Strategies for "Not Linear" Optimization

Robert Fourer

AMPL Optimization
Northwestern University

4er@ampl.com — 4er@northwestern.edu

5th INFORMS Optimization Society Conference
Houston, TX — 6-8 March 2014

Session TA-01

Can I Linearize it?

Robert Fourer

AMPL Optimization
Northwestern University

4er@ampl.com — 4er@northwestern.edu

5th INFORMS Optimization Society Conference
Houston, TX — 6-8 March 2014

Session TA-01

What is Linearizing?

"Linear" solvers

- Linear and convex quadratic objectives & constraints
- Continuous or integer variables (or both)
- CPLEX, Gurobi, Xpress, MOSEK, SCIP, CBC, . . .

"Not Linear" problems

- Objectives & constraints in any other form
- Same continuous or integer variables

Goals

- Apply linear solvers to not linear problems
- Make this ***as easy as possible***

Example 1: Product of Variables

Can I linearize it?

- . . . + $c\,x\,y$ + . . . in my objective
  - where x, y are variables; c is a positive constant

It depends . . .

- What kinds of variables are x and y?
- Are you minimizing or maximizing?

Example 1

Case 1: Binary, Minimize

Original formulation

```
param c > 0;
var x binary;
var y binary;
minimize Obj: ... + c * x * y + ...
```

Linearization

```
var z;
minimize Obj: ... + c * z + ...
subject to z0Defn:  z >= 0;
subject to zxyDefn: z >= x + y - 1;
```

. . . z can be continuous
(minimization forces it to 0 or 1)

Example 1

Case 1 *(cont'd)*

Many other reformulations possible

- Best choice depends on problem *and* solver
- This one seems the best overall choice
  - see tests in Jared Erickson's dissertation:
    `JaredErickson2012@u.northwestern.edu`

Extends to product of two linear terms

- Multiply them out

Example 1

Case 1 *(cont'd)*

General 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;
```

Example 1

Case 1 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to solver
- Solver . . .
  - transforms products of binaries to linear formulations

Solved . . .

```
ampl: model xy1.mod;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.6.0.0: optimal integer solution; objective 232.6083992
395 MIP simplex iterations
0 branch-and-bound nodes
```

Example 1

Case 2: Binary, Maximize

Original formulation

```
param c > 0;
var x binary;
var y binary;
maximize Obj: ... + c * x * y + ...
```

Linearization

```
var z;
maximize Obj: ... + c * z + ...
subject to zxDefn: z <= x;
subject to zyDefn: z <= y;
```

. . . z can be continuous
(maximization forces it to 0 or 1)

Example 1

Case 2 *(cont'd)*

Constraints depend on objective sense

- Minimize: `z >= 0, z >= x + y - 1`
- Maximize: `z <= x, z <= y`

Would it help to include all?

- No, the continuous relaxation is not tightened
  - But may need all when extending this idea to xy in constraints

Example 1

Case 3: Binary & Continuous, Minimize

Original formulation

```
param c > 0;
var x binary;
var y >= L, <= U;
minimize Obj: ... + c * x * y + ...
```

Linearization

```
var z;
minimize Obj: ... + c * z + ...
subject to zLDefn: z >= L * x;
subject to zUDefn: z >= y - U * (1 - x);
```

Example 1

Case 3 *(cont'd)*

Extends in obvious ways

- Maximization form is symmetric
- y may be integer rather than continuous
  - reduces to binary case with $[L,U] = [0,1]$

Extends to product of two linear terms

- Equate the y term to a new variable *(optional)*
- Multiply terms out

Example 1

Case 3 *(cont'd)*

General model . . .

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

var X {1..n} binary;
var Y {1..n} >= 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 {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;
```

Example 1

Case 3 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to Gurobi
- Gurobi 5.6 solver . . .
  - transforms products of variables to linear formulations

Solved by Gurobi 5.6

```
ampl: model xy3.mod;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.6.0: optimal solution; objective 177.090486
216 simplex iterations
9 branch-and-cut nodes
```

Example 1

Case 3 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to CPLEX
- CPLEX 12.5 solver . . .
  - checks quadratic function for convexity

Rejected by CPLEX 12.5

```
ampl: model xy3.mod;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.5.0.1: QP Hessian is not positive semi-definite.
```

Example 1

Case 3 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to CPLEX
- CPLEX 12.5.1 solver . . .
  - transforms products of variables to linear formulations

Solved by CPLEX 12.5.1

```
ampl: model xy3.mod;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.5.1.0: optimal integer solution; objective 177.090486
148 MIP simplex iterations
```

Example 1

Case 4: Continuous, Maximize

Original formulation

```
param c > 0;
var x >= Lx, <= Ux;
var y >= Ly, <= Uy;
maximize Obj: c * x * y;
```

Conic reformulation

```
maximize Obj: c * z;
subject to zDefn: z^2 <= x * y;
```

Example 1

Case 4 *(cont'd)*

Handled by "linear" solvers

- Quadratic conic constraint region is convex
  - Original objective was quasi-concave

Can't sum terms in objective

- Optimal solutions are preserved, *but*
- Objective value changes (to square root of actual)
  - Not a problem if maximizing $cx^{1/2}y^{1/2}$

Can't do this with minimize!

- But can minimize a convex quadratic $(2xy + x^2 + y^2)$
- But can minimize product of negative powers

. . . more with conics in example 3

Example 1

Case 4 *(cont'd)*

General model . . .

```
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] >= 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;
```

Example 1

Case 4 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to Gurobi
- Gurobi solver . . .
  - checks quadratic function for convexity

Rejected by Gurobi

```
ampl: model xy4a.mod;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.6.0: quadratic objective is not positive definite
```

Example 1

Case 4 *(cont'd)*

Model transformed "by hand" . . .

```
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: .......
```

Example 1

Case 4 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - detects quadratic ***constraint*** terms
  - sends quadratic coefficient list to Gurobi
- Gurobi solver . . .
  - detects conic constraint structure

Solved by Gurobi barrier

```
ampl: model xy4b.mod;
ampl: option solver gurobi;
ampl: solve;
Gurobi 5.6.0: optimal solution; objective 29.78950231
11 barrier iterations
ampl: display ZX*ZY;
ZX*ZY = 887.414   # equals Obj^2
```

Example 1

Case 4 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to CPLEX

Solved by CPLEX branch-and-bound

```
ampl: model xy4a.mod;
ampl: option solver cplex, cplex_options 'reqconvex 3';
ampl: solve;
CPLEX 12.6.0.0: optimal integer solution; objective 887.4144789
24 MIP simplex iterations
5 branch-and-bound nodes
ampl: option cplex_options 'reqconvex 3 minimize';
ampl: solve;
CPLEX 12.6.0.0: optimal integer solution; objective 88.62097665
67 MIP simplex iterations
15 branch-and-bound nodes
```

Example 1

Key Questions

Can it be transformed?

- Yes or no?
- Transformed to what?

. . . very sensitive to mathematical form

Who will make the transformation?

- The human modeler?
- The modeling system?
- The solver interface?
- The solver?

. . . often some combination of these

Example 2: Zero or Range

Can I linearize it?

- If ***x*** isn't zero then I want it to be at least L
  - where $x \geq 0$ is a variable and $L > 0$ is a constant

Yes, and . . .

- You can also express it as a discontinuous domain
- You can also express it as a logical condition

Example 2

Scheduling

Minimize number of workers needed

- How many workers are assigned to each schedule?
- If a schedule is used at all,
  at least *L* workers must be assigned to it

Data: shifts in each schedule; least assignment L

```
set SHIFTS;

param Nsched;
set SCHEDS = 1..Nsched;

set SHIFT_LIST {SCHEDS} within SHIFTS;

param rate {SCHEDS} >= 0;
param required {SHIFTS} >= 0;

param least_assign >= 0;
```

Example 2

Case 1: Traditional MIP

Zero-one variables and inequalities

```
var Work {SCHEDS} >= 0 integer;
var Use  {SCHEDS} >= 0 binary;

minimize Total_Cost:
   sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
   sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use1 {j in SCHEDS}:
   least_assign * Use[j] <= Work[j];

subject to Least_Use2 {j in SCHEDS}:
   Work[j] <= (max {i in SHIFT_LIST[j]} required[i]) * Use[j];
```

Example 2

Case 1 *(cont'd)*

Solved by CPLEX

```
ampl: model sched1.mod;
ampl: data sched.dat;

ampl: option solver cplex;
ampl: let least_assign := 17;

ampl: solve;

Reduced MIP has 269 rows, 252 columns, and 1134 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 0 indicators.

Total (root+branch&cut) = 563.38 sec. (138138.56 ticks)

CPLEX 12.6.0.0: optimal integer solution; objective 267
24903192 MIP simplex iterations
3816760 branch-and-bound nodes
```

Example 2

Case 2: Discontinuous Domain

Union of a point and an interval

```
var Work {j in SCHEDS} integer in {0} union
   interval[least_assign, max {i in SHIFT_LIST[j]} required[i]];

minimize Total_Cost:
   sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
   sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];
```

Example 2

Case 2 *(cont'd)*

Transformed automatically

- AMPL processor . . .
  - adds auxiliary zero-one variables
  - generates appropriate constraints

Solved by CPLEX as usual

```
Reduced MIP has 269 rows, 252 columns, and 1134 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 0 indicators.

Total (root+branch&cut) = 342.49 sec. (85757.81 ticks)

CPLEX 12.6.0.0: optimal integer solution; objective 267
15087185 MIP simplex iterations
2306392 branch-and-bound nodes
```

Example 2

Case 2 *(cont'd)*

Same formulation as case 1

```
ampl: solexpand;

. . . . . .

subject to (Work[1]+IUlb):
Work[1] - 17*(Work[1]+b) >= 0;

subject to (Work[1]+IUub):
-Work[1] + 100*(Work[1]+b) >= 0;

subject to (Work[2]+IUlb):
Work[2] - 17*(Work[2]+b) >= 0;

subject to (Work[2]+IUub):
-Work[2] + 100*(Work[2]+b) >= 0;

. . . . . .
```

Example 2

Case 3: Implication

CPLEX indicator constraint

```
var Work {SCHEDS} >= 0 integer;
var Use  {SCHEDS} >= 0 binary;

minimize Total_Cost:
   sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
   sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use {j in SCHEDS}:
   Use[j] = 1 ==> Work[j] >= least_assign else Work[j] = 0;
```

Example 2

Case 3 *(cont'd)*

Logic passed to solver

- AMPL writes "logical" constraints as expression trees
- AMPL-CPLEX driver "walks" the trees
  - detects indicator forms
  - converts to CPLEX library calls

Solved by CPLEX with extensions

```
Reduced MIP has 143 rows, 252 columns, and 882 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 126 indicators.

Total (root+branch&cut) = 5936.45 sec. (1533625.65 ticks)

CPLEX 12.6.0.0: optimal integer solution; objective 267
250228203 MIP simplex iterations
29437722 branch-and-bound nodes
```

Example 2

Case 4: Disjunction

Logical constraint using "or" operator

```
var Work {j in SCHEDS} >= 0 integer;

minimize Total_Cost:
   sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
   sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use {j in SCHEDS}:
   Work[j] = 0 or Work[j] >= least_assign;
```

Example 2

Case 4 *(cont'd)*

Logic passed to solver

- AMPL writes "logical" constraints as expression trees
- AMPL-CPLEX driver "walks" the trees
  - looks for indicator forms

Rejected by CPLEX

```
ampl: option solver cplex;
ampl: solve;
CPLEX 12.5.0.1: logical constraint not indicator constraint.
```

Example 2

Case 4 *(cont'd)*

Logic passed to solver

- AMPL writes "logical" constraints as expression trees
- AMPL-CPLEX driver "walks" the trees
  - passes constraints as written to C++ "Concert" interface

Accepted by CPLEX after internal transformation

```
ampl: option solver ilogcp;
ampl: option ilogcp_options 'optimizer cplex mipdisplay 2';
ampl: solve;
Reduced MIP has 269 rows, 252 columns, and 1134 nonzeros.
Reduced MIP has 126 binaries, 126 generals, and 252 indicators.
<BREAK> (ilogcp)
Total (root+branch&cut) = 95272.30 sec. (23592380.69 ticks)
CPLEX 12.6.0.0: aborted, integer solution exists; objective 267
2.89e+009 MIP simplex iterations
351291725 branch-and-bound nodes
```

Example 3: Ratio

Can I linearize it?

- Minimize $\sum_{(i,j)\in A} t_{ij} x_{ij} \,/\, T$
- Where $t_{ij} = b_{ij} + s_{ij} x_{ij} \,/\, (1 - x_{ij}/c_{ij})$
  - where $0 \le x_{ij} < c_{ij}$ is a variable

Yes, but . . .

- The transformation isn't so obvious
- You will need conic quadratic constraints

Example 3

Traffic Network

Minimize average travel time for given throughput

- How much traffic is routed on each link?
- Travel time increases nonlinearly with traffic
- Flow is conserved at nodes

Example 3

Traffic Network *(cont'd)*

Model: flows and associated travel times

```
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 3

Traffic Network *(cont'd)*

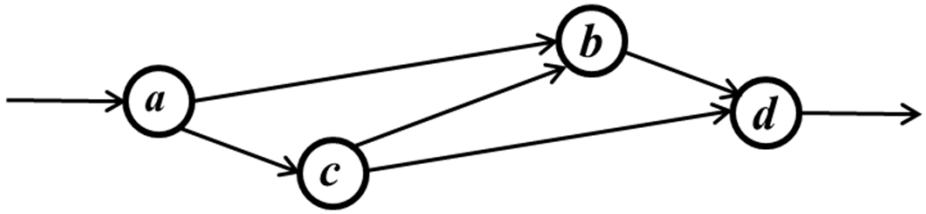
Data: Braess network

```
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 ;
```



Example 3

Case 1: General Nonlinear Solver

Solved "locally" by KNITRO

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

ampl: option solver knitro;
ampl: solve;

KNITRO 9.0.0: Locally optimal solution.
objective 61.04695019; feasibility error 1.24e-13
13 iterations; 20 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
;
```

Example 3

Case 1 *(cont'd)*

Same with integer-valued variables

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

```
ampl: solve;

KNITRO 9.0.0: Locally optimal solution.
objective 76.26375; integrality gap 1.56e-13
1 nodes; 2 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
;
```

Example 3

Case 2: "Linear" Solver

Rejected by CPLEX

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

ampl: option solver cplex;
ampl: solve;

CPLEX 12.6.0.0:
Constraint _scon[1] is not convex quadratic
since it is an equality constraint.
```

Example 3

Case 2 *(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;
```

Example 3

Case 2 *(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;
```

Example 3

Case 2 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to CPLEX
- CPLEX solver . . .
  - applies numerical test for elliptic quadratic constraints

Rejected by CPLEX as nonconvex

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

ampl: option solver cplex;
ampl: solve;

CPLEX 12.6.0.0:
QP Hessian is not positive semi-definite.
```

Example 3

Case 2 *(cont'd)*

Quadratically constrained re-reformulation

```
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;
```

Example 3

Case 2 *(cont'd)*

Transformed automatically

- AMPL interface . . .
  - multiplies out the linear objective terms
  - sends quadratic coefficient list to CPLEX
- CPLEX solver . . .
  - applies ***symbolic*** test for ***conic*** quadratic constraints

Solved (globally) by CPLEX

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

ampl: option solver cplex;
ampl: solve;

CPLEX 12.6.0.0: primal optimal; objective 61.04693968
15 barrier iterations
```

Example 3

Case 2 *(cont'd)*

Same with integer-valued variables

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

```
ampl: solve;

CPLEX 12.6.0.0: optimal integer solution within mipgap or absmipgap;
   objective 76.26375017
19 MIP simplex 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
;
```

Example 3

Case 3

Works just as well with Gurobi

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

```
ampl: solve;

Gurobi 5.6.0: optimal solution; objective 76.26374998
32 simplex iterations

ampl: display Flow;

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

Example 3

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

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

Rotated cone constraints

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

Sum-of-squares objectives

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

SOCP-solvable

SOC-Representable

Definition

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

Minimization property

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

Combination properties

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

. . . requires a recursive detection algorithm!

SOCP-solvable

SOC-Representable (1)

Vector norm

- $\|\mathbf{a} \cdot (\mathbf{Fx} + \mathbf{g})\| = \sqrt{\sum_{i=1}^{n} a_i^2 (\mathbf{f}_i\mathbf{x} + g_i)^2} \leq a_{n+1}(\mathbf{f}_{n+1}\mathbf{x} + g_{n+1})$
  - 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: $x_5^2 \le v_1v_2,\ v_1^2 \le x_1x_2,\ v_2^2 \le x_3x_4$
  - 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 $\mathbf{f}_i\mathbf{x}+g_i$ to have proper sign

SOCP-solvable

SOC-Representable (3)

p-norm

- $(\sum_{i=1}^{n}|\mathbf{f}_i\mathbf{x}+g_i|^p)^{1/p} \le \mathbf{f}_{n+1}\mathbf{x}+g_{n+1},\ p\in\mathbb{Q}^+,\ p\ge 1$
  - $(|x_1|^5+|x_2|^5)^{1/5}\le x_3$ 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 $\alpha_i\equiv 2$

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)

SOCP-solvable

Challenges

Applying "linear" solvers to these forms

- Recursive *detection* tree-walk
- Recursive *transformation* tree-walk
- Heuristic nonnegativity check for linear expressions

Assessing usefulness . . .

- Results from Jared Erickson's dissertation:
  `JaredErickson2012@u.northwestern.edu`

SOCP-solvable

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

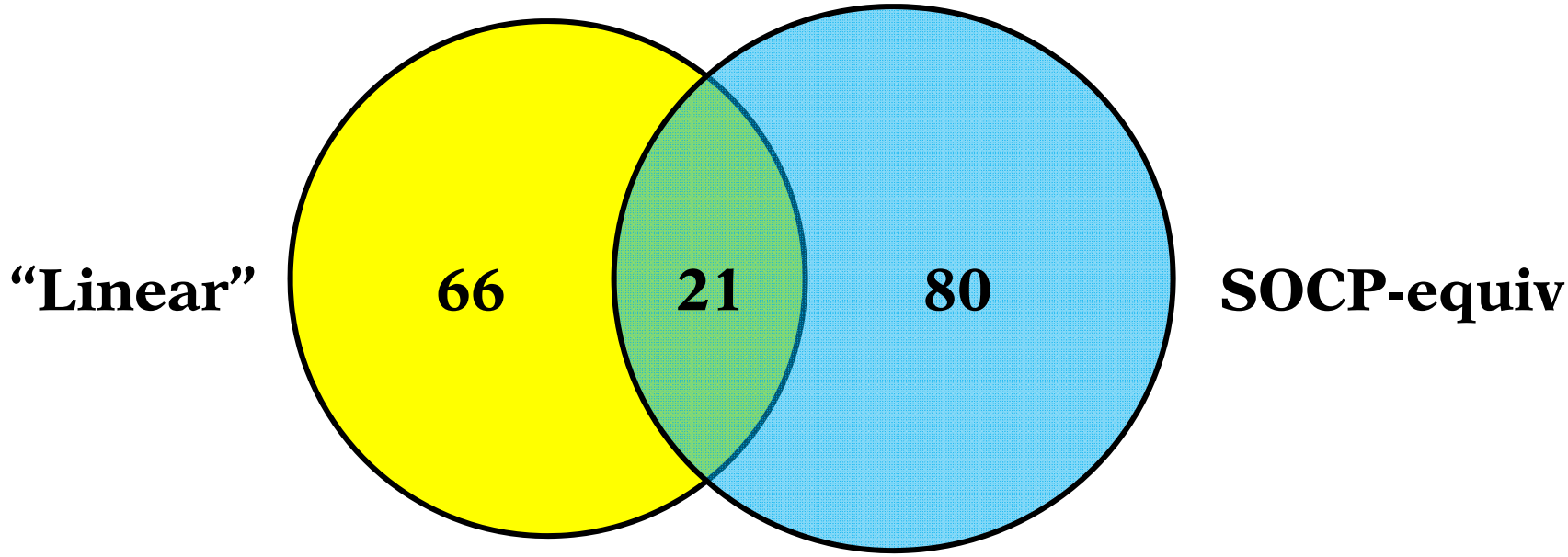
- `hs064` has $4/x_1 + 32/x_2 + 120/x_3 \leq 1$
- `hs036` minimizes $-x_1 x_2 x_3$
- `hs073` has $1.645\sqrt{0.28x_1^2 + 0.19x_2^2 + 20.5x_3^2 + 0.62x_4^2} \leq \ldots$
- `polak4` is a max of sums of squares
- `hs049` minimizes $(x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6$
- `emfl_nonconvex`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

SOCP-solvable

Survey of Test Problems (2)

Counted number of test problems . . .

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



Who Should Linearize It?

The AMPL user

The AMPL processor

The AMPL-solver interface

The solver

The AMPL User

Advantages

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

Disadvantages

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

The AMPL Processor

Advantages

- Makes the same linearization 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 AMPL-Solver Interface

Advantages

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

Disadvantages

- Creates an extra layer of complication

The Solver

Advantages

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

Disadvantages

- May not incorporate best practices
- Is complicated enough already