# Choosing a Solution Strategy for Discrete Quadratic Optimization

*Robert Fourer*

**4er@ampl.com**

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

## INFORMS Annual Meeting

Nashville — 13-16 November 2016 — Session TD13
*Software and Methodologies for (Nonlinear) Integer Programming*

# Choosing a Solution Strategy for Discrete Quadratic Optimization

The combination of integer variables with quadratic objectives and constraints is a powerful formulation tool. But when it comes to solving the resulting optimization problems, there are numerous good approaches but no one best way — even in simpler cases where the objective is convex or the constraints are linear. Both linearization of quadratic terms and quadratic generalization of linear methods turn out to be preferable in some circumstances. This presentation exhibits a variety of examples to illustrate the questions that should be asked and the decisions that must be made in choosing an effective formulation and solver.

# Solvers for Discrete Quadratic

## *Alternatives*

- Linear/quadratic mixed-integer
  - CPLEX, Gurobi, Xpress
- Local nonlinear mixed-integer
  - Knitro
- Global nonlinear mixed-integer
  - BARON

## *Focus of this talk:* *Linear/quadratic mixed-integer*

- Most efficient, due to specialization
- Limited to convex constraints

# Outline

## *Quadratic objective*

- Binary convex
- Binary nonconvex
- Binary × general nonconvex
- Binary logic
- General nonconvex

## *Convex quadratic constraints*

- Conic inequality
- Elliptic inequality
- Disaggregated conic

# Quadratic Objective

## *General form*

- $x^T Q x + q x$

## *Convex case*

- $Q$ positive semi-definite
- Test *numerically* using elimination on $Q$

*Quadratic Objective*

# Binary 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;
```

*Quadratic Objective*

# Binary Convex *(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)***

*Quadratic Objective*

# Binary Convex *(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
```

*Quadratic Objective*

# Binary Convex Strategies

## *Quadratic branch-and-bound (CPLEX 12.5)*

- Solve a continuous QP at each node

## *Conversion to linear (CPLEX 12.6)*

- Replace each objective term $x_i x_j$ by binary $y_{ij} \geq x_i + x_j - 1$
- Solve a larger continuous LP at each node

***. . . option for 12.5 behavior added to 12.6.1***

# Binary 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 {i in 1..n} c[i]*X[i]) * (sum {j in 1..n} d[j]*Y[j]);

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

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

*Quadratic Objective*

# Binary Nonconvex *(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)***

*Quadratic Objective*

# Binary Nonconvex *(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
```

*Quadratic Objective*

# Binary Nonconvex Strategies

## *Conversion to convex quadratic (CPLEX 12.5)*

- Add $M_j(x_j^2 - x_j)$ to objective as needed to convexify
- Solve a continuous QP at each node

## *Conversion to linear (CPLEX 12.6)*

- Replace each objective term $x_i x_j$ by binary $y_{ij} \geq x_i + x_j - 1$
- Solve a larger continuous LP at each node

. . . ***algorithms same as before***

*Quadratic Objective*

# Binary × General Nonconvex

## *Reformulation of 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;
var Ysum;

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

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

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

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

*Quadratic Objective*

# Binary × General Nonconvex *(cont'd)*

## *CPLEX 12.5*

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

*Quadratic Objective*

# Binary × General Nonconvex *(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
```

*Quadratic Objective*

# Binary × General Nonconvex Strategies

## *Conversion to binary × general linear*

- Replace sum of binaries by general $y_{\text{sum}} = \sum_{j=1}^{n} d_j y_j$
- Replace each objective term $x_i y_{\text{sum}}$ by
  $z_i \geq L x_i$, $z_i \geq y_{\text{sum}} - U(1 - x_i)$, where $L \leq y_{\text{sum}} \leq U$
- Introduce fewer but more complex variables, constraints

## *Many refinements and generalizations*

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

*Quadratic Objective*

# Binary Logic

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

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

*Quadratic Objective*

# Binary Logic *(cont'd)*

## *CPLEX 12.6 handles using linear MIP techniques*

```
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 Logic Strategies

## *Reversion to implied logic*

- Replace each objective term $(c_i x_i) \sum_{j=1}^{n} d_j y_j$ by general $z_i$
- Add disjunctive conditions
  - $x_i = 0$ and $z_i = 0$
  - $x_i = 1$ and $z_i = c_i \sum_{j=1}^{n} d_j y_j$

## *Solution by branch-and-bound*

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

*Quadratic Objective*

# General Nonconvex

*Neither integer variable is binary*

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

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

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

*Quadratic Objective*

# General Nonconvex *(cont'd)*

## *CPLEX default setting*

```
ampl: solve;

CPLEX 12.6.3: QP Hessian is not positive semi-definite.
```

*Quadratic Objective*

# General Nonconvex *(cont'd)*

## *CPLEX setting to request nonconvex solve*

```
ampl: solve;

CPLEX 12.6.3.0: reqconvex 3
mipdisplay 2
mipinterval 1000

Reduced MIQP has 3 rows, 440 columns, and 80 nonzeros.
Reduced MIQP has 0 binaries, 40 generals, 0 SOSs, and 0 indicators.
Reduced MIQP objective Q matrix has 800 nonzeros.

.......

Total (root+branch&cut) =  758.41 sec.

CPLEX 12.6.3: optimal integer solution within mipgap or absmipgap;
   objective 69.30360303

8447893 MIP simplex iterations
637937 branch-and-bound nodes

absmipgap = 0.00675848, relmipgap = 9.75199e-05
```

***(n = 20)***

*Quadratic Objective*

# General Nonconvex *(cont'd)*

## *BARON (general nonlinear global solver)*

```
ampl: solve;

BARON 16.7.29 (2016.07.29)

This BARON run may utilize the following subsolver(s)
For LP/MIP: CLP/CBC
For NLP: IPOPT, FILTERSD

.......

Wall clock time:                50.69
Total CPU time used:            29.92

BARON 16.7.29 (2016.07.29): 708 iterations,
   optimal within tolerances.

Objective 69.30360303
```

# General Nonconvex *(cont'd)*

## *BARON using CPLEX*

```
ampl: solve;

BARON 16.7.29 (2016.07.29): lpsolver cplex

This BARON run may utilize the following subsolver(s)
For LP/MIP: ILOG CPLEX
For NLP: IPOPT, FILTERSD

.......

Wall clock time:                0.41
Total CPU time used:            0.38

BARON 16.7.29 (2016.07.29): 15 iterations,
   optimal within tolerances.

Objective 69.30360303
```

# General Nonconvex Strategies

*Nonconvex extension to quadratic MIP solver*

*Global nonlinear solver*

- Using built-in open source solvers
- Using commercial solvers
  - For linear MIP subproblems
  - For nonlinear subproblems

# Convex Quadratic Constraints

## *Elliptic form*

- $x^T Q x + q x \leq b$, where $Q$ is positive semi-definite
- Tested numerically

## *Conic form*

- $x_1^2 + \ldots + x_n^2 \leq x_{n+1}^2, \quad x_{n+1} \geq 0$
- $x_1^2 + \ldots + x_n^2 \leq x_{n+1}\, x_{n+2}, \quad x_{n+1} \geq 0,\ x_{n+2} \geq 0$
- Detected symbolically

# Conic Constraints

## *Traffic network 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;
```

# Conic Constraints *(cont'd)*

## *Add data and solve with Gurobi?*

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 7.0.0:
Gurobi can't handle nonquadratic nonlinear constraints
```

# Conic Constraints *(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;
```

# Conic Constraints *(cont'd)*

## *Add data and solve with Gurobi?*

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 7.0.0:
quadratic constraint is not positive definite
```

# Conic Constraints *(cont'd)*

## *Quadratically constrained 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;
```

# Conic Constraints *(cont'd)*

## *Add data and solve with Gurobi!*

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

ampl: option solver gurobi;
ampl: solve;

Gurobi 7.0.0: optimal solution; objective 76.26375
10 simplex iterations

ampl: display Flow;

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

# Conic Constraint Strategies

## *Solve with general nonlinear solver (not shown)*

- Local mixed-integer nonlinear solver (Knitro)
- Global solver (BARON)

## *Transform to conic problem*

- Many transformations known
- Could be done automatically
  - Jared Erickson and Robert Fourer,
    Detection and Transformation of Second-Order Cone Programming
    Problems in a General-Purpose Algebraic Modeling Language

# Elliptic Constraints

## *Portfolio model . . .*

```
set A;                       # asset categories
set T := {1973..1994};       # years

param R {T,A};               # returns on asset categories
param mu default 2;          # weight on variance

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

param Rtilde {i in T, j in A} := R[i,j] - mean[j];


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


maximize Reward: sum {j in A} mean[j] * Frac[j];;


subject to TotalOne: sum {j in A} Frac[j] = 1;
```

# Elliptic Constraints *(cont'd)*

## *Portfolio model restrictions on solution*

```
var RestrictVariance {T} binary;

subject to mostVar {i in T}:
   (sum {j in A} Rtilde[i,j]*Frac[j])^2
      <= maxVarT + (1-RestrictVariance[i]);

subject to RestrictDefn:
   sum {i in T} RestrictVariance[i] >= minRestrVar;

# -------

var Use {A} binary;

subject to UseDefn {j in A}:
   Frac[j] <= mostFrac * Use[j];

subject to LeastFrac {j in A}:
   Frac[j] >= leastFrac * Use[j];

subject to LeastUse:
   sum {j in A} Use[j] >= leastUse;
```

# Elliptic Constraints *(cont'd)*

## *Gurobi (0,0)*

```
Gurobi 7.0.0: premiqcpform=0
miqcpmethod=0

Presolved: 76 rows, 92 columns, 314 nonzeros
Presolved model has 19 second-order cone constraints

Gurobi 7.0.0: optimal solution; objective 1.126943182
1208 barrier iterations
236 branch-and-cut nodes

_solve_elapsed_time = 2.516
```

# Elliptic Constraints *(cont'd)*

## *Gurobi (0,1)*

```
Gurobi 7.0.0: premiqcpform=0
miqcpmethod=1
Presolved: 19 rows, 35 columns, 67 nonzeros
Variable types: 0 continuous, 35 integer (27 binary)
Gurobi 7.0.0: optimal solution; objective 1.126943182
134 simplex iterations
25 branch-and-cut nodes
_solve_elapsed_time = 0.093
```

# Elliptic Constraints *(cont'd)*

## *Gurobi (1,0)*

```
Gurobi 7.0.0: premiqcpform=1
miqcpmethod=0

Presolved: 109 rows, 88 columns, 336 nonzeros
Presolved model has 18 second-order cone constraints

Gurobi 7.0.0: optimal solution; objective 1.126943177
815 barrier iterations
139 branch-and-cut nodes

13 integer variables rounded to integers; maxerr = 2.30058e-06

_solve_elapsed_time = 1.172
```

*Convex Quadratic Constraints*

# Elliptic Constraints *(cont'd)*

## *Gurobi (1,1)*

```
Gurobi 7.0.0: premiqcpform=1
miqcpmethod=1

Presolved: 76 rows, 92 columns, 314 nonzeros
Variable types: 57 continuous, 35 integer (27 binary)

Gurobi 7.0.0: optimal solution; objective 1.126943182
801 simplex iterations
300 branch-and-cut nodes

_solve_elapsed_time = 0.156
```

# Elliptic Constraint Strategies

*Type of relaxation*

- Quadratic
- Linear (outer approximation)

*Type of quadratic constraint*

- Original elliptic constraint
- Transformation to conic constraint

*Convex Quadratic Constraints*

# Disaggregated Conics

## *Model with conic constraints . . .*

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

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

minimize Obj:
   sum {j in 1..n} (X[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 Conic {i in 1..n}:
   sum {j in 1..n} c[j]*X[j]^2 <= Y[i]^2;
```

*Convex Quadratic Constraints*

# Disaggregated Conics *(cont'd)*

## *Gurobi (0,1)*

```
Gurobi 7.0.0: premiqcpform=0
miqcpmethod=1

Presolved: 2 rows, 136 columns, 136 nonzeros
Variable types: 0 continuous, 136 integer (28 binary)

Gurobi 7.0.0: optimal solution; objective 103
24141 simplex iterations
8372 branch-and-cut nodes

_solve_elapsed_time = 15.922
```

***(n = 100)***

# Disaggregated Conics *(cont'd)*

## *Gurobi (1,1)*

```
Gurobi 7.0.0: premiqcpform=1
miqcpmethod=1

Presolved: 3702 rows, 3836 columns, 7536 nonzeros
Variable types: 3600 continuous, 236 integer (28 binary)

Gurobi 7.0.0: optimal solution; objective 103
23205 simplex iterations
4245 branch-and-cut nodes

_solve_elapsed_time = 7.235
```

*Convex Quadratic Constraints*

# Disaggregated Conics *(cont'd)*

## *Gurobi (2,1)*

```
Gurobi 7.0.0: premiqcpform=2
miqcpmethod=1

Solve qcp for cone disaggregation ...
Presolve removed 0 rows and 64 columns

Presolved: 3802 rows, 7436 columns, 11236 nonzeros
Variable types: 7200 continuous, 236 integer (28 binary)

Gurobi 7.0.0: optimal solution; objective 103
294892 simplex iterations
2657 branch-and-cut nodes

_solve_elapsed_time = 168.297
```

# Disaggregated Conic Strategies

## *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*

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

# Disaggregated Conic Strategies *(cont'd)*

## *One conic constraint with n terms*

- $x_1^2 + \ldots + x_n^2 \leq x_{n+1}^2,\; x_{n+1} \geq 0$

## *n conic constraints with one term each*

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

## *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