The Surprisingly Complicated Case of [Convex] Quadratic Optimization

Robert Fourer

4er@ampl.com

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

U.S.-Mexico Workshop on
Optimization and Its Applications

Mérida — 4-8 January 2016

The Surprisingly Complicated Case of Convex Quadratic Optimization

The convex quadratic case sits at the boundary between easy (linear, convex) and hard (nonlinear, nonconvex) optimization problems. Perhaps for this reason it gives rise to an unexpectedly large number of complications in modeling. It is not always clear when a nonlinear problem can be can be converted to convex quadratic, when a quadratic problem is best transformed to linear, or even when a quadratic problem has a convex formulation. The difficulties multiply when one admits conic as well as elliptic quadratic constraints, and when discrete as well as continuous variables are involved. This presentation surveys a variety of challenges, with emphasis on their implications for improved design of modeling software.

Motivation

From my mailbox (1) . . .

```
I am trying to solve a quadratically constrained quadratic program
that is written in AMPL.

For a particular configuration of the parameters I get this output
when trying to solve with CPLEX:

The return code is 501 with the description "failure."

Any idea what's going on here?
```

```
In my model I have the following:

Qij= Variable to decide the number of products to send from i to j;
Xij = binary variable to decide if I send products from i to j

Max= Qij*Xij;

Because I am multiplying two variables this problem becomes a
quadratic problem, I would like to know if you could recomend me
any solver to solve this quadratic problem
```

Motivation

From my mailbox (2) . . .

```
Sorry. I mad mistake. C and f are positive non-integer variables.
They can't be integer given the problem formulation.

If so. Will I be able to use Cplex to solve this problem
given this quadratic constraint like f[i]*C[i] <= 1 ??
```

```
I have faced with a similar problem but I cannot understand the
difference between these two options:

1 - This case works fine in ampl as it is defined as a
predefined var: ...

2 - But if I do the above as a constraint, (I would like to do it
this way because I can have my constraint working also with glpsol,
as glpsol doesn't have the predefined var) ampl give the error of
not convex quadratic: ...

Why would ampl treat these two statements differently?
```

Motivation

Kinds of Large-Scale Solvers

Linear

- CPLEX, Gurobi, Xpress, MOSEK, SCIP, CBC, . . .

Quadratic

- For linear solvers, an *extension*
- For nonlinear solvers, a *special case*

Nonlinear

- Knitro, MINOS, CONOPT, SNOPT, Ipopt, . . .

Motivation

Kinds of Expressions

Linear

- ❖ `V + log(q) * sum {j in 1..n} (a[j] + c[j] * X[j])`

Quadratic

- ❖ `sum {j in 1..n} c[j] * X[j]^2`
- ❖ `a[j] * (sum {j in 1..n} c[j] * X[j])^2`
- ❖ `(sum {j in 1..n} X[j]) * (sum {j in 1..n} Y[j])`

Nonlinear

- ❖ `log(V) + sum {j in 1..n} sin(a[j] / c[j] * X[j])`

Motivation

Conveying Expressions to Solvers

Linear expression mechanism

- coefficient lists

Nonlinear expression mechanism

- expression trees

Quadratic expression mechanism

- coefficient lists extracted from expression trees

Conveying Expressions

Linear Mechanism

AMPL . . .

- verifies linearity of expressions
- writes lists of nonzero coefficients

AMPL interface . . .

- sends a linear coefficient list to the solver

Solver . . .

- applies a linear algorithm

Conveying Expressions

Nonlinear Mechanism

AMPL . . .

- writes nonlinear expression trees

AMPL-solver interface . . .

- sets up nonlinear function evaluation data structure
- invokes the solver

Solver . . .

- applies a general nonlinear algorithm
- *calls back to the AMPL-solver interface* to evaluate functions and derivatives at successive points

. . . some solvers (like MINOS) use both mechanisms

Quadratic Mechanism

AMPL . . .

- writes nonlinear expression trees

AMPL interface . . .

- verifies quadraticity of expressions
- extracts coefficients of quadratic terms
- sends a coefficient list to the solver

Solver . . .

- analyzes and transforms quadratic functions
- applies an appropriate linear or *extended linear* algorithm

Difficulties & Challenges

Difficulties of detection

- What kind of optimization problem is this?

Difficulties of transformation

- Can this be transformed to an easier quadratic problem?
- Can this be transformed to an easier linear problem?

Challenges of algorithmic choice

- What algorithmic approach should be applied?

A variety of cases to consider . . .

Survey

Continuous

- Elliptic quadratic objectives and constraints
- Nonconvex quadratic objectives
- Conic quadratic constraints

Discrete

- Integer convex quadratic constraints
- Binary quadratic objectives

Convex ^ "Elliptic" Quadratics

Formulation

- Minimize $x^T Q x + q x$
- Subject to $x_k^T Q_k x_k \le q_k x + c_k$

Detection (numerical)

- Q, Q_k must be positive semi-definite: numerical test on quadratic coefficients

Optimization

- extension to linear simplex method (objective only)
- extension to linear interior-point method

Nonconvex Quadratic Objectives

Formulation

- Minimize $x^T Q x + qx$

Detection (numerical)

- Q not positive semi-definite

Optimization

- impossible
- local
- global

Linear Solver

CPLEX Option 1 (default): rejected

```
ampl: model nonconvquad.mod;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.6.2.0: QP Hessian is not positive semi-definite.
```

CPLEX Option 2: local optimum

```
ampl: option cplex_options 'reqconvex 2'; solve;
CPLEX 12.6.2.0: locally optimal solution of indefinite QP;
   objective 12.62598015
164 QP barrier iterations
_solve_elapsed_time = 0.219
```

Nonconvex Quadratics

Linear Solver *(cont'd)*

CPLEX Option 2: local optimum

```
ampl: option cplex_options 'reqconvex 2'; solve;

CPLEX 12.6.2.0: locally optimal solution of indefinite QP;
   objective 12.62598015

164 QP barrier iterations

_solve_elapsed_time = 0.219
```

CPLEX Option 3: global optimum

```
ampl: option cplex_options 'reqconvex 3'; solve;

CPLEX 12.6.2.0: optimal integer solution;
   objective 0.1387763988

479250 MIP simplex iterations
11114 branch-and-bound nodes

_solve_elapsed_time = 352.203
```

Nonconvex Quadratics

Local Nonlinear Solver

Knitro (default)

```
ampl: option solver knitro; solve;

KNITRO 9.1.0: Locally optimal solution.

objective 5.985858772; feasibility error 6.39e-14
45 iterations; 53 function evaluations

_solve_elapsed_time = 0.328
```

Knitro multistart: 100 solves

```
ampl: option knitro_options
   'ms_enable 1 ms_maxsolves 100 par_numthreads 2'; solve;

KNITRO 9.1.0: Locally optimal solution.

objective 0.24752033; feasibility error 2.13e-14
3763 iterations; 4163 function evaluations

_solve_elapsed_time = 2.484
```

Local Nonlinear Solver *(cont'd)*

Knitro multistart: 100 solves

```
ampl: option knitro_options
   'ms_enable 1 ms_maxsolves 100 par_numthreads 2'; solve;

KNITRO 9.1.0: Locally optimal solution.

objective 0.24752033; feasibility error 2.13e-14
3763 iterations; 4163 function evaluations

_solve_elapsed_time = 2.484
```

Knitro multistart: 1000 solves

```
ampl: option knitro_options
   'ms_enable 1 ms_maxsolves 1000 par_numthreads 2'; solve;

KNITRO 9.1.0: Locally optimal solution.

objective 0.1387772422; feasibility error 7.11e-15
39008 iterations; 43208 function evaluations

_solve_elapsed_time = 31.109
```

Nonconvex Quadratics

Global Nonlinear Solver

BARON

```
ampl: option solver baron; solve;

BARON 15.9.22 (2015.09.22):

1871 iterations, optimal within tolerances.
Objective 0.1387763988

_solve_elapsed_time = 287.484
```

Convex ^ "Conic" Quadratics (SOCPs)

Second-order cone formulation

- $\|(x_1, \ldots, x_n)\|_2 \le x_{n+1}$

Quadratic formulation

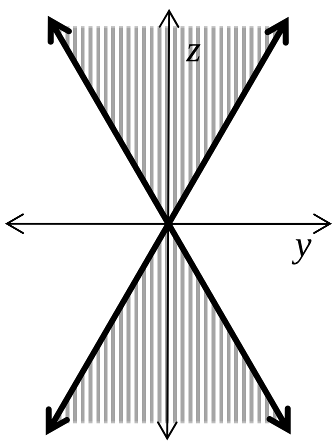
- $x_1^2 + \ldots + x_n^2 \le x_{n+1}^2,\ \ x_{n+1} \ge 0$

Conic Quadratics

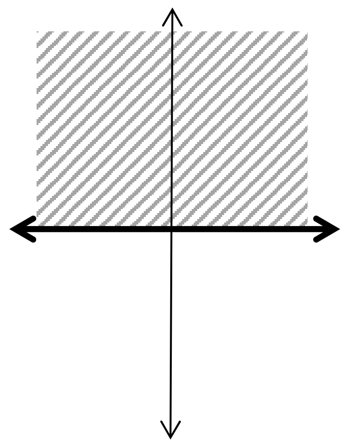
Geometry

Standard cone

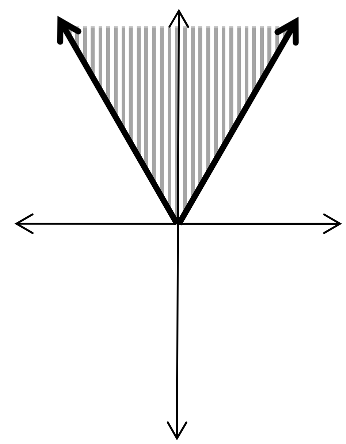
$x^2 + y^2 \le z^2$



$z \ge 0$



$x^2 + y^2 \le z^2,\ z \ge 0$



. . . boundary not smooth

Rotated cone

- $x^2 \le yz,\ y \ge 0, z \ge 0, \ldots$

“Conic” Quadratics (SOCPs)

Formulations

- $x_1^2 + \ldots + x_n^2 \le x_{n+1}^2,\ x_{n+1} \ge 0$
- $x_1^2 + \ldots + x_n^2 \le x_{n+1}\, x_{n+2},\ x_{n+1} \ge 0,\ x_{n+2} \ge 0$

Detection (symbolic)

- recognized pattern of simple quadratic terms
  (details vary by solver)

Optimization

- extension to linear interior-point method

Conic Quadratics

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

Formulation

Determine

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

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

to minimize

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

Average travel time from e to f

Formulation *(cont'd)*

Subject to

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

Travel times increase as flow approaches capacity

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

Flow out equals flow in at any intersection

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

Flow into the entrance equals the specified throughput

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
```

Traffic Network

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;
```

Example: Traffic Network

AMPL 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

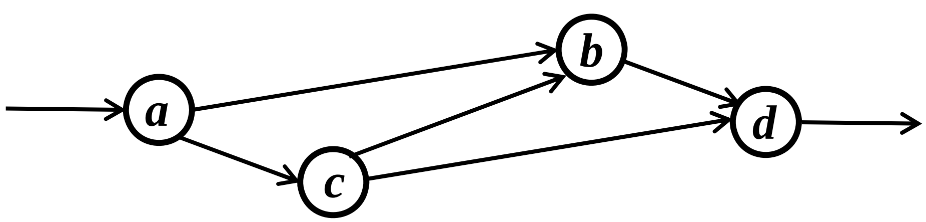
Explicit data independent of symbolic model

```
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 ;
```



Traffic Network

Linear Solver

Model + data = problem to solve, using Gurobi?

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 6.5.0:
Gurobi can't handle nonquadratic nonlinear constraints.
```

Linear Solver *(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;
```

Traffic Network

Linear Solver *(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;
```

Traffic Network

Linear Solver *(cont'd)*

Model + data = problem to solve, using Gurobi?

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 6.5.0:
quadratic constraint is not positive definite
```

Traffic Network

Linear Solver *(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;
```

Linear Solver *(cont'd)*

Model + data = problem to solve, using Gurobi!

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 6.5.0: optimal solution; objective 61.0469834
53 barrier iterations

ampl: display Flow;

Flow :=
a b    9.5521
a c   10.4479
b d   11.0044
c b    1.45228
c d    8.99562
;
```

Traffic Network

Local Nonlinear Solver

Model + data = problem to solve, using Knitro

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

ampl: option solver knitro;
ampl: solve;

Knitro 10.0.0: Locally optimal solution.
objective 61.04695019; feasibility error 3.18e-09
11 iterations; 21 function evaluations

ampl: display Flow;

Flow :=
a b    9.55146
a c   10.4485
b d   11.0044
c b    1.45291
c d    8.99562
;
```

Global Nonlinear Solver

Model + data = problem to solve, using BARON

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

ampl: option solver baron;
ampl: solve;

BARON 15.9.22 (2015.09.22):
1 iterations, optimal within tolerances.
Objective 61.04695019

ampl: display Flow;

Flow :=
a b    9.55146
a c   10.4485
b d   11.0044
c b    1.45291
c d    8.99562
;
```

SOCP-Solvable Forms

Quadratic

- Elliptic
- Conic

SOC-representable

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

Other objective functions

- Generalized product-of-powers
- Logarithmic Chebychev

Jared Erickson and Robert Fourer,
Detection and Transformation of Second-Order Cone Programming Problems in a General-Purpose Algebraic Modeling Language

SOCP-solvable

"Elliptic" Quadratic (1)

Original constraint formulation

- $x_k^T Q_k x_k \le q_k x + c_k,\ \ Q_k \succcurlyeq 0$

Conic reformulation

- $w_k^T w_k \le y_k z_k$
- $w_k = Q_k^{1/2} x_k$
- $y_k = q_k x + c_k$
- $z_k = 1$

. . . preferred by some solvers (like MOSEK)

SOCP-solvable

“Elliptic” Quadratic (2)

Original objective formulation

- Minimize $x^T Q x + qx,\ \ Q \succcurlyeq 0$

Conic reformulation

- Minimize $v + qx$
- Subject to $x^T Q x \leq v$

. . . then apply the constraint reformulation

SOCP-solvable

Conic Quadratic (1)

Generalized cone constraints

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

Symbolic detection

- Convert to simpler formulation before sending to solver
- $\sum_{i=1}^{n} y_i^2 \leq y_{n+1}^2,\ y_{n+1} \geq 0$
  $y_i = a_i^{1/2}(\mathbf{f}_i\mathbf{x} + g_i),\ i = 1, \ldots, n+1$

Numerical detection

- Multiply out and send quadratic coefficients to solver
- Apply numerical test in solver to detect conic form
- *Practicality uncertain!*

Ashutosh Mahajan and Todd Munson,
"Exploiting Second-Order Cone Structure for Global Optimization"

SOCP-solvable

Conic Quadratic (2)

Generalized cone constraints

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

Generalized rotated cone constraints

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

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

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

Quadratic-linear ratio

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

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:
    $x_5^2 \le v_1 v_2,\ v_1^2 \le x_1 x_2,\ v_2^2 \le x_3 x_4$
  - 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 $\mathbf{f}_i\mathbf{x}+g_i$ to have proper sign

SOCP-solvable

SOC-Representable (3)

p-norm

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

Generalizations

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

Integer Conic Constraints

Formulation

- Linear objective
- Conic quadratic constraints
- Some integer-valued variables

Detection

- Check for conic quadratic & look for integer variables

Optimization

- branch-and-bound with quadratic relaxations
- outer approximation:
  branch-and-bound with linear relaxations

Traffic Network

Linear Integer Solver

CPLEX with quadratic relaxations

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

ampl: option solver cplex;
ampl: option cplex_options 'miqcpstrat 1';
ampl: solve;

CPLEX 12.6.2.0: optimal (non-)integer solution; objective 76.26375004

20 MIP simplex iterations
0 branch-and-bound nodes

3 integer variables rounded (maxerr = 1.92609e-06).
Assigning integrality = 1e-06 might help.
Currently integrality = 1e-05.
```

Traffic Network

Linear Integer Solver *(cont'd)*

CPLEX with linear relaxations

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

ampl: option solver cplex;
ampl: option cplex_options 'miqcpstrat 2';
ampl: solve;

CPLEX 12.6.2.0: optimal integer solution within mipgap or absmipgap;
   objective 76.26375017

19 MIP simplex iterations
0 branch-and-bound nodes

absmipgap = 4.74295e-07, relmipgap = 6.21914e-09


ampl: display Flow;

:   b   c    d
a   9   11   .
b   .   .    11
c   2   .     9
```

Integer Conic

Disaggregation

One conic constraint with n terms

- $x_1^2 + \ldots + x_n^2 \le x_{n+1}^2,\ x_{n+1} \ge 0$

Transformations

- $x_1(x_1/x_{n+1}) + \ldots + x_n(x_n/x_{n+1}) \le x_{n+1}$
- $y_1 + \ldots + y_n \le x_{n+1}$
  $x_1(x_1/x_{n+1}) \le y_1,\ \ldots,\ x_n(x_n/x_{n+1}) \le y_n$

n conic constraints with one term each

- $y_1 + \ldots + y_n \le x_{n+1}$
  $x_1^2 \le x_{n+1}y_1,\ \ldots,\ x_n^2 \le x_{n+1}y_n$

Disaggregation

One conic constraint with n terms

- $x_1^2 + \ldots + x_n^2 \le x_{n+1}^2,\ x_{n+1} \ge 0$

n conic constraints with one term each

- $y_1 + \ldots + y_n \le x_{n+1}$
  $x_1^2 \le x_{n+1} y_1,\ \ldots,\ x_n^2 \le x_{n+1} y_n$

Advantageous when . . .

- Some variables are integral
- Branch-and-bound uses linear relaxations
- Conic constraints are "long enough"

. . . automated by some solvers (like CPLEX, Gurobi)

Extended Formulations in Mixed Integer Conic Quadratic Programming. J. P. Vielma, I. Dunning, J. Huchette and M. Lubin

Binary Quadratic Objective

Formulation

- Minimize $x^T Q x + qx$
- Subject to linear constraints

Detection

- Variables are binary: $x_j \in \{0,1\}$

Optimization

- *if convex,*
  branch-and-bound with convex quadratic subproblems
- *conversion to linear* followed by
  branch-and-bound with linear subproblems

. . . *replace* $\boldsymbol{x_i x_j}$ *by* $\boldsymbol{y_{ij} \geq x_i + x_j - 1}$

Binary Quadratic

Case 1: Convex

Sample model . . .

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

var X {1..n} binary;

minimize Obj:
   (sum {j in 1..n} c[j]*X[j])^2;

subject to SumX: sum {j in 1..n} j * X[j] >= 50*n+3;
```

Binary Quadratic

Case 1 *(cont'd)*

CPLEX 12.5

```
ampl: solve;

.......

Cover cuts applied:  2
Zero-half cuts applied:  1

.......

Total (root+branch&cut) = 0.42 sec.

CPLEX 12.5.0: optimal integer solution within mipgap or absmipgap;
objective 29576.27517

286 MIP simplex iterations
102 branch-and-bound nodes
```

(n = 200)

Binary Quadratic

Case 1 *(cont'd)*

CPLEX 12.6

```
ampl: solve;

MIP Presolve added 39800 rows and 19900 columns.
Reduced MIP has 39801 rows, 20100 columns, and 79800 nonzeros.
Reduced MIP has 20100 binaries, 0 generals, and 0 indicators.

.......

Cover cuts applied:  8
Zero-half cuts applied:  5218
Gomory fractional cuts applied:  6

.......

Total (root+branch&cut) = 2112.63 sec.

CPLEX 12.6.0: optimal integer solution; objective 29576.27517

474330 MIP simplex iterations
294 branch-and-bound nodes
```

Binary Quadratic

Case 1: Transformations Performed

CPLEX 12.5

- None needed

CPLEX 12.6

- Define a (binary) variable for each term $x_i x_j$
- Introduce $O(n^2)$ new variables and constraints

. . . option for 12.5 behavior added to 12.6.1

Binary Quadratic

Case 2: Nonconvex

Sample model . . .

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

var X {1..n} binary;
var Y {1..n} binary;

minimize 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] >= 2*n+3;
subject to SumY: sum {j in 1..n} j * Y[j] >= 2*n+3;

subject to SumXY: sum {j in 1..n} (X[j] + Y[j]) = n;
```

Binary Quadratic

Case 2 *(cont'd)*

CPLEX 12.5

```
ampl: solve;

Repairing indefinite Q in the objective.

. . . . . . .

Total (root+branch&cut) = 1264.34 sec.

CPLEX 12.5.0: optimal integer solution within mipgap or absmipgap;
objective 290.1853405

23890588 MIP simplex iterations
14092725 branch-and-bound nodes
```

(n = 50)

Binary Quadratic

Case 2 *(cont'd)*

CPLEX 12.6

```
ampl: solve;

MIP Presolve added 5000 rows and 2500 columns.
Reduced MIP has 5003 rows, 2600 columns, and 10200 nonzeros.
Reduced MIP has 2600 binaries, 0 generals, and 0 indicators.

. . . . . . .

Total (root+branch&cut) = 6.05 sec.

CPLEX 12.6.0: optimal integer solution; objective 290.1853405

126643 MIP simplex iterations
1926 branch-and-bound nodes
```

Binary Quadratic

Case 2: Transformations Performed

CPLEX 12.5

- Add $M_j(x_j^2 - x_j)$ to objective as needed to convexify

CPLEX 12.6

- Define a (binary) variable for each term $x_i y_j$
- Introduce $O(n^2)$ new variables and constraints

Binary Quadratic

Case 3: Nonconvex *revisited*

Alternative quadratic model . . .

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

var X {1..n} binary;
var Y {1..n} binary;
var Ysum;

minimize Obj:
   (sum {j in 1..n} c[j]*X[j]) * Ysum;

subj to YsumDefn: Ysum = sum {j in 1..n} d[j]*Y[j];

subject to SumX: sum {j in 1..n} j * X[j] >= 2*n+3;
subject to SumY: sum {j in 1..n} j * Y[j] >= 2*n+3;

subject to SumXY: sum {j in 1..n} (X[j] + Y[j]) = n;
```

Binary Quadratic

Case 3 *(cont'd)*

CPLEX 12.5

```
ampl: solve;
CPLEX 12.5.0: QP Hessian is not positive semi-definite.
```

Binary Quadratic

Case 3 *(cont'd)*

CPLEX 12.6

```
ampl: solve;

MIP Presolve added 100 rows and 50 columns.
Reduced MIP has 104 rows, 151 columns, and 451 nonzeros.
Reduced MIP has 100 binaries, 0 generals, and 0 indicators.
.......
Total (root+branch&cut) = 0.17 sec.

CPLEX 12.6.0: optimal integer solution; objective 290.1853405

7850 MIP simplex iterations
1667 branch-and-bound nodes
```

Binary Quadratic

Case 3: Transformations Performed

Human modeler

- Introduce a (general) variable $y_{\text{sum}} = \sum_{j=1}^{n} d_j y_j$

CPLEX 12.5

- Reject problem as nonconvex

CPLEX 12.6

- Define a (general integer) variable for each term $x_i\, y_{\text{sum}}$
- Introduce $O(n)$ new variables and constraints

F. Glover and E. Woolsey,
Further reduction of zero-one polynomial programming
problems to zero-one linear programming problems (1973)

Binary Quadratic

Case 3: Well-Known Approach

Many refinements and generalizations

- F. Glover and E. Woolsey, Further reduction of zero-one polynomial programming problems to zero-one linear programming problems. *Operations Research* 21 (1973) 156-161.
- F. Glover, Improved linear integer programming formulations of nonlinear integer problems. *Management Science* 22 (1975) 455-460.
- M. Oral and O. Kettani, A linearization procedure for quadratic and cubic mixed-integer problems. *Operations Research* 40 (1992) S109-S116.
- W.P. Adams and R.J. Forrester, A simple recipe for concise mixed 0-1 linearizations. *Operations Research Letters* 33 (2005) 55-61.

Binary Quadratic

Case 4: Nonconvex *reconsidered*

Model with "indicator" constraints . . .

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

var X {1..n} binary;
var Y {1..n} binary;
var Z {1..n};

minimize Obj: sum {i in 1..n} Z[i];

subj to ZDefn {i in 1..n}:
   X[i] = 1 ==> Z[i] = c[i] * sum {j in 1..n} d[j]*Y[j]
           else Z[i] = 0;

subject to SumX: sum {j in 1..n} j * X[j] >= 2*n+3;
subject to SumY: sum {j in 1..n} j * Y[j] >= 2*n+3;

subject to SumXY: sum {j in 1..n} (X[j] + Y[j]) = n;
```

Binary Quadratic

Case 4 *(cont'd)*

CPLEX 12.6 transforms to linear MIP

```
ampl: solve;

Reduced MIP has 53 rows, 200 columns, and 2800 nonzeros.
Reduced MIP has 100 binaries, 0 generals, and 100 indicators.
.......

Total (root+branch&cut) = 5.74 sec.

CPLEX 12.6.0: optimal integer solution within mipgap or absmipgap;
   objective 290.1853405

377548 MIP simplex iterations
95892 branch-and-bound nodes
```

Binary Quadratic

Case 4: Transformations Performed

Human modeler

- Define a (general) variable for each term $x_i \sum_{j=1}^{n} d_j y_j$
- Introduce $O(n)$ new variables
- Introduce $O(n)$ new indicator constraints

CPLEX 12.6

- Enforce indicator constraints in branch and bound?
- Transform indicator constraints to linear ones?

Goals

Handle quadratics more automatically

- Given a quadratic problem and a solver choice
- Decide how best to solve

Suggest appropriate solvers

- Given a quadratic problem
- Identify appropriate solvers to try

Who Should Do the Work?

The modeler

The modeling language processor

The solver interface

The solver

The Modeler

Advantages

- Can exploit special knowledge of the problem
- Doesn't have to be programmed

Disadvantages

- May not know the best way to transform
- May have better ways to use the time
- Can make mistakes

The Modeling Language Processor

Advantages

- Makes the same transformation available to all solvers
- Has a high-level view of the problem

Disadvantages

- Is a very complicated program
- Can't take advantage of special solver features

The Solver Interface

Advantages

- Works on simplified problem instances
- Can use same ideas for many solvers, *but also*
- Can tailor transformation to solver features

Disadvantages

- Creates an extra layer of complication

The Solver

Advantages

- Ought to know what's best for it
- Can integrate transformation with other activities

Disadvantages

- May not incorporate best practices
- Is complicated enough already