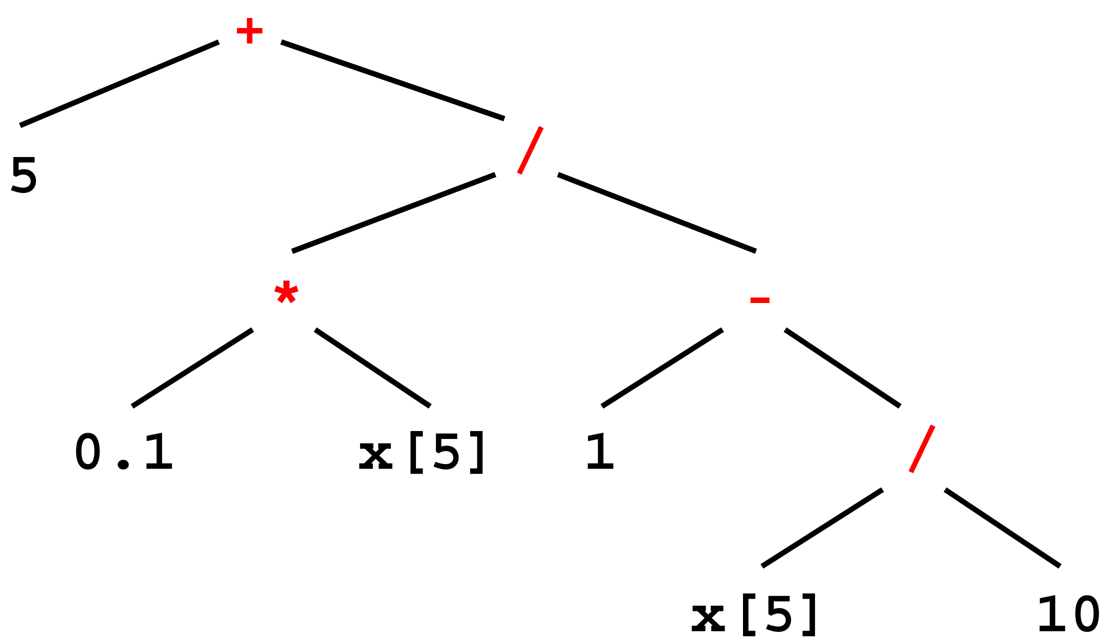
. . . Ph.D. project of Jared Erickson,
Northwestern University

Representation

Expression

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

Expression tree



. . . actually a DAG

Principles

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 );

}
```

Principles

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)`

Example: Sum-of-Norms Objective

Given

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

Transform to

- Minimize $\sum_{i=1}^{m} a_i y_i$
- $\sum_{j=1}^{n_i} z_{ij}^2 \le y_i^2,\ \ y_i \ge 0, \qquad i = 1,\ldots,m$
- $z_{ij} = \mathbf{f}_{ij}\mathbf{x} + g_{ij}, \qquad i = 1,\ldots,m,\ \ j = 1,\ldots,n_i$

Sum of Norms

Detection

SumOfNorms

Sum: $f_1 + f_2$ is SumOfNorms if f_1, f_2 are SumOfNorms

Product: $f_1 f_2$ is SumOfNorms if
  f_1 is SumOfNorms and f_2 is PosConstant or
  f_2 is SumOfNorms and f_1 is PosConstant

Square root: $\sqrt{f}$ is SumOfNorms if f is SumOfSquares

SumOfSquares

Sum: $f_1 + f_2$ is SumOfSquares if f_1, f_2 are SumOfSquares

Product: $f_1 f_2$ is SumOfSquares if
  f_1 is SumOfSquares and f_2 is PosConstant or
  f_2 is SumOfSquares and f_1 is PosConstant

Square: f^2 is SumOfSquares if f is Linear

Constant: c is SumOfSquares if c is PosConstant

Detection *(cont'd)*

Mathematical

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

Practical

- Constant multiples inside any sum
- Recursive nesting of constant multiples & sums
- Constant as a special case of a square
  - $\sqrt{3(4x_1 + 7(x_2 + 2x_3) + 6)^2 + (x_4 + x_5)^2 + 17}$

Sum of Norms

Transformation

TransformSumOfNorms (f, o, k)

Sum: $f_1 + f_2$ where f_1, f_2 are SumOfNorms

TransformSumOfNorms (f_1, o, k)

TransformSumOfNorms (f_2, o, k)

Product: $f_1 c_2$ where f_1 is SumOfNorms and c_2 is PosConstant

TransformSumOfNorms $(f_1, o, c_2 k)$

Product: $c_1 f_2$ where f_2 is SumOfNorms and c_1 is PosConstant

TransformSumOfNorms $(f_2, o, c_1 k)$

Square root: $\sqrt{f}$ where f is SumOfSquares

y_i := NewNonnegVar(); o += $k y_i$

q_i := NewLeCon(); q_i += $-y_i^2$

TransformSumOfSquares$(f, q_i, 1)$

Sum of Norms

Transformation *(cont'd)*

TRANSFORMSUMOFSQUARES (f, q, k)

Sum: $f_1 + f_2$ where f_1, f_2 are SUMOFSQUARES

TRANSFORMSUMOFSQUARES (f_1, q, k)

TRANSFORMSUMOFSQUARES (f_2, q, k)

Product: $f_1 c_2$ where f_1 is SUMOFSQUARES and c_2 is POSCONSTANT

TRANSFORMSUMOFSQUARES ($f_1, q, c_2 k$)`

Product: $c_1 f_2$ where f_2 is SUMOFSQUARES and c_1 is POSCONSTANT

TRANSFORMSUMOFSQUARES ($f_2, q, c_1 k$)

Square: v^2 where v is VARIABLE

$q \mathrel{+}= v^2$

Square: f^2 where f is LINEAR

z_{ij} := NEWVAR(); $q \mathrel{+}= z_{ij}^2$

e_{ij} := NEWEQCON(); $e_{ij} \mathrel{+}= z_{ij} - f$

Constant: c is POSCONSTANT

z_{ij} := NEWVAR(); $q \mathrel{+}= z_{ij}^2$

e_{ij} := NEWEQCON(); $e_{ij} \mathrel{+}= z_{ij} - \sqrt{c}$

Transformation *(cont'd)*

Mathematical

- Minimize $\sum_{i=1}^{m} a_i y_i$
- $\sum_{j=1}^{n_i} z_{ij}^2 \leq y_i^2,\ y_i \geq 0$
- $z_{ij} = \mathbf{f}_{ij}\mathbf{x} + g_{ij}$

Practical

- Handle all previously mentioned generalizations
- Don't define z_{ij} when $\mathbf{f}_{ij}\mathbf{x} + g_{ij}$ is a single variable
- Trigger by calling TRANSFORMSUMOFNORMS(f, o, k) with
  - f the root node
  - o an empty objective
  - $k = 1$

Challenges

Extending to all cases previously cited

- All prove amenable to recursive tree-walk
- Details much harder to work out

Checking nonnegativity of linear expressions

- Heuristic catches many non-obvious instances

Assessing usefulness

- With continuous variables . . .
- With discrete variables . . .

Survey of Test Problems

12% of 1238 nonlinear problems were SOC-solvable!

- not counting QPs with sum-of-squares objectives
- from Vanderbei's CUTE & non-CUTE, and netlib/ampl

A variety of forms detected

- `hs064` has $4/x_1 + 32/x_2 + 120/x_3 \leq 1$
- `hs036` minimizes $-x_1 x_2 x_3$
- `hs073` has $1.645\sqrt{0.28x_1^2 + 0.19x_2^2 + 20.5x_3^2 + 0.62x_4^2} \leq \ldots$
- `polak4` is a max of sums of squares
- `hs049` minimizes $(x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6$
- `emfl_nonconvex` has $\sum_{k=1}^{2}\left(x_{jk} - a_{ik}\right)^2 \leq s_{ij}^2$

. . . survey of integer programs to come

. . . solver tests to come