Convex Quadratic Programming in AMPL

Robert Fourer
Northwestern University / AMPL Optimization
4er@ampl.com

Jared Erickson
Northwestern University / Ziena Optimization
jarederickson2012@u.northwestern.edu

4th International Conference on Continuous Optimization
Caparica, Lisbon, Portugal — 29 July–1 August 2013
Thu.A.23: Extending the Power and Expressiveness of Optimization Modeling Languages
Convex Quadratic Programming in AMPL

A surprising variety of optimization applications can be written in terms of convex quadratic objectives and constraints that are handled effectively by extensions to linear solvers. “Elliptical” convex quadratic programs are easily recognized once the matrices of quadratic coefficients are extracted, through a test for positive-semidefiniteness. “Conic” problems are also convex quadratic and can in principle also be detected numerically, but are more commonly recognized by their equivalence to certain canonical forms. Additionally, varied combinations of sums-of-squares, Euclidean norms, quadratic-linear ratios, products of powers, p-norms, and log-Chebychev terms can be identified symbolically and transformed to quadratic problems that have conic formulations. The power and convenience of an algebraic modeling language may be extended to support these cases, with the help of a recursive tree-walk approach that detects and (where necessary) transforms arbitrarily complex instances; modelers are thereby freed from the time-consuming and error-prone work of maintaining the equivalent canonical formulations explicitly. We describe the challenges of creating the requisite detection and transformation routines for the AMPL language, and report computational tests that suggest the usefulness of these routines.
Outline

What is convex quadratic?
- What kind of solver do you want to use?

Introductory examples
- Product of linear terms
- Traffic network

Detection and transformation
- Where they are done now
- Where they should be done
- Our new extensions
  * Theory
  * Implementation
  * Testing
What is Convex Quadratic?

Convex quadratic objective
  - PSD quadratic + linear

Convex quadratic constraints
  - Linear
  - PSD quadratic \( \leq \) constant
  - Conic quadratic

Anything transformable to the above

What kind of solver do you want to use?
What kind of solver?

General Nonlinear Solver

MINOS, KNITRO, Ipopt, SNOPT, CONOPT, . . .

Advantages

- Accepts any form of problem
- Tolerates nonconvexities

Disadvantages

- Relies on smoothness
- Uses complex mechanisms
  * Function evals, line searches, convergence tests, . . .
- Reports only local optimality
What kind of solver?

Extended Linear Solver

**CPLEX, Gurobi, Xpress, MOSEK, ...**

**Advantages**
- Uses mechanisms adapted from linear programming
  - Sparse coefficient lists, fast interior-point methods, ...
- Tolerates nonsmooth functions & regions
- Reports global optimality

**Disadvantages**
- Requires recognizable convex quadratic formulations
- Rejects problems not in required form
Convex Quadratic Programming in AMPL
Using “Linear” Solvers

Robert Fourer
Northwestern University / AMPL Optimization
4er@ampl.com

Jared Erickson
Northwestern University / Ziena Optimization
jarederickson2012@u.northwestern.edu

4th International Conference on Continuous Optimization
Caparica, Lisbon, Portugal — 29 July–1 August 2013
Thu.A.23: Extending the Power and Expressiveness of Optimization Modeling Languages
Possibilities for Integer Variables

**Zero-one**
- Extend linear branch-and-bound
- Transform to linear
  - requires just one binary in each quadratic term
  - many alternatives available
- Transform to PSD quadratic
  - based on $x^2 = x$ for any binary $x$

**General integer**
- Extend linear branch-and-bound
- Transform to zero-one
  - creates $\log_2 U$ binaries for domain of size $U$
Convex Quadratic Programming in AMPL
Using “Linear” Solvers

Robert Fourer
Northwestern University / AMPL Optimization
4er@ampl.com

Jared Erickson
Northwestern University / Ziena Optimization
jarederickson2012@u.northwestern.edu

4th International Conference on Continuous Optimization
Caparica, Lisbon, Portugal — 29 July–1 August 2013
Thu.A.23: Extending the Power and Expressiveness of Optimization Modeling Languages
Example 1: Product of Linear Terms

Original formulation

- Maximize $(\sum_{j=1}^{n} c_j x_j) (\sum_{j=1}^{n} d_j y_j)$
- $\sum_{j=1}^{n} c_j x_j \geq 0$, $\sum_{j=1}^{n} d_j y_j \geq 0$

Conic reformulation

- Maximize $z$
- $z^2 \leq z_x z_y$, $z_x \geq 0$, $z_y \geq 0$
- $z_x = \sum_{j=1}^{n} c_j x_j$, $z_y = \sum_{j=1}^{n} d_j y_j$
**AMPL Model**

**Direct formulation**

```plaintext
param n > 0;
param c {1..n} > 0;
param d {1..n} > 0;

var X {1..n} >= 0, <= 2;
var Y {1..n} >= 0, <= 2;

maximize Obj:
    (sum {j in 1..n} c[j]*X[j]) * (sum {j in 1..n} d[j]*Y[j]);

subject to SumX: sum {j in 1..n} j * X[j] >= 17;
subject to SumY: sum {j in 1..n} j * Y[j] >= 17;
subject to SumXY: sum {j in 1..n} (X[j] + Y[j]) = 7;
```
Product of Linear Terms

AMPL Solution

Solved by KNITRO

```ampl
AMPL: model xy4a.mod;
AMPL: option solver knitro;
AMPL: solve;
KNITRO 8.1.1: Locally optimal solution.
objective 887.414414; feasibility error 7.05e-08
10 iterations; 11 function evaluations
```

Rejected by Gurobi

```ampl
AMPL: model xy4.mod;
AMPL: option solver gurobi;
AMPL: solve;
Gurobi 5.5.0: quadratic objective is not positive definite
```
Product of Linear Terms

AMPL Model

Conic reformulation

```
param n > 0;
param c {1..n} > 0;
param d {1..n} > 0;
var X {1..n} >= 0, <= 2;
var Y {1..n} >= 0, <= 2;
var ZX >= 0;
var ZY >= 0;
var Z;

maximize Obj: Z;

subject to ZXdef: ZX = sum {j in 1..n} c[j]*X[j];
subject to ZYdef: ZY = sum {j in 1..n} d[j]*Y[j];
subject to Zdef: Z^2 <= ZX * ZY;  # still not positive semidefinite
subject to SumX: ........
```
Product of Linear Terms

AMPL Solution

Now solved by Gurobi

```
AMPL: model xy4b.mod;
AMPL: option solver gurobi;
AMPL: solve;
Gurobi 5.5.0: optimal solution; objective 29.78950153
11 barrier iterations
AMPL: print Z^2;
887.4144013356272
```

Related cases

- Minimize can’t be reformulated
- $(\sum_{j=1}^{n} x_j)^{1/2} (\sum_{j=1}^{n} y)^{1/2}$ offers more possibilities
- Many other products of powers can be handled
Example 2: 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

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

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

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

Flow out equals flow in at any intersection

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

Flow into the entrance equals the specified throughput
**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
```
Symbolic model

\[
\begin{align*}
\text{var } & \text{Flow } \{ (i,j) \text{ in ROADS} \} \geq 0, \leq .9999 \times \text{cap}[i,j]; \\
\text{var } & \text{Time } \{ \text{ROADS} \} \geq 0; \\
\text{minimize } & \text{Avg\_Time:} \\
&(\text{sum } \{(i,j) \text{ in ROADS}\} \text{Time}[i,j] \times \text{Flow}[i,j]) / \text{through}; \\
\text{subject to } & \text{Travel\_Time } \{(i,j) \text{ in ROADS}\}: \\
& \text{Time}[i,j] = \text{base}[i,j] + (\text{sens}[i,j] \times \text{Flow}[i,j]) / (1 - \text{Flow}[i,j] / \text{cap}[i,j]); \\
\text{subject to } & \text{Balance\_Node } \{ i \text{ in INTERS}\}: \\
& \text{sum}\{(i,j) \text{ in ROADS}\} \text{Flow}[i,j] = \text{sum}\{(j,i) \text{ in ROADS}\} \text{Flow}[j,i]; \\
\text{subject to } & \text{Balance\_Enter}: \\
& \text{sum}\{(EN,j) \text{ in ROADS}\} \text{Flow}[EN,j] = \text{through};
\end{align*}
\]
Traffic Network

AMPL Data

Explicit data independent of symbolic model

```plaintext
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 ;
```

Traffic Network schematic with nodes a, b, c, d and directed edges a to b, a to c, c to a, c to b, b to d, and d to c.
Traffic Network

AMPL Solution

Model + data = problem to solve, using KNITRO

```
ampl: model traffic.mod;
ampl: data traffic.dat;
ampl: option solver knitro;
ampl: solve;

KNITRO 8.1.1: Locally optimal solution.
objective 61.04695019; feasibility error 1.42e-14
9 iterations; 15 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
```

Traffic Network
Traffic Network

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 12.5.1.0:
Constraint _scon[1] is not convex quadratic
since it is an equality constraint.
```
Traffic Network

AMPL Solution (cont’d)

Look at the model again . . .

```
var Flow {(i,j) in ROADS} >= 0, <= 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 Solution (cont’d)**

**Quadratically constrained reformulation**

```AMPL
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;
```

Traffic Network
Traffic Network

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 12.5.1.0:
QP Hessian is not positive semi-definite.
```
Traffic Network

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;
```
Traffic Network

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 12.5.1.0: 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
;```
Detection and Transformation

Where they are done now
- In AMPL
- In the AMPL-solver interface
- In the solver

Where they should be done

How we have extended them
**In AMPL**

*Model instantiated with data*

*Expression trees written to problem file*

- \((10.1x_2)(5.3y_5 + 1.7y_8)\)
- \((10.1 \times x[2]) \times (5.3 \times y[5] + 1.7 \times y[8])\)
In the AMPL-Solver Interface

Quadratic problem detected
- Products of linear terms multiplied out
- Quadraticity test applied by recursive tree walk

Nonzero quadratic coefficients sent to solver
- Coefficients extracted from tree
- Solver-specific routines called
In the Solver

Test for recognized convex quadratics

“Elliptic” case: numerical test

\[ \text{Min } x^T Q x + q x \] 
\[ x^T Q x \leq q x + c \] \hspace{1cm} \text{where } Q \text{ is positive semidefinite}

“Conic” case: symbolic test

\[ x_1^2 + \ldots + x_n^2 \leq x_{n+1}^2, \quad x_{n+1} \geq 0 \]
\[ x_1^2 + \ldots + x_n^2 \leq x_{n+1} x_{n+2}, \quad x_{n+1} \geq 0, \quad x_{n+2} \geq 0 \]

\ldots second-order cone programs (SOCPs) 

Where Should Detection and Transformation Be Done?

In AMPL?
- Some solution strategies may be ruled out

In the solver?
- Each solver will have its own implementation

In the AMPL-solver interface?
- Recognition routines can be shared where appropriate
- Representation details can be different for each solver
- New ideas can be tried out

... interface source is open
Example 3: Schittkowski #255 (err)

\[
\text{var } x \{1..4\} \geq -20, \leq 20;
\]

\[
\text{minimize } f: \quad 100*(x[2] - x[1]^2) + (1-x[1])^2 + 90*(x[4]-x[3]^2) + (1-x[3])^2 + 10.1*((x[2]-1)^2 + (x[4]-1)^2) + 19.8*(x[2]-1)*(x[4]-1);
\]
s255 (err)

AMPL Solution by KNITRO

Starting point 1

```ampl
ampl: model s255.mod;
ampl: let {j in 1..4} x[j] := -1;
ampl: solve;
KNITRO 8.0.0: Locally optimal solution.
objective -75216.1247; feasibility error 0
7 iterations; 8 function evaluations
```

Starting point 2

```ampl
ampl: model s255.mod;
ampl: let {j in 1..4} x[j] := +1;
ampl: solve;
KNITRO 8.0.0: Locally optimal solution.
objective -75376.125; feasibility error 0
8 iterations; 9 function evaluations
```
AMPL Solution by “Linear” Solvers

Rejected by CPLEX

AMPL: model s255.mod;
AMPL: option solver cplex;
AMPL: solve;
CPLEX 12.5.1.0: QP Hessian is not positive semi-definite.

Solved by Gurobi

AMPL: model s255.mod;
AMPL: option solver gurobi;
AMPL: solve;
Gurobi 5.5.0: optimal solution; objective -75376.125
7 barrier iterations
Detection and Transformation of SOCP-Equivalent Forms

Theory
- Targets for transformation
- SOCP-equivalent forms

Implementation via recursive tree walks
- Detection
- Transformation

Testing
- Existence of SOCP-equivalent problems
- Performance of “linear” vs. nonlinear solvers

Prospects . . .
**Theory: Conic Constraint Forms**

**Standard cone**

\[ x^2 + y^2 \leq z^2 \]

\[ z \geq 0 \]

\[ x^2 + y^2 \leq z^2, \quad z \geq 0 \]

\[ \ldots \text{boundary not smooth} \]

**Rotated cone**

\[ x^2 \leq yz, \quad y \geq 0, z \geq 0, \ldots \]
Targets for Transformation

Symbolic detection

- $x_1^2 + \ldots + x_n^2 \leq x_{n+1}^2$, $x_{n+1} \geq 0$
- $x_1^2 + \ldots + x_n^2 \leq x_{n+1} x_{n+2}$, $x_{n+1} \geq 0$, $x_{n+2} \geq 0$
  * implemented through recursive tree walks

Numerical detection

- $x^T Q x + q x \leq r$, where $Q$ has one negative eigenvalue
  * see Ashutosh Mahajan and Todd Munson, “Exploiting Second-Order Cone Structure for Global Optimization”
  * not addressed in our work
SOCP-Equivalent Forms

**Quadratic**
- Constraints
- Objectives

**SOC-representable**
- Quadratic-linear ratios
- Generalized geometric means
- Generalized $p$-norms

**Other objective functions**
- Generalized product-of-powers
- Logarithmic Chebychev
SOCP-equivalent

Quadratic Generalizations

Standard cone constraints

\[ \sum_{i=1}^{n} a_i (f_i x + g_i)^2 \leq a_{n+1} (f_{n+1} x + g_{n+1})^2, \]
\[ a_1, \ldots, a_{n+1} \geq 0, \quad f_{n+1} x + g_{n+1} \geq 0 \]
\[ \sum_{i=1}^{n} v_i^2 \leq v_{n+1}, \quad v_{n+1} \geq 0 \]
\[ v_i = a_i^{1/2} (f_i x + g_i), \quad i = 1, \ldots, n+1 \]

Rotated cone constraints

\[ \sum_{i=1}^{n} a_i (f_i x + g_i)^2 \leq a_{n+1} (f_{n+1} x + g_{n+1}) (f_{n+2} x + g_{n+2}), \]
\[ a_1, \ldots, a_{n+1} \geq 0, \quad f_{n+1} x + g_{n+1} \geq 0, \quad f_{n+2} x + g_{n+2} \geq 0 \]

Sum-of-squares objectives

\[ \text{Minimize} \ \sum_{i=1}^{n} a_i (f_i x + g_i)^2 \]
\[ \text{Minimize} \ \quad \quad \quad \quad \quad v \]
\[ \text{Subject to} \ \sum_{i=1}^{n} a_i (f_i x + g_i)^2 \leq v^2, \quad v \geq 0 \]
SOC-Representable

**Definition**

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

**Minimization property**

- Minimize $s(x)$ is SOC-equivalent
  * 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-equivalent

SOC-Representable (1)

Vector norm

\[ \|a \cdot (Fx + g)\| = \sqrt{\sum_{i=1}^{n} a_i^2 (f_i x + g_i)^2} \leq a_{n+1}(f_{n+1} x + g_{n+1}) \]

* square both sides to get standard SOC

\[ \sum_{i=1}^{n} a_i^2 (f_i x + g_i)^2 \leq a_{n+1}^2(f_{n+1} x + g_{n+1})^2 \]

Quadratic-linear ratio

\[ \frac{\sum_{i=1}^{n} a_i(f_i x + g_i)^2}{f_{n+2} x + g_{n+2}} \leq a_{n+1}(f_{n+1} x + g_{n+1}) \]

* where \( f_{n+2} x + g_{n+2} \geq 0 \)

* multiply by denominator to get rotated SOC

\[ \sum_{i=1}^{n} a_i (f_i x + g_i)^2 \leq a_{n+1} (f_{n+1} x + g_{n+1})(f_{n+2} x + g_{n+2}) \]
SOCP-equivalent

SOC-Representable (2)

Negative geometric mean

* $-\prod_{i=1}^{p}(f_i x + g_i)^{1/p} \leq f_{n+1} x + g_{n+1}$, $p \in \mathbb{Z}^{+}$
  * $-x_1^{1/4} x_2^{1/4} x_3^{1/4} x_4^{1/4} \leq -x_5$ becomes rotated SOCs:
    $x_5^2 \leq v_1 v_2, v_1^2 \leq x_1 x_2, v_2^2 \leq x_3 x_4$
  * apply recursively $\lfloor \log_2 p \rfloor$ times

Generalizations

* $-\prod_{i=1}^{n}(f_i x + g_i)^{\alpha_i} \leq a_{n+1}(f_{n+1} x + g_{n+1})$: $\sum_{i=1}^{n} \alpha_i \leq 1$, $\alpha_i \in \mathbb{Q}^{+}$
* $\prod_{i=1}^{n}(f_i x + g_i)^{-\alpha_i} \leq a_{n+1}(f_{n+1} x + g_{n+1})$, $\alpha_i \in \mathbb{Q}^{+}$
  * all require $f_i x + g_i$ to have proper sign
SOCP-equivalent

SOC-Representable (3)

\(p\)-norm

\[ (\sum_{i=1}^{n} |f_i x + g_i|^p)^{1/p} \leq f_{n+1} x + g_{n+1}, \quad p \in \mathbb{Q}^+, \quad p \geq 1 \]

\[ (|x_1|^5 + |x_2|^5)^{1/5} \leq x_3 \]

which becomes

\[ v_1 + v_2 \leq x_3 \]

with

\[ -v_1^{1/5} x_3^{4/5} \leq \pm x_1, \quad -v_1^{1/5} x_3^{4/5} \leq \pm x_2 \]

\[ \Rightarrow \]

reduces to product of powers

Generalizations

\[ (\sum_{i=1}^{n} |f_i x + g_i|^\alpha_i)^{1/\alpha_0} \leq f_{n+1} x + g_{n+1}, \quad \alpha_i \in \mathbb{Q}^+, \quad \alpha_i \geq \alpha_0 \geq 1 \]

\[ \sum_{i=1}^{n} |f_i x + g_i|^\alpha_i \leq (f_{n+1} x + g_{n+1})^{\alpha_0} \]

Minimize \( \sum_{i=1}^{n} |f_i x + g_i|^\alpha_i \)

\[ \ldots \] standard SOCP has \( \alpha_i \equiv 2 \]
Other Objective Functions

Unrestricted product of powers

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

Logarithmic Chebychev approximation

- Minimize \(\max_{i=1}^{n}|\log(f_i x) - \log(g_i)|\)

Why no constraint versions?

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

Principles

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

Example: Sum of norms
Principles

Representation

Expression

\[ \text{base}[i,j] + \frac{(\text{sens}[i,j]\times\text{Flow}[i,j])}{(1-\frac{\text{Flow}[i,j]}{\text{cap}[i,j]})} \]

Expression tree

\[ + \quad \frac{5}{\quad \frac{\star}{\quad -}} \]

\[ 0.1 \quad \times[5] \quad 1 \quad \frac{\star}{\quad -} \]

\[ x[5] \quad 10 \]

\[ \ldots \text{actually a DAG} \]
Principles

Detection: isQuadr()

```plaintext
boolean isQuadr (Node);

case of Node {
    PLUS: return( isQuadr(Node.left) and isQuadr(Node.right) );
    MINUS: return( isLinear(Node.left) and isLinear(Node.right) or
                   isQuadr(Node.left) and isConst(Node.right) or
                   isConst(Node.left) and isQuadr(Node.right) );
    TIMES: return( isLinear(Node.left) and
                   isConst(Node.right) and value(Node.right) == 2 );
    POWER: return( TRUE );
    VAR: return( TRUE );
    CONST: return( TRUE );
}
```
**Principles**

**Detection: isLinear()**

```java
boolean isLinear (Node);

case of Node {
  PLUS: return( isLinear(Node.left) and isLinear(Node.right) );
  MINUS: return( isConst(Node.left) and isLinear(Node.right) or
             isLinear(Node.left) and isConst(Node.right) );
  TIMES: return( 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)
Transformation: buildLinear()

\[
\begin{align*}
(coef, const) &= \text{buildLinear}(\text{Node}); \\
\text{if Node.L then} \ (coefL, consL) &= \text{buildLinear}(\text{Node.L}); \\
\text{if Node.R then} \ (coefR, consR) &= \text{buildLinear}(\text{Node.R}); \\
\text{case of Node} \ {\{}
\begin{align*}
\text{PLUS:} & \quad \text{coeff} = \text{mergeLists} (\text{coefL, coefR}); \\
& \quad \text{const} = \text{consL} + \text{consR}; \\
\text{TIMES:} & \quad \ldots \\
\text{DIV:} & \quad \text{coeff} = \text{coefL} / \text{consR}; \\
& \quad \text{const} = \text{consL} / \text{consR}; \\
\text{VAR:} & \quad \text{coeff} = \text{makeList} (1, \text{Node.index}); \\
& \quad \text{const} = 0; \\
\text{CONST:} & \quad \text{coeff} = \text{makeList} (); \\
& \quad \text{const} = \text{Node.value};
\end{align*}
\}
\]

\ldots to transform, call \text{buildLinear}(\text{root})
Example: Sum-of-Norms Objective

Given

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

Transform to

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

Two steps

- Detection
- Transformation
Detection

**SUMOFNORMS**

**Sum:** $e_1 + e_2$ is **SUMOFNORMS** if $e_1, e_2$ are **SUMOFNORMS**

**Product:** $e_1 e_2$ is **SUMOFNORMS** if
- $e_1$ is **SUMOFNORMS** and $e_2$ is **POSCONSTANT** or
- $e_2$ is **SUMOFNORMS** and $e_1$ is **POSCONSTANT**

**Square root:** $\sqrt{e}$ is **SUMOFNORMS** if $e$ is **SUMOF SQUARES**

**SUMOF SQUARES**

**Sum:** $e_1 + e_2$ is **SUMOF SQUARES** if $e_1, e_2$ are **SUMOF SQUARES**

**Product:** $e_1 e_2$ is **SUMOF SQUARES** if
- $e_1$ is **SUMOF SQUARES** and $e_2$ is **POSCONSTANT** or
- $e_2$ is **SUMOF SQUARES** and $e_1$ is **POSCONSTANT**

**Square:** $e^2$ is **SUMOF SQUARES** if $e$ is **LINEAR**

**Constant:** $c$ is **SUMOF SQUARES** if $c$ is **POSCONSTANT**
**Sum of Norms**

**Detection Issues**

**Mathematical**

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

**Practical**

- Constant multiples inside any sum
- Recursive nesting of constant multiples & sums
- Constant as a special case of a square

\[
\star \sqrt{3(4x_1 + 7(x_2 + 2x_3) + 6)^2 + (x_4 + x_5)^2 + 17}
\]
Sum of Norms
Transformation

**TRANSFORMSUMOFNORMS** (*Expr e, Obj o, real k*)

**Sum:** \( e_1 + e_2 \) where \( e_1, e_2 \) are **SUMOFNORMS**
  - **TRANSFORMSUMOFNORMS** (*e_1, o, k*)
  - **TRANSFORMSUMOFNORMS** (*e_2, o, k*)

**Product:** \( e_1 \cdot c_2 \) where \( e_1 \) is **SUMOFNORMS** and \( c_2 \) is **POSCONSTANT**
  - **TRANSFORMSUMOFNORMS** (*e_1, o, c_2 \cdot k*)

**Product:** \( c_1 \cdot e_2 \) where \( e_2 \) is **SUMOFNORMS** and \( c_1 \) is **POSCONSTANT**
  - **TRANSFORMSUMOFNORMS** (*e_2, o, c_1 \cdot k*)

**Square root:** \( \sqrt{e} \) where \( e \) is **SUMOFSQUARES**
  - \( y_i := \text{NEWNONNEGVAR}(); o += k \cdot y_i \)
  - \( q_i := \text{NEWLECON}(); q_i += -y_i^2 \)
  - **TRANSFORMSUMOFSQUARES** (*e, q_i, 1*)
**Sum of Norms**

**Transformation (cont’d)**

**TRANSFORMSUMOFSQUARES** *(Expr e, LeCon qi, real k)*

**Sum:** \( e_1 + e_2 \) where \( e_1, e_2 \) are **SUMOFSQUARES**
- \( \text{TRANSFORMSUMOFSQUARES} (e_1, o, k) \)
- \( \text{TRANSFORMSUMOFSQUARES} (e_2, o, k) \)

**Product:** \( e_1 \times c_2 \) where \( e_1 \) is **SUMOFSQUARES** and \( c_2 \) is **POSCONSTANT**
- \( \text{TRANSFORMSUMOFSQUARES} (e_1, o, c_2 \times k) \)

**Product:** \( c_1 \times e_2 \) where \( e_2 \) is **SUMOFSQUARES** and \( c_1 \) is **POSCONSTANT**
- \( \text{TRANSFORMSUMOFSQUARES} (e_2, o, c_1 \times k) \)

**Square:** \( \text{sqr}(z_{ij}) \) where \( z_{ij} \) is **VARIABLE**
- \( q_i \leftarrow k \times z_{ij}^2 \)

**Square:** \( \text{sqr}(e) \) where \( e \) is **LINEAR**
- \( z_{ij} := \text{NEWVAR}(); \ q_i \leftarrow k \times z_{ij}^2 \)
- \( l_{ij} := \text{NEWEQCON}(); \ l_{ij} \leftarrow z_{ij} - e \)

**Constant:** \( c \) is **POSCONSTANT**
- \( z_{ij} := \text{NEWVAR}(); \ q_i \leftarrow k \times z_{ij}^2 \)
- \( l_{ij} := \text{NEWEQCON}(); \ l_{ij} \leftarrow z_{ij} - \text{sqrt}(c) \)
Sum of Norms

Transformation Issues

Mathematical

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

Practical

- Generalization: handle all previously mentioned
- Efficiency: don’t define $z_{ij}$ when $f_{ij} x + g_{ij}$ is a single variable
- Trigger by calling $\text{TRANSFORMSUMOFNORMS}(e, o, k)$ with
  * $e$ 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 . . .
Testing

Survey of nonlinear test problems

Comparison of performance
Survey of Test Problems (1)

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 $-x_1x_2x_3$
- 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 $\sum_{k=1}^2 (x_{jk} - a_{ik})^2 \leq s_{ij}^2$
Survey of Test Problems (2)

*Counted number of test problems* . . .

- Solvable already by a “linear” solver
- Detected as SOCP-equivalent by our routines

![Venn Diagram]

- CPLEX: 66, 21, 80
- SOCP-equiv: 80

Robert Fourer, Jared Erickson, Convex Quadratic Programming in AMPL
ICCOPT 2013 — Lisbon 29 July-1 August 2013
Comparison of Performance

SOCP-equivalent with nonsmooth functions

```
var x {1..5} integer;
var y {1..5} >= 0;

minimize obj: sum {i in 1..5} ( sqrt( (x[i]+2)^2 + (y[i]+1)^2 ) + sqrt( (x[i]+y[i])^2 ) + y[3]^2 );

subj to xsum: sum {i in 1..5} x[i] <= -12;
subj to ysum: sum {i in 1..5} y[i] >= 10;

subj to socprep:
    max {i in 1..5} ( (x[i]^2 + 1)/(i+y[i]) + y[i]^3 ) <= 30;
```
Comparison (cont’d)

General nonlinear solver (integer)

KNITRO 8.0.0: Convergence to an infeasible point.
Problem may be locally infeasible.

General nonlinear solver (continuous relaxation)

KNITRO 8.0.0:
--- ERROR evaluating objective gradient.
--- ERROR evaluating constraint gradients.
Evaluation error.
objective 17.14615551; feasibility error 0
233 iterations; 1325 function evaluations
Comparison

Convex quadratic solver (integer)

CLEX 12.4.0
Total time (root+branch&cut) = 0.21 sec.
Solution value = 17.246212

: x y
1 -3 3
2 -2 1.99993
3 -2 0.000300084
4 -3 3
5 -2 1.99993
;

Comparison

Convex quadratic solver (continuous relaxation)

CPLEX 12.4.0
Total time = 0.04 sec.
Solution value  = 17.141355

:   x         y
 1   -2.49707 2.49707
 2   -2.49707 2.49707
 3   -2.01171 0.011716
 4   -2.49707 2.49707
 5   -2.49707 2.49707
;
Prospects

Robust implementation needed
- Focus on forms most likely to be worthwhile
- Modularize to work with varied solver interfaces

Value will be established gradually
- Teach users about convex quadratic features
- Collect experience with new models