MIXED-INTEGER CONVEX OPTIMIZATION

Miles Lubin
with Emre Yamangil, Russell Bent, Juan Pablo Vielma, Chris Coey

September 16, 2016

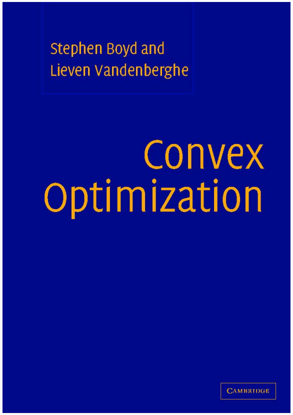
MIT & Los Alamos National Laboratory

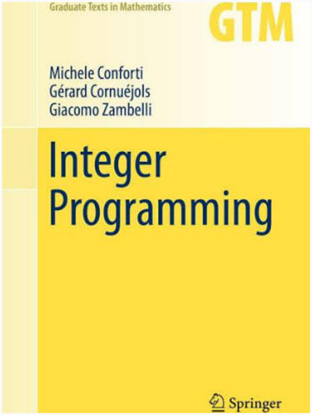
Nonlinear (nonconvex) optimization,

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & g_j(x) \leq 0, \quad \forall j \end{aligned}$$

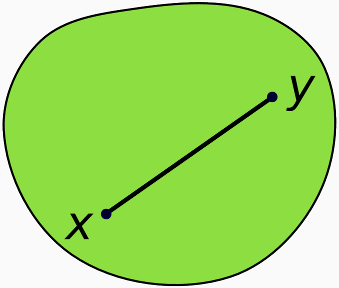
- Example: AC power flow, PDEs
- Global solution is **hard**
- Be happy with local search, or
- Approximate as another optimization problem we know how to solve
- Search for global solutions via convex relaxations

Which optimization problems can we solve to global optimality?





Convex optimization



- All local optima are global
- Typically efficient, polynomial-time algorithms

Image source: `https://en.wikipedia.org/wiki/Convex_set#/media/File:Convex_polygon_illustration1.svg`

Mixed-integer linear programming (MILP),

$$\begin{aligned}\min_{x} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0, \\ & x_i \in \mathbb{Z}, \quad \forall i \in I\end{aligned}$$

- Despite NP-Hardness, many problems of practical interest can be solved to optimality or near optimality
- Algorithms are based on LP relaxations, branch & bound, cuts, preprocessing, heuristics, ...
- 50+ years of commercial investment in developing these techniques

WHO USES MILP?

http://www.gurobi.com/company/example-customers

EXAMPLE: UNIT COMMITMENT

Minimize unit-production costs + fixed operating costs

subject to:

- Total generation = total demand, hourly over 24h
- Generation and transmission limits
  - If a generator is on, it produces within some interval $[l, u]$, otherwise its production is zero
- Linear approximation of nonlinear powerflow laws

1% reduction in generation gives billions of dollars per year cost reduction. Worth spending time to improve optimality.

Mixed-integer convex programming (MICP),

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & g_j(x) \le 0, \quad \forall j \\ & x_i \in \mathbb{Z}, \quad \forall i \in I \end{aligned}$$

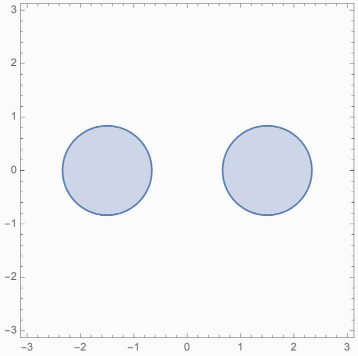
Objective function linear (WLOG) and constraints g_j **convex**

- **Maybe MICP can be a more useful approximation of nonconvex problems than convex or MILP alone**

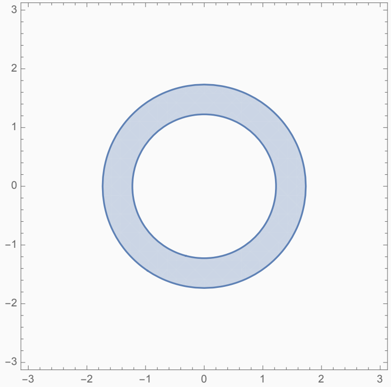
WHAT CAN BE MODELED USING MICP?

Disjunctions/unions of compact convex sets (Stubbs and Mehrotra, 1999)

$$\{x : f_1(x) \leq 0 \text{ or } f_2(x) \leq 0\}$$



The following set is not MICP representable



State of the art for solving MICP

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & g_j(x) \leq 0, \forall j \\ & x_i \in \mathbb{Z}, \forall i \in I \end{aligned}$$

- g_j convex, **smooth**
- Bonmin, KNITRO, α-ECP, DICOPT, FilMINT, MINLP_BB, SBB, ...

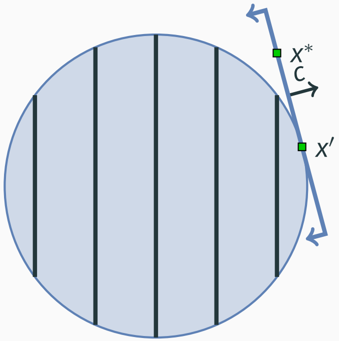
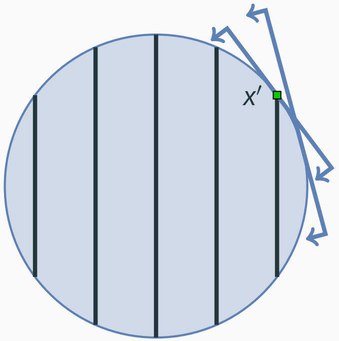
OUTER APPROXIMATION ALGORITHM

Idea: Reformulate as another optimization problem we know how to solve (MILP)

Given points $x_1^*, \ldots, x_R^*$, solve mixed-integer linear relaxation:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & g_j(x_r^*) + \nabla g_j(x_r^*)^T (x - x_r^*) \leq 0, \quad \forall j, r = 1, \ldots, R \\ & x_i \in \mathbb{Z}, \quad \forall i \in I \end{aligned}$$

Then solve convex problem with integer values fixed to get a new feasible solution, and repeat with $R = R + 1$.


x^*
c
x'

x'

NOT GOOD ENOUGH...

> *[...] solution algorithms for [MICP] have benefit from the technological progress made in solving MILP and NLP. However, in the realm of [MICP], the progress has been far more modest, and the dimension of solvable [MICP] by current solvers is small when compared to MILPs and NLPs.*

– Bonami, Kilinç, and Linderoth (2012)

WHAT CAN GO WRONG?

Classical approaches form **polyhedral outer approximations** by a finite collection of "gradient linearizations". But a good polyhedral outer approximation might need **too many linear constraints**.

Recall:

$$\mathcal{B}_1 := \{x : ||x||_1 \leq 1\} = \left\{x : \sum_{i=1}^{n} s_i x_i \leq 1, s \in \{-1, +1\}^n\right\}$$

$$\mathcal{B}_1 := \{x : ||x||_1 \leq 1\} = \left\{x : \sum_{i=1}^{n} s_i x_i \leq 1, s \in \{-1,+1\}^n\right\}$$

Standard trick in linear programming is to **introduce extra variables:**

$$\mathcal{B}_1 = \left\{x \in \mathbb{R}^n : \exists y \in \mathbb{R}^n \text{ s.t. } x \leq y, x \geq -y, \sum\nolimits_i y_i \leq 1\right\}$$

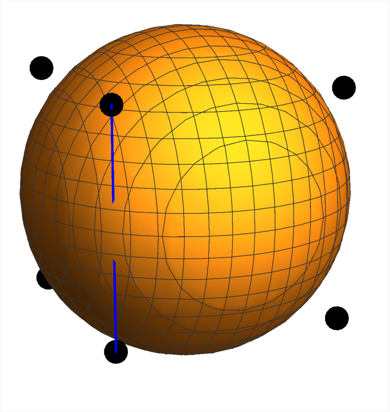
The new formulation has $2n$ variables and $2n+1$ constraints versus n variables and 2^n constraints. Not bad.

We call this an **extended formulation**.

THE SAME HAPPENS WITH SMOOTH CONSTRAINTS

Hijazi et al. (2014):

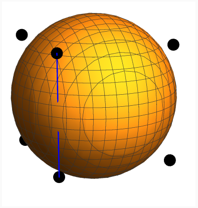
$$B^n := \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} \left(x_j - \frac{1}{2} \right)^2 \leq \frac{n-1}{4} \right\}$$



WHAT CAN GO WRONG?

Hijazi et al. (2014):

$$B^n := \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} \left(x_j - \frac{1}{2} \right)^2 \leq \frac{n-1}{4} \right\}$$



- B^n is empty
- No outer approximating hyperplane can exclude more than one integer lattice point
- So any polyhedral outer approximation which contains no integer vertices needs **at least** 2^n **hyperplanes**

HOW TO FIX IT?

$$B^n := \left\{ x \in \{0,1\}^n : \sum_{j=1}^n \left(x_j - \frac{1}{2} \right)^2 \leq \frac{n-1}{4} \right\}$$

Consider the equivalent formulation in an extended set of variables:

$$\hat{B}^n := \left\{ x \in \{0,1\}^n, z \in \mathbb{R}^n : \sum_{j=1}^n z_j \leq \frac{n-1}{4}, \left(x_j - \frac{1}{2} \right)^2 \leq z_j \, \forall j \right\}$$

- $B^n = \operatorname{proj}_x \hat{B}^n$

HOW TO FIX IT?

$$B^n := \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} \left(x_j - \frac{1}{2} \right)^2 \leq \frac{n-1}{4} \right\}$$

Consider the equivalent formulation in an extended set of variables:

$$\hat{B}^n := \left\{ x \in \{0,1\}^n, z \in \mathbb{R}^n : \sum_{j=1}^{n} z_j \leq \frac{n-1}{4}, \left(x_j - \frac{1}{2} \right)^2 \leq z_j \, \forall j \right\}$$

- $B^n = \operatorname{proj}_x \hat{B}^n$
- If you form a polyhedral relaxation of each $\left(x_j - \frac{1}{2}\right)^2 \leq z_j$ individually, then $2n$ hyperplanes in $\mathbb{R}^{2n}$ is sufficient to exclude all integer points

HOW TO FIX IT?

$$B^n := \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} \left(x_j - \frac{1}{2} \right)^2 \le \frac{n-1}{4} \right\}$$

Consider the equivalent formulation in an extended set of variables:

$$\hat{B}^n := \left\{ x \in \{0,1\}^n, z \in \mathbb{R}^n : \sum_{j=1}^{n} z_j \le \frac{n-1}{4}, \left(x_j - \frac{1}{2} \right)^2 \le z_j \; \forall j \right\}$$

- $B^n = \operatorname{proj}_x \hat{B}^n$
- If you form a polyhedral relaxation of each $\left(x_j - \frac{1}{2}\right)^2 \le z_j$ individually, then $2n$ hyperplanes in $\mathbb{R}^{2n}$ is sufficient to exclude all integer points
- Outer approximation algorithm now converges in 2 iterations

HOW TO FIX IT?

$$B^n := \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} \left(x_j - \frac{1}{2}\right)^2 \leq \frac{n-1}{4} \right\}$$

Consider the equivalent formulation in an extended set of variables:

$$\hat{B}^n := \left\{ x \in \{0,1\}^n, z \in \mathbb{R}^n : \sum_{j=1}^{n} z_j \leq \frac{n-1}{4}, \left(x_j - \frac{1}{2}\right)^2 \leq z_j \, \forall j \right\}$$

- $B^n = \text{proj}_x \hat{B}^n$
- If you form a polyhedral relaxation of each $\left(x_j - \frac{1}{2}\right)^2 \leq z_j$ individually, then $2n$ hyperplanes in $\mathbb{R}^{2n}$ is sufficient to exclude all integer points
- Outer approximation algorithm now converges in 2 iterations
- Why? **Small polyhedron in higher dimension can have exponentially many facets in lower dimension**

“extended formulation for second-order cone”

$$SOC_n := \{(t, x) \in \mathbb{R}^n : ||x||_2 \leq t\}$$

Second order cone programming (SOCP) generalizes convex quadratic programming (QP).

"EXTENDED FORMULATION FOR SECOND-ORDER CONE"

$$SOC_n := \{(t, x) \in \mathbb{R}^n : ||x||_2 \leq t\}$$

Second order cone programming (SOCP) generalizes convex quadratic programming (QP).

Vielma, Dunning, Huchette, Lubin (2015):

- For MISOCP: Rewrite
$$\sum_i x_i^2 \leq t^2$$
as
$$\sum_i z_i \leq t, \text{ where } z_i \geq x_i^2/t$$
- Implemented by CPLEX and Gurobi within months of publication

WE KNOW WHAT TO DO...

Hijazi et al. (2014) propose to reformulate constraints of the form

$$\sum_{i=1}^{n} g_i(x_i) \leq 0$$

to

$$\sum_{i=1}^{n} z_i \leq 0 \text{ and } g_i(x_i) \leq z_i \; i = 1, \ldots, n,$$

where g_i univariate, smooth, convex.

- Very impressive computational results
- Paper submitted in 2011, but there's still no solver which automates this transformation

Why has nobody implemented this?

Why has nobody implemented this?

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & g_j(x) \leq 0, \quad \forall j \\ & x_i \in \mathbb{Z}, \quad \forall i \in I \end{aligned}$$

- MICP solvers view f and g_j through “black-box” oracles for evaluations of values and derivatives
- Summation structure not accessible through the “black box”

So what if we open the black box and access the algebraic structure? Is it sufficient?

So what if we open the black box and access the algebraic structure? Is it sufficient?

If $h(x) = f(x) + g(x)$ is convex, not necessary for f and g to be convex. (E.g., $f(x) = x_1^2 - x_2^2$ and $g(x) = 2x_2^2$.)

So what if we open the black box and access the algebraic structure? Is it sufficient?

If $h(x) = f(x) + g(x)$ is convex, not necessary for f and g to be convex. (E.g., $f(x) = x_1^2 - x_2^2$ and $g(x) = 2x_2^2$.)

So if we have access to the algebraic representation, in principle need to solve a subproblem of convexity detection before constructing extended formulation. **Convexity detection is a hard problem.**

Disciplined convex programming (DCP) as a solution

- Algebraic modeling concept proposed by Grant, Boyd, and Ye. Implemented in CVX, CVXPY, and Convex.jl.
- Users forced to prove convexity of their functions by following basic composition rules.

Disciplined convex programming (DCP) as a solution

- $\sqrt{x^2 + y^2}$ not DCP valid, use norm($\begin{bmatrix} x_1 & x_2 \end{bmatrix}$)
- $\log(\exp(x_1) + \exp(x_2))$ not DCP valid, use logsumexp($\begin{bmatrix} x_1 & x_2 \end{bmatrix}$)
- DCP library of atoms, plus valid compositions
    - Sum of convex functions is convex
    - $f(g(x))$ is convex when f is convex and nondecreasing and g is convex

DCP composition rules correspond exactly to extended formulations!!

- Sum of convex functions is convex
    - $f(x) + g(x) \leq 0 \longrightarrow f(x) \leq t_1, g(x) \leq t_2, t_1 + t_2 \leq 0$
- $f(g(x))$ is convex when f is convex and nondecreasing and g is convex
    - $f(g(x)) \leq 0 \longrightarrow g(x) \leq t, f(t) \leq 0$
    - Tawarmalani and Sahinidis (2005) show that polyhedral outer approximations that exploit composition structure have extra strength.

DCP composition rules correspond exactly to extended formulations!!

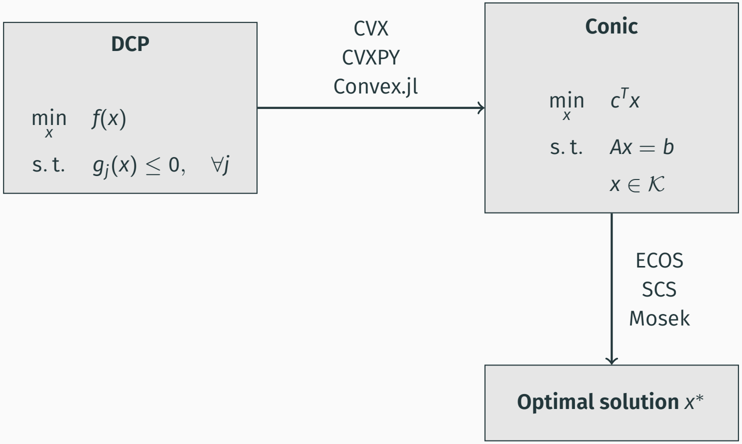
Not a coincidence. To enforce $f(x) + g(x) \leq 0$, it is sufficient to know how to optimize over the intersection of the *epigraphs* $\{(t, x) : f(x) \leq t\}$, $\{(t, x) : g(x) \leq t\}$ and linear constraints.

DCP was designed to access state-of-the-art conic solvers, **does not use gradients**.

CONIC OPTIMIZATION

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \in \mathcal{K} \end{aligned}$$

- $\mathcal{K}$ is a cone if $x \in \mathcal{K}$ implies $\alpha x \in \mathcal{K} \, \forall \alpha \geq 0$
- Usually we consider $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_s$ where each $\mathcal{K}_j$ is one of a small number of cones like $\mathbb{R}^n_+$, SOC_n, the cone of positive semidefinite matrices...


DCP
$\min_x \quad f(x)$
$\text{s.t.} \quad g_j(x) \le 0, \quad \forall j$
CVX
CVXPY
Convex.jl
Conic
$\min_x \quad c^T x$
$\text{s.t.} \quad Ax = b$
$x \in \mathcal{K}$
ECOS
SCS
Mosek
Optimal solution x^*

Example:

$$\sum_i \exp(c_i^T x + b_i) \le t$$

is expressed in conic form as

$$\sum_i z_i \le t \text{ and } (c_i^T x + b_i, 1, z_i) \in EXP.$$

where

$$EXP = \mathrm{cl}\{(x, y, z) \in \mathbb{R}^3 : y \exp(x/y) \le z, y > 0\}$$

is the **exponential cone**.

Example:

$$\sum_i \exp(c_i^T x + b_i) \leq t$$

is expressed in conic form as

$$\sum_i z_i \leq t \text{ and } (c_i^T x + b_i, 1, z_i) \in EXP.$$

where

$$EXP = \mathrm{cl}\{(x, y, z) \in \mathbb{R}^3 : y \exp(x/y) \leq z, y > 0\}$$

is the **exponential cone**.

The translation to conic form already produces a formulation with additional variables ideal for polyhedral outer approximation.

Example:

$$\exp(x^2 + y^2) \leq t$$

is DCP compliant. CVX will generate equivalent formulation

$$\exp(z) \leq t \text{ where } x^2 + y^2 \leq z$$

which is then translated to EXP and second-order cones.

- Mixed-integer second order cone programming (MISOCP) has rapidly improving commercial support by Gurobi/CPLEX
    - Gurobi claims 2.8x improvement from version 6.0 to 6.5 (1 year)
- Otherwise (exponential cones, SDP), mainly research codes (`ecos_bb` by Han Wang, 2014; `SCIP-SDP` by Mars, 2012)
- For MISOCP, no established open-source codes

How general is MISOCP?

MINLPLIB2 benchmark library designed for derivative-based mixed-integer nonlinear solvers. We classified all of the convex instances by conic representability.

Of the 333 convex instances, **65%** are MISOCP representable.

- `tls6` instance previously unsolved. We translated it to Convex.jl and it was solved by Gurobi!

What about the remaining 35% of problems?

Of remaining 116 instances, in 107 the only nonlinear expressions involve exp and log.

What about the remaining 35% of problems?

Of remaining 116 instances, in 107 the only nonlinear expressions involve exp and log.

Recall the exponential cone:

$$EXP = \mathrm{cl}\{(x, y, z) \in \mathbb{R}^3 : y \exp(x/y) \leq z, y > 0\}$$

- Closed, convex, nonsymmetric cone. Theoretically tractable and supported by ECOS (Akle, 2015)

What about the remaining 9 of 333 instances?

- Seven representable by combination of *EXP* and second-order cones (*SOC*)
- Two representable by the power cone

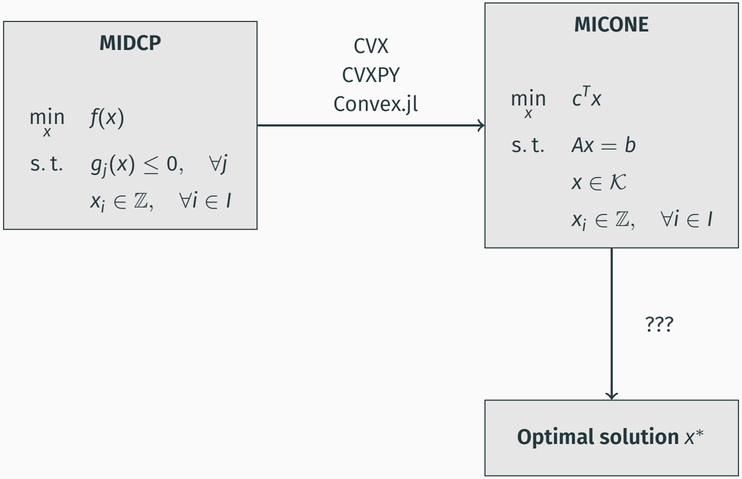
$$POW_\alpha = \{(x, y, z) \in \mathbb{R}^3 : |z| \leq x^a y^{1-a}, x \geq 0, y \geq 0\}$$

Folklore: **Almost all convex optimization problems of practical interest can be represented as conic programming problems using second-order, positive semidefinite, exponential, and power cones.**

So cones are general and provide a good extended formulation of the original convex problem.

How do we apply polyhedral outer approximation to mixed-integer conic problems?

- We'll state the first finite-time outer approximation algorithm for general mixed-integer conic optimization


MIDCP
$\min_x \quad f(x)$
$\text{s.t.} \quad g_j(x) \le 0, \quad \forall j$
$x_i \in \mathbb{Z}, \quad \forall i \in I$
CVX
CVXPY
Convex.jl
MICONE
$\min_x \quad c^T x$
$\text{s.t.} \quad Ax = b$
$x \in \mathcal{K}$
$x_i \in \mathbb{Z}, \quad \forall i \in I$
???
Optimal solution x^*

MIXED-INTEGER CONIC PROGRAMMING

$$\begin{aligned} \min_{x,z} \quad & c^T z \\ \text{s.t.} \quad & A_x x + A_z z = b \\ & L \leq x \leq U, \quad x \in \mathbb{Z}^n, \quad z \in \mathcal{K}, \end{aligned} \qquad \text{(MICONE)}$$

- $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2 \times \cdots \times \mathcal{K}_r$, where each $\mathcal{K}_i$ is a standard cone: nonnegative orthant, second-order cone, exponential cone, power cone, or even positive semidefinite cone
- WLOG, integer variables do not belong to cones, have zero objective coefficients

Need to outer approximate cone $\mathcal{K}$ with finite number of linear constraints...

Need to outer approximate cone $\mathcal{K}$ with finite number of linear constraints...

Given a closed, convex cone $\mathcal{K}$, the **dual cone** $\mathcal{K}^*$ is the set such that $z \in \mathcal{K}$ iff $z^T\beta \geq 0 \ \forall \beta \in \mathcal{K}^*$.

- Nonnegative orthant, second-order and positive semidefinite cone are *self dual*. Exponential and power cones are not
- Any finite subset of $\mathcal{K}^*$ yields a polyhedral outer approximation of $\mathcal{K}$!

$$
\begin{aligned}
\min_{x,z} \quad & c^T z \\
\text{s.t.} \quad & A_x x + A_z z = b \qquad \qquad \text{(MIOA(T))} \\
& L \le x \le U, \quad x \in \mathbb{Z}^n, \\
& \beta^T z \ge 0 \quad \forall \beta \in T,
\end{aligned}
$$

- If $T = \mathcal{K}^*$, then equivalent to MICONE
- If $T \subseteq \mathcal{K}^*$ and $|T| < \infty$, then *polyhedral outer approximation* $\rightarrow$ MILP relaxation of MICONE
- **Claim:** $\exists T \subseteq \mathcal{K}^*, |T| < \infty$ such that MIOA(T) is equivalent to *MICONE*

Consider the subproblem with integer values $x = \hat{x}$ fixed:

$$\begin{aligned} v_{\hat{x}} = \min_{z} \quad & c^T z \\ \text{s.t.} \quad & A_z z = b - A_x \hat{x}, \\ & z \in \mathcal{K}. \end{aligned} \tag{$CP(\hat{x})$}$$

The dual of $CP(\hat{x})$ is

$$\begin{aligned} \max_{\beta,\lambda} \quad & \lambda^T (b - A_x \hat{x}) \\ \text{s.t.} \quad & \beta = c - A_z^T \lambda \\ & \beta \in \mathcal{K}^*. \end{aligned}$$

THE CONIC OUTER APPROXIMATION ALGORITHM

- Start with $T = \emptyset$ (or small)
- Until the optimal objective of MIOA(T) is $\geq$ the optimal objective of the best feasible solution:
  - Let $\hat{x}$ be the optimal integer solution of the MILP problem MIOA(T)
  - Solve conic problem $CP(\hat{x})$, add dual solution $\beta_{\hat{x}}$ to T
  - If $CP(\hat{x})$ was feasible, record new feasible solution to MICONE

THE CONIC OUTER APPROXIMATION ALGORITHM

- Start with $T = \emptyset$ (or small)
- Until the optimal objective of MIOA(T) is $\geq$ the optimal objective of the best feasible solution:
    - Let $\hat{x}$ be the optimal integer solution of the MILP problem MIOA(T)
    - Solve conic problem $CP(\hat{x})$, add dual solution $\beta_{\hat{x}}$ to T
    - If $CP(\hat{x})$ was feasible, record new feasible solution to MICONE

Finite termination assuming strong duality holds at subproblems

THE CONIC OUTER APPROXIMATION ALGORITHM

- Easy to implement, requires black box MILP and conic solvers
- Very often, number of iterations is < 10, sometimes 10-30. Worst observed is 171.
- Each MILP subproblem could be NP-Hard, but takes advantage of fast solvers

RECAP

- Polyhedral approximations can be made *much* better by introducing extra variables

RECAP

- Polyhedral approximations can be made *much* better by introducing extra variables
- Summation structure and compositions of convex functions should be broken up

RECAP

- Polyhedral approximations can be made *much* better by introducing extra variables
- Summation structure and compositions of convex functions should be broken up
- Nobody automated this before! (Derivative-based view prevents implementation)

RECAP

- Polyhedral approximations can be made *much* better by introducing extra variables
- Summation structure and compositions of convex functions should be broken up
- Nobody automated this before! (Derivative-based view prevents implementation)
- DCP to the rescue. Conic form is general and encodes the information we need as a solver

RECAP

- Polyhedral approximations can be made *much* better by introducing extra variables
- Summation structure and compositions of convex functions should be broken up
- Nobody automated this before! (Derivative-based view prevents implementation)
- DCP to the rescue. Conic form is general and encodes the information we need as a solver
- Proposed first outer approximation algorithm for MICONE, based on conic duality

Computational results

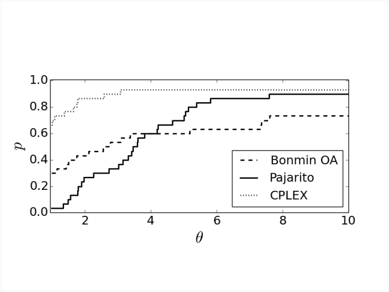
Pajarito (Polyhedral Approximation in Julia...)

New solver!

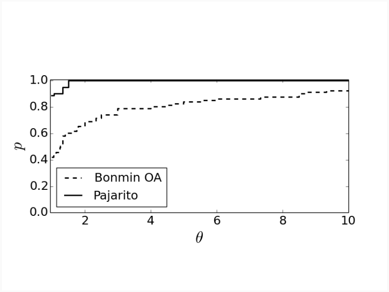
- 1500 lines of Julia
- Input is in conic form, accessible through Convex.jl DCP language and JuMP
- Works with any MILP and conic solvers supported in Julia
- `Pkg.add(”Pajarito”)`

Translated 194 convex instances from MINLPLIB2 from AMPL into Convex.jl (heroic parsing work by E. Yamangil).

Compare with Bonmin's outer approximation algorithm using CPLEX as the MILP solver. We use the same version of CPLEX plus KNITRO as NLP (conic) solver.

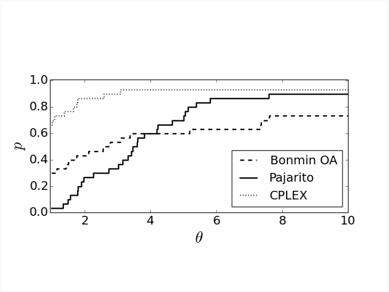


(a) Solution time

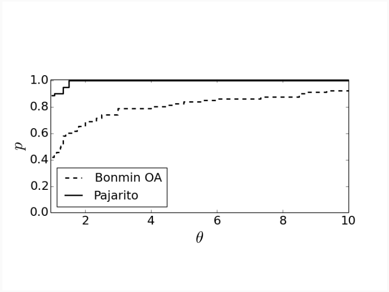


(b) Number of OA iterations

Figure: Comparison performance profiles for SOC representable instances (Bonmin >30 sec)



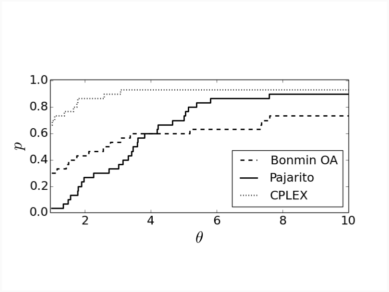
(a) Solution time



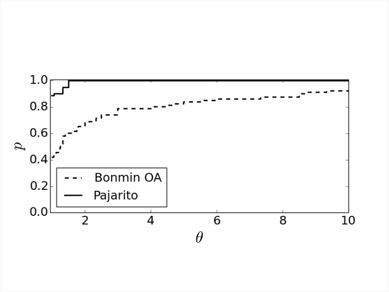
(b) Number of OA iterations

Figure: Comparison performance profiles for SOC representable instances (Bonmin >30 sec)

- Dominates on iteration count. Haven't optimized Pajarito so Bonmin is faster on the easier problems.



(a) Solution time



(b) Number of OA iterations

Figure: Comparison performance profiles for SOC representable instances (Bonmin >30 sec)

- Dominates on iteration count. Haven't optimized Pajarito so Bonmin is faster on the easier problems.
- CPLEX is usually the fastest on MISOCPs. Use that instead.

Almost none of the non-MISOCP instances in MINLPLIB2 are hard!

gams01

`gams01` unsolved instance which is EXP+SOC representable

- 145 variables. 1268 constraints (1158 are linear)
- Previous best bound: 1735.06. Best known solution: 21516.83
- Pajarito solved in 6 iterations ($<$ 10 hours). Optimal solution is 21380.20.

Claim: Pajarito is the fastest open-source MISOCP solver

80 MISOCP instances from CBLIB, 20 minute timeout

	Pajarito	Bonmin OA	Bonmin BB	Bonmin OA-D	CPLEX
Optimal	47	9	16	11	69
Timeout	33	49	45	47	11
Error	0	22	19	22	0

Pajarito using ECOS or ConicIP as SOCP solver. Bonmin uses Ipopt.
Both use CBC as MILP solver.

ONGOING AND FUTURE WORK

- Computational results with exponential cone solvers, branch & cut
- Extensions to semidefinite programming
- What happens when strong duality fails?

Pajarito Mountain, New Mexico

Thanks! Questions?

https://github.com/mlubin/Pajarito.jl

http://dx.doi.org/10.1007/978-3-319-33461-5_9

http://arxiv.org/abs/1607.03566

DOE CSGF