Polyhedral Approaches for Mixed-integer Convex Optimization

GERAD

Miles Lubin

February 8, 2019

Google, New York City (This work was conducted at MIT.)

Optimization

Given: Feasible region S and objective function $f : S \to \mathbb{R}$

Find: $\boldsymbol{x} \in S$ with lowest possible value of $f(\boldsymbol{x})$ (global solution)

Optimization

Given: Feasible region S and objective function $f : S \to \mathbb{R}$

Find: $\boldsymbol{x} \in S$ with lowest possible value of $f(\boldsymbol{x})$ (global solution)

How to solve?

- Local search (e.g., gradient descent), genetic algorithms, ...
- Approximate or reformulate (if lucky) as another optimization problem we know how to solve to a global solution

Which optimization problems can we solve to global optimality?

Which optimization problems can we solve to global optimality?

- Integer programming
- Convex optimization

In this talk, we consider **mixed-integer convex optimization**, or mixed-integer convex programming (MICP).

These are convex optimization problems with integrality constraints.

We look for global solutions or at least global bounds.

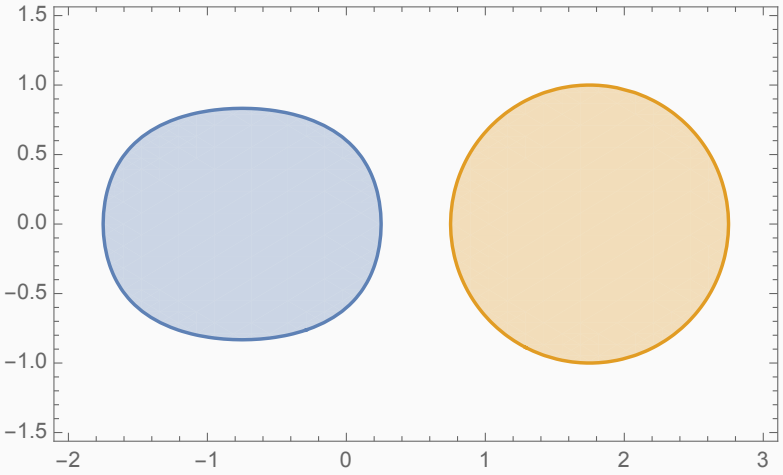
Themes

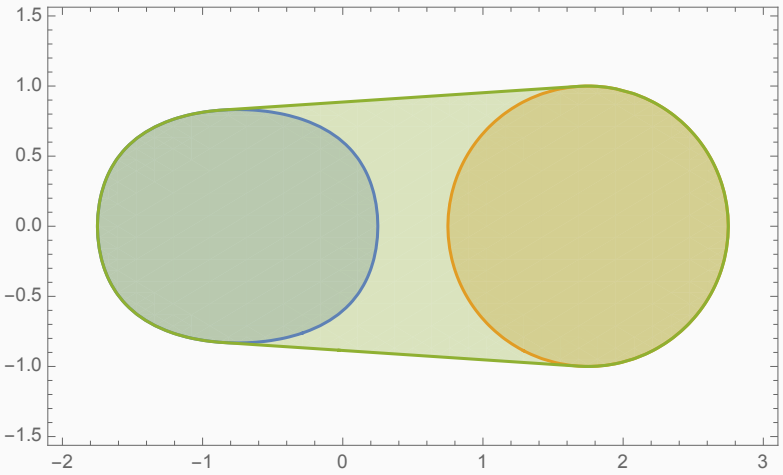
We apply extended formulations, conic duality, and branch and cut to get a state-of-the-art open-source MICP solver.

This work motivated developments that will appear in MOSEK 9 (presented at ISMP 2018).

What can you model with MICP?

It is possible to model a broad scope of problems using MILP. What *more* can you do if allowed to model with convex constraints?





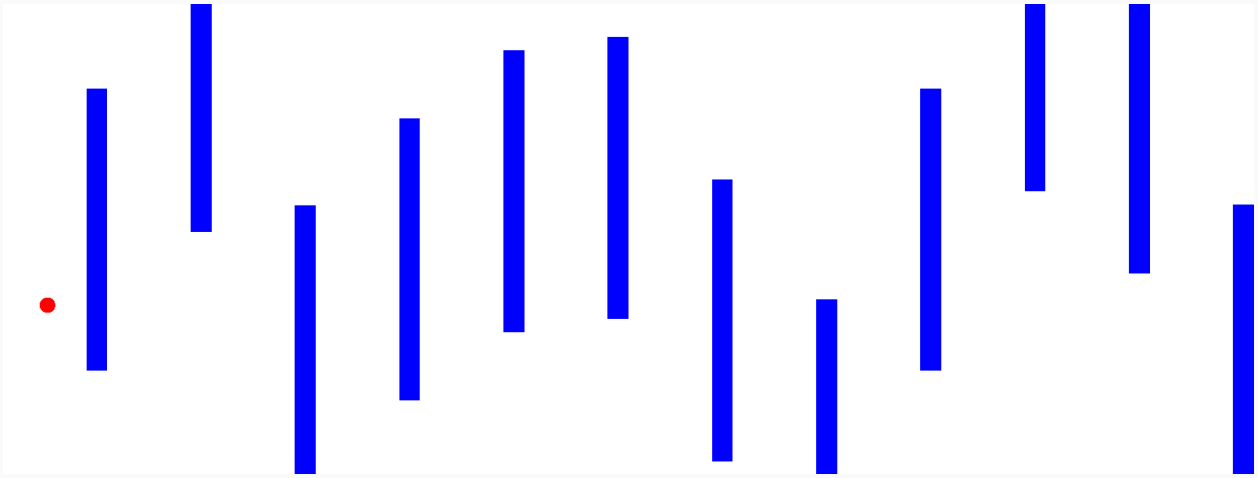
Ceria and Soares (1999) develop MICP formulations for unions of certain (e.g., bounded) convex sets.

See L., Zadik, and Vielma (IPCO 2017, arXiv 1706.05135) for much more on this.

A fun example

Mixed-integer sum-of-squares optimization

- Sum-of-squares constraints (which are convex) can be used to model polynomial constraints, including trajectories of a moving object
- Control decisions with discrete aspect (e.g., minimize number of thrusts) yield integer constraints



Credit: Joey Huchette (SIAM Conference on Optimization, 2017)

We define MICP general form as:

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^N} \quad & \boldsymbol{c}^T \boldsymbol{x} : \\ & \boldsymbol{x} \in \mathcal{X}, \\ & x_i \in \mathbb{Z}, \qquad \forall i \in [\![I]\!] = \{1, \ldots, I\}, \end{aligned}$$

where $\mathcal{X}$ is a closed, convex set, and the first I of N variables are constrained to take integer values.

We can always assume objective is linear, via an epigraph transformation.

When $\mathcal{X}$ is a polyhedron, this reduces to MILP.

Solution methods

- Branch & bound – recursively partition possible assignments to integer variables and use convex relaxations to prune the search tree (Gupta and Ravindran, 1985; Bonami et al., 2013)

Solution methods

- Branch & bound – recursively partition possible assignments to integer variables and use convex relaxations to prune the search tree (Gupta and Ravindran, 1985; Bonami et al., 2013)
- **Iterative outer approximation (OA)** – based on polyhedral relaxation of convex constraints, solve a sequence of MILP relaxations and convex subproblems (Duran and Grossmann, 1986; Leyffer, 1993; Bonami et al., 2008)

Solution methods

- Branch & bound – recursively partition possible assignments to integer variables and use convex relaxations to prune the search tree (Gupta and Ravindran, 1985; Bonami et al., 2013)
- **Iterative outer approximation (OA)** – based on polyhedral relaxation of convex constraints, solve a sequence of MILP relaxations and convex subproblems (Duran and Grossmann, 1986; Leyffer, 1993; Bonami et al., 2008)
- **LP-based branch & bound** – use polyhedral relaxations within a single branch & bound (B&B) search tree (Quesada and Grossmann, 1992; Bonami et al., 2008)

Developing polyhedral relaxations

Suppose:

$$\mathcal{X} = \left\{ \boldsymbol{x} \in \mathbb{R}^n : g_j(\boldsymbol{x}) \leq 0, \forall j \in [\![J]\!] \right\}.$$

Then under assumptions on g_j:

$$\mathcal{X} = \left\{ \boldsymbol{x} \in \mathbb{R}^n : g_j(\boldsymbol{x}') + \nabla g_j(\boldsymbol{x}')^T(\boldsymbol{x} - \boldsymbol{x}') \leq 0, \forall \boldsymbol{x}' \in \mathbb{R}^n, j \in [\![J]\!] \right\}.$$

Any finite subset of linearization points yields a polyhedral outer approximation.

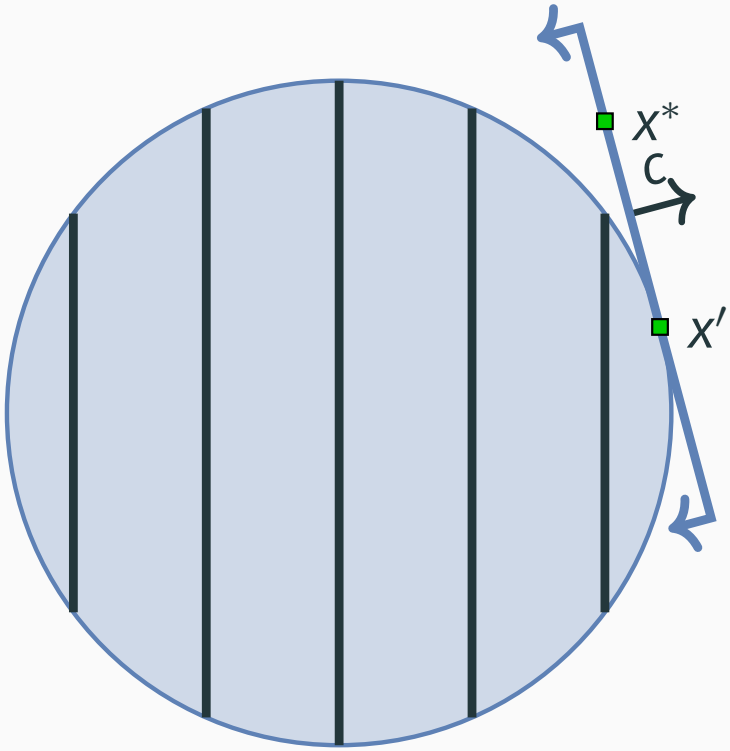
Given a polyhedral outer approximation $P \supset \mathcal{X}$, consider:

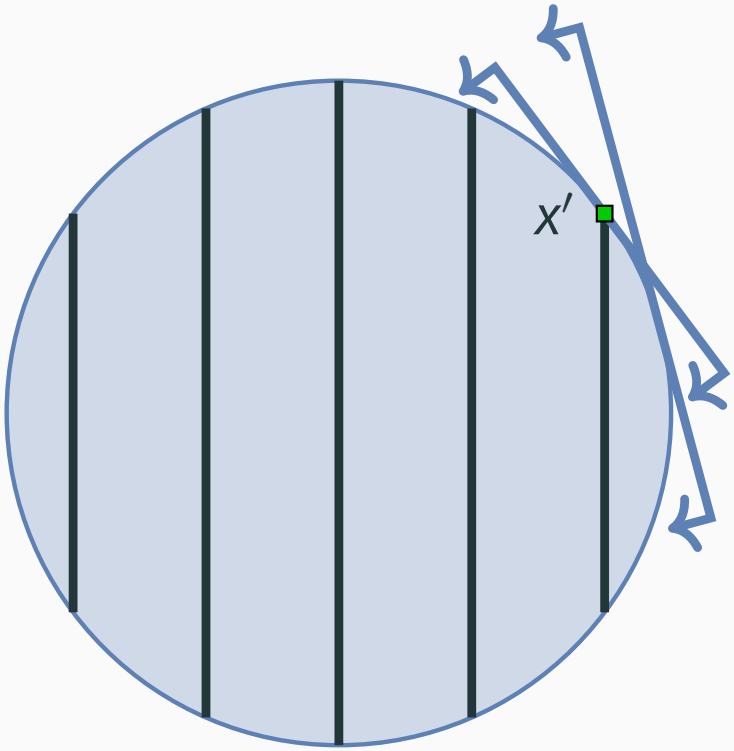
$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} : \\ & \boldsymbol{x} \in P, \\ & x_i \in \mathbb{Z}, \qquad\qquad \forall i \in [\![I]\!] . \end{aligned}$$

The above problem is a mixed integer linear relaxation.

Algorithmic idea: Iteratively improve the relaxation until lower bound matches feasible solution.

An illustration:





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 the ℓ_1 ball:

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

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

Standard LP extended formulation:

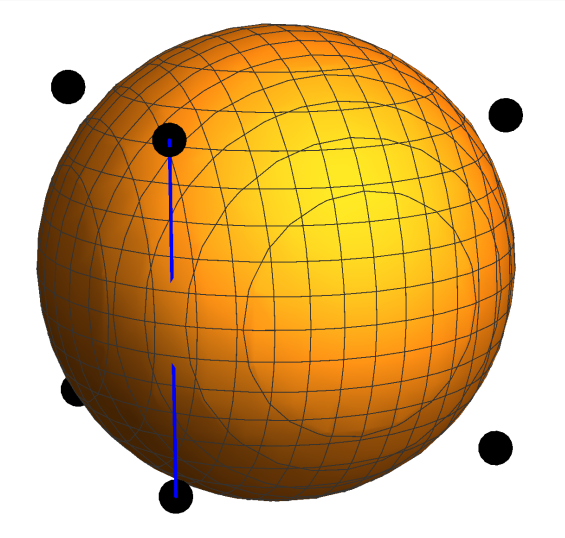
$$\mathcal{B}_1 = \left\{\boldsymbol{x} \in \mathbb{R}^n : \exists \boldsymbol{y} \in \mathbb{R}^n : -\boldsymbol{y} \leq \boldsymbol{x} \leq \boldsymbol{y}, \sum_{i=1}^{n} y_i \leq 1\right\}$$

The new formulation has $2n$ variables and $2n + 1$ constraints versus n variables and 2^n constraints.

The same happens with smooth constraints

Hijazi et al. (2014) consider the set:

$$\mathcal{B}_2 = \left\{ \boldsymbol{x} \in \{0,1\}^n : \sum_{i=1}^{n} \left(x_i - \frac{1}{2} \right)^2 \leq \frac{n-1}{4} \right\}$$



How to fix it?

Consider the equivalent extended formulation ($\mathcal{B}_2 = \text{proj}_x \hat{\mathcal{B}}_2$):

$$\hat{\mathcal{B}}_2 = \left\{ (\boldsymbol{x}, \boldsymbol{z}) \in \{0,1\}^n \times \mathbb{R}^n : \sum_{i=1}^{n} z_i \leq \frac{n-1}{4}, \left(x_i - \frac{1}{2} \right)^2 \leq z_i, \forall i \in [\![n]\!] \right\}$$

How to fix it?

Consider the equivalent extended formulation ($\mathcal{B}_2 = \text{proj}_x \hat{\mathcal{B}}_2$):

$$\hat{\mathcal{B}}_2 = \left\{ (\boldsymbol{x}, \boldsymbol{z}) \in \{0,1\}^n \times \mathbb{R}^n : \sum_{i=1}^{n} z_i \leq \frac{n-1}{4}, \left(x_i - \frac{1}{2} \right)^2 \leq z_i, \forall i \in [\![n]\!] \right\}$$

A polyhedral outer approximation in the space of $(\boldsymbol{x}, \boldsymbol{z})$ needs only $2n$ hyperplanes to exclude all integer points.

Say a user writes down a constraint:

$$z \geq f(\boldsymbol{x}) + g(\boldsymbol{x}).$$

If f and g are convex, then extended formulation is obtained by introducing t_1, t_2 such that:

$$z \geq t_1 + t_2, t_1 \geq f(\boldsymbol{x}), t_2 \geq g(\boldsymbol{x}).$$

But $f + g$ convex doesn't imply f and g convex (e.g., $f(\boldsymbol{x}) = x_1^2 - x_2^2$ and $g(\boldsymbol{x}) = 2x_2^2$).

Say a user writes down a constraint:

$$z \geq f(\boldsymbol{x}) + g(\boldsymbol{x}).$$

If f and g are convex, then extended formulation is obtained by introducing t_1, t_2 such that:

$$z \geq t_1 + t_2, t_1 \geq f(\boldsymbol{x}), t_2 \geq g(\boldsymbol{x}).$$

But $f + g$ convex doesn't imply f and g convex (e.g., $f(\boldsymbol{x}) = x_1^2 - x_2^2$ and $g(\boldsymbol{x}) = 2x_2^2$).

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 hard.

Say a user writes down a constraint:

$$z \geq f(\boldsymbol{x}) + g(\boldsymbol{x}).$$

If f and g are convex, then extended formulation is obtained by introducing t_1, t_2 such that:

$$z \geq t_1 + t_2, t_1 \geq f(\boldsymbol{x}), t_2 \geq g(\boldsymbol{x}).$$

But $f + g$ convex doesn't imply f and g convex (e.g., $f(\boldsymbol{x}) = x_1^2 - x_2^2$ and $g(\boldsymbol{x}) = 2x_2^2$).

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 hard. (But some solvers try.)

Conic optimization solves the problem

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T\boldsymbol{x} :$$
$$\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \in \mathcal{K}$$

A set $\mathcal{K} \subseteq \mathbb{R}^n$ is a **closed convex cone** if it is closed and contains all conic combinations of its points, i.e.,

$$\alpha_1 \boldsymbol{y}_1 + \alpha_2 \boldsymbol{y}_2 \in \mathcal{K} \qquad \forall \alpha_1, \alpha_2 \geq 0 \qquad \forall \boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{K}.$$

An example

The constraint

$$\log\left(\sum_{i=1}^{k}\exp(x_i)\right) \leq z$$

is convex, not separable. Converting it to conic form exposes the summation structure.

An example

The constraint

$$\log\left(\sum_{i=1}^{k}\exp(x_i)\right) \leq z$$

is convex, not separable. Converting it to conic form exposes the summation structure.

$$\sum_{i=1}^{k}\exp(x_i) \leq \exp(z)$$

$$\sum_{i=1}^{k}\exp(x_i - z) \leq 1$$

$$\exists \boldsymbol{y} : \sum_{i=1}^{k} y_i \leq 1 \text{ and } \forall\, i, (y_i, 1, x_i - z) \in \mathcal{E},$$

where $\mathcal{E} = \text{cone}(\{(r,t) : r \geq \exp(t)\})$.

In general...

- “Lectures on Modern Convex Optimization” by Ben-Tal and Nemirovski is the canonical source for how to write a problem in conic form.
- Disciplined convex programming (DCP) provides a systematic and easy-to-use approach to translating algebraic expressions into conic form (CVX, CVXPY, `Convex.jl`)
- If not convinced, read more at “Polyhedral Approximation in Mixed-integer Convex Optimization”, Math Programming B, 2017.

Which cones are needed?

- Nonnegative orthant $\mathbb{R}^n_+ = \{\boldsymbol{t} \in \mathbb{R}^n : \boldsymbol{t} \geq \boldsymbol{0}\}$
- Second-order cone $\mathcal{L}^{1+n} = \left\{(r, \boldsymbol{t}) \in \mathbb{R}^{1+n} : r \geq \|\boldsymbol{t}\|_2\right\}$
- Exponential cone $\mathcal{E} = \mathrm{cl}\left(\left\{(r, s, t) \in \mathbb{R}^3 : s > 0, r \geq s \exp\left(\frac{t}{s}\right)\right\}\right)$
- PSD cone $\mathbb{S}^n_+ = \{\boldsymbol{T} \in \mathbb{S}^n : \lambda_{\min}(\boldsymbol{T}) \geq 0\}$
- Power cone $\mathcal{W}_\alpha = \{(x, y, z) \in \mathbb{R}^3 : |z| \leq x^\alpha y^{1-\alpha}, x \geq 0, y \geq 0\}$

We categorized the 333 mixed-integer convex instances in the MINLPLIB2 library according to conic representability. Excluding $\mathbb{R}^n_+$, the following cones are needed:

$\mathcal{L}$ only	$\mathcal{E}$ only	$\mathcal{L}$ and $\mathcal{E}$	$\mathcal{W}$ only	Total
217	107	7	2	333

Open problems solved from MINLPLIB2

- `gams01`, $\mathcal{L}$+$\mathcal{E}$-representable
  - 145 variables, 1268 constraints (1158 linear)
  - Using CPLEX as MILP solver and KNITRO as convex solver, iterative OA solved in 6 iterations ($<$ 10 hours)
  - Optimal solution is 21380.20 (previous best bound was 1735.06, and best known solution was 21516.83)
- `tls5` and `tls6`, $\mathcal{L}$-representable, unsolved since 2001
  - Solved in less than 24h by converting to MISOCP and then using `Gurobi`

Mosek 9 adds support for the exponential and power cones (first commercial solver).

`https://docs.mosek.com/slides/2018/ismp2018/ismp-andersen.pdf`

Recall:

- Iterative OA: Solve a sequence of MILP relaxations
- LP-based B&B: Use a single B&B tree with relaxations computed by solving LPs

How to generate cuts in the conic case?

A set $\mathcal{K} \subseteq \mathbb{R}^n$ is a **closed convex cone** if it is closed and contains all conic combinations of its points, i.e.,

$$\alpha_1 \boldsymbol{y}_1 + \alpha_2 \boldsymbol{y}_2 \in \mathcal{K} \qquad \forall \alpha_1, \alpha_2 \geq 0 \qquad \forall \boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{K}. \tag{1}$$

The **dual cone** of a set is defined as:

$$\mathcal{K}^* = \left\{ \boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{y}^T \boldsymbol{z} \geq 0, \forall \boldsymbol{y} \in \mathcal{K} \right\}.$$

If $\mathcal{K}$ is a closed convex cone, then:

$$\boldsymbol{y} \in \mathcal{K} \Leftrightarrow \boldsymbol{y}^T \boldsymbol{z} \geq 0, \forall \boldsymbol{z} \in \mathcal{K}^*.$$

Any finite subset $\mathcal{Z} \subseteq \mathcal{K}^*$ yields a polyhedral outer approximation of $\mathcal{K}$. That is:

$$P := \left\{ \boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x}^T \boldsymbol{z} \geq 0, \forall z \in \mathcal{Z} \right\}.$$

P is a polyhedron that contains $\mathcal{K}$.

We say that an element $\boldsymbol{z} \in \mathcal{K}^*$ corresponds to a $\mathcal{K}^*$ cut.

- The second-order cone $\mathcal{L}$ and PSD cone $\mathcal{P}$ are **self-dual**
- The exponential cone $\mathcal{E}$ is *not* self dual, but $\mathcal{E}^*$ is easily described
- $\mathcal{K}^*$ cuts can be validated by checking membership in dual cone
- $\mathcal{K}^*$ cuts can be safely rounded by projection onto boundary of dual cone

Compare with gradient inequalities:

$$\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : g(\boldsymbol{x}) \leq \boldsymbol{0}\}.$$

Then $\boldsymbol{\alpha}^T \boldsymbol{x} \leq \beta$ is a gradient inequality iff there exists $\boldsymbol{x}' \in \mathbb{R}^n$ such that:

$$\begin{aligned} \boldsymbol{\alpha} &= \nabla g_j(\boldsymbol{x}'), \\ \beta &= \nabla g_j(\boldsymbol{x}')^T \boldsymbol{x}' - g(\boldsymbol{x}'). \end{aligned}$$

This is much harder to check.

Conic duality

Primal

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T\boldsymbol{x} :$$
$$\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \in \mathcal{K}$$

Dual

$$\max_{\boldsymbol{z}} \quad -\boldsymbol{b}^T\boldsymbol{z} :$$
$$\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{z} = 0$$
$$\boldsymbol{z} \in \mathcal{K}^*$$

Reduces to LP duality when $\mathcal{K} = \mathbb{R}^N_+$.

Conic duality can fail

The following MISOCP instance cannot be solved by the iterative OA algorithm:

$$\begin{aligned} \min_{x,y,z} \quad & z : \\ & xy \geq z^2, \\ & x = 0, \\ & y \geq 0, \\ & x \in \{0, 1\}. \end{aligned}$$

Conic duality can fail

The following MISOCP instance cannot be solved by the iterative OA algorithm:

$$\begin{aligned} \min_{x,y,z} \quad & z : \\ & xy \geq z^2, \\ & x = 0, \\ & y \geq 0, \\ & x \in \{0, 1\}. \end{aligned}$$

Why not?

- The optimal solution is zero
- The dual of the root node relaxation is infeasible
- *Any* polyhedral OA is unbounded below

MI-conic form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} : \\ & \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \in \mathcal{K} \\ & x_i \in \mathbb{Z} \qquad\qquad \forall i \in [\![I]\!] \end{aligned}$$

(When applying B&B, we also assume known lower and upper bounds on the integer-constrained variables.)

Certificate $\mathcal{K}^*$ cuts

Suppose we solve:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} : \\ & \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \in \mathcal{K} && (\boldsymbol{z}) \\ & l_i \leq x_i \leq u_i && \forall i \in [\![I]\!] && \text{(assuming } l_i \leq u_i\text{)} \end{aligned}$$

and obtain an optimal objective value T and dual solution $\boldsymbol{z}^*$ (ignoring multipliers on inequalities). Then

$$\begin{aligned} T = \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} : \\ & (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T\boldsymbol{z}^* \geq 0 \\ & l_i \leq x_i \leq u_i && \forall i \in [\![I]\!] \end{aligned}$$

Iterative OA uses certificate cuts with integer variables fixed to the optimal solution returned from the MILP relaxation. ($\Rightarrow$ finite convergence)

LP-based B&B uses conic certificate cuts at multiple points in the tree. Must use at least whenever a potential new incumbent integer feasible solution is found.

Recall $\alpha \boldsymbol{z}^* \in \mathcal{K}^* \forall \alpha \geq 0$. Does it matter how we scale $\boldsymbol{z}^*$?

LP solvers almost never guarantee exact feasibility. Suppose instead that we solve,

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} : \\ \alpha(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T \boldsymbol{z}^* &\geq -\epsilon_{feas} \\ l_i \leq x_i &\leq u_i \qquad \forall i \in [\![I]\!] \end{aligned}$$

This has optimal value $\geq T - \frac{\epsilon_{feas}}{\alpha}$!

Recall $\alpha \boldsymbol{z}^* \in \mathcal{K}^* \forall \alpha \geq 0$. Does it matter how we scale $\boldsymbol{z}^*$?

LP solvers almost never guarantee exact feasibility. Suppose instead that we solve,

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} : \\ \alpha(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T \boldsymbol{z}^* &\geq -\epsilon_{feas} \\ l_i \leq x_i &\leq u_i \qquad \forall i \in [\![I]\!] \end{aligned}$$

This has optimal value $\geq T - \frac{\epsilon_{feas}}{\alpha}$!

So choose α to make $\frac{\epsilon_{feas}}{\alpha}$ on the scale of the optimal objective, e.g., $0.01T$.

Separation $\mathcal{K}^*$ cuts

Given $\boldsymbol{x}$ such that $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \notin \mathcal{K}$, any hyperplane that separates $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ from $\mathcal{K}$ can be shifted (if needed) to become a $\mathcal{K}^*$ cut.

Example: If $\boldsymbol{T} \notin \mathbb{S}_+^n$ then $\lambda_{\min}(\boldsymbol{T}) < 0$. Let $\boldsymbol{\tau}$ be an eigenvector corresponding to the smallest eigenvalue of $\boldsymbol{T}$. Then:

$$\boldsymbol{\tau}^T \boldsymbol{\tau} \cdot \boldsymbol{T} = \lambda_{\min}(\boldsymbol{T}) < 0, \tag{2}$$

so $\boldsymbol{\tau}\boldsymbol{\tau}^T \in (\mathbb{S}_+^n)^*$ corresponds to a $\mathcal{K}^*$ cut that separates $\boldsymbol{T}$ from $\mathbb{S}_+^n$.

Example: If $\boldsymbol{T} \notin \mathbb{S}^n_+$ then $\lambda_{\min}(\boldsymbol{T}) < 0$. Let $\boldsymbol{\tau}$ be an eigenvector corresponding to the smallest eigenvalue of $\boldsymbol{T}$. Then:

$$\boldsymbol{\tau}^T \boldsymbol{\tau} \cdot \boldsymbol{T} = \lambda_{\min}(\boldsymbol{T}) < 0, \tag{2}$$

so $\boldsymbol{\tau}\boldsymbol{\tau}^T \in (\mathbb{S}^n_+)^*$ corresponds to a $\mathcal{K}^*$ cut that separates $\boldsymbol{T}$ from $\mathbb{S}^n_+$.

`SCIP-SDP` (Gally et al., 2017) uses these cuts for MISDP. `SCIP`, `Gurobi`, `CPLEX`, and `Xpress` use separation cuts for MISOCP.

Extensions to $\mathcal{K}^*$ cuts

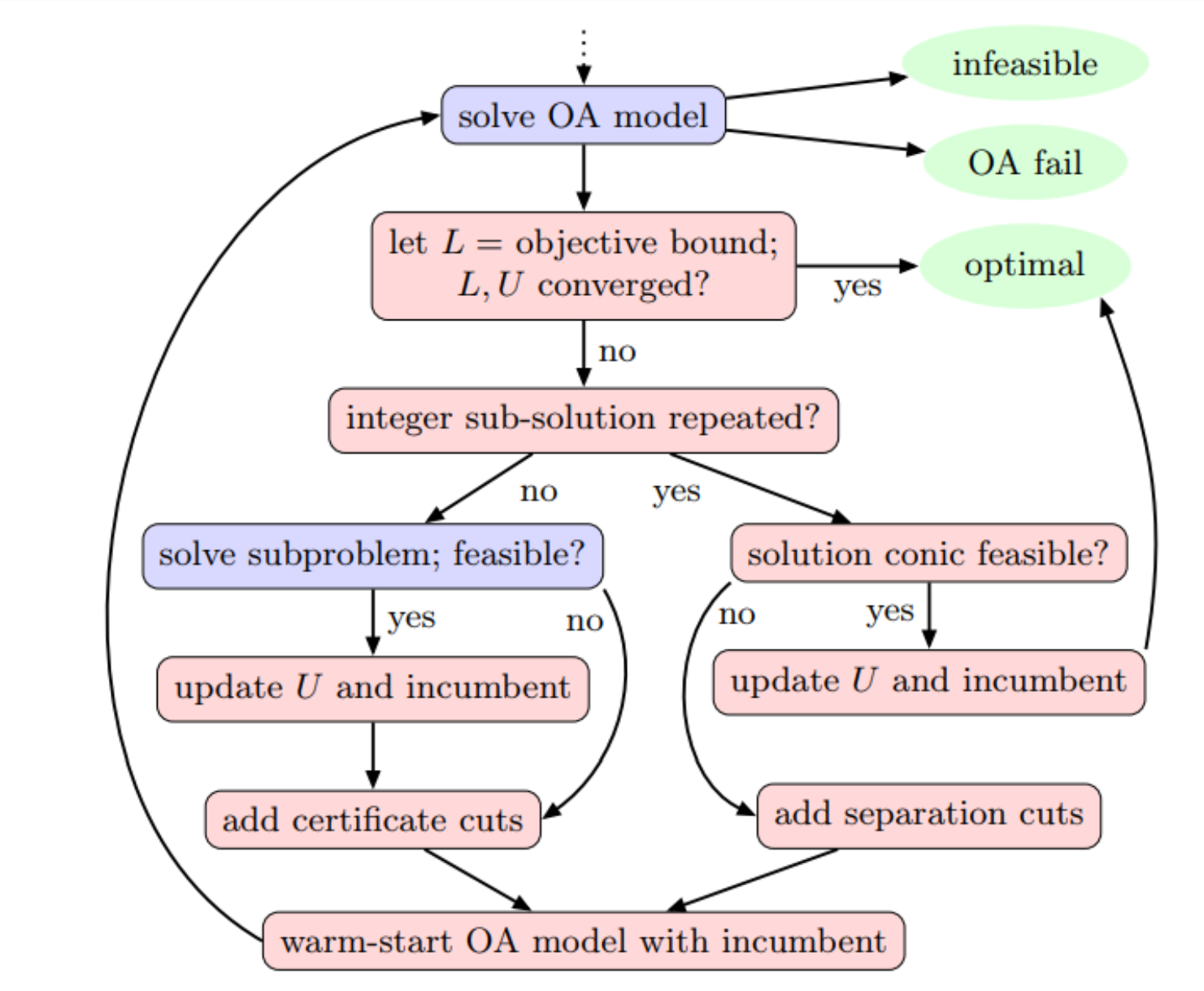
Natural extensions to the $\mathcal{K}^*$ cuts framework include:

1. Obtaining multiple $\mathcal{K}^*$ cuts from a single one by decomposing it by products of cones or into extreme rays
2. Second-order cone outer approximation of the PSD cone
3. Lifting of the second-order cone (Vielma et al., 2016)

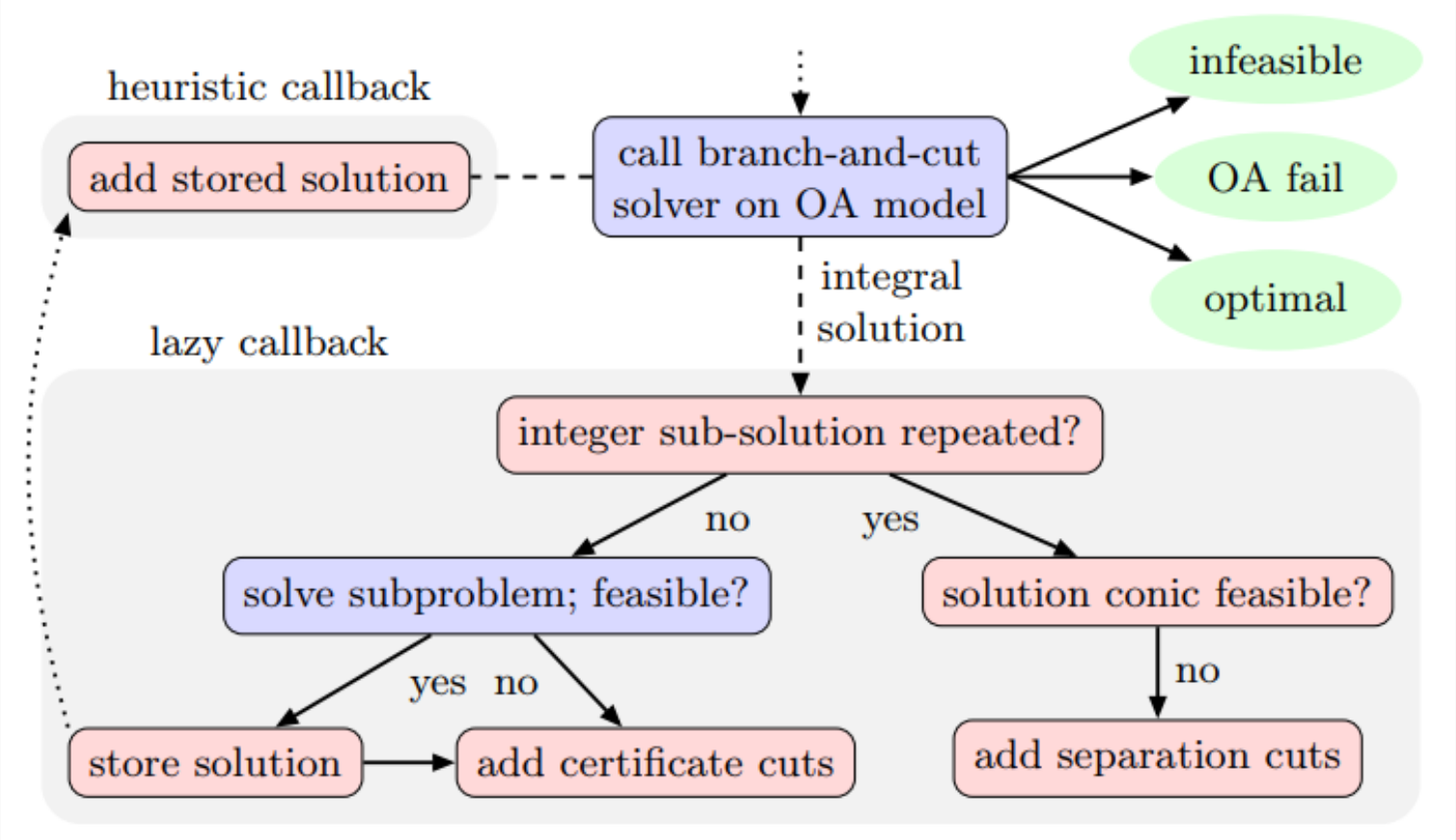
Pajarito

- An open-source MICP solver released in 2016
- Written in Julia, accessible through JuMP and via CBF files
- Can use any conic and MILP solver available to JuMP
- Implements iterative OA and LP-based B&B using callbacks

Note: For convex MINLP see Pavito. This functionality was split off last year from Pajarito.



The iterative OA algorithm in Pajarito.



The LP-based B&B algorithm in Pajarito. We call it MIP-solver-driven (MSD) because the MIP solver drives the algorithm.

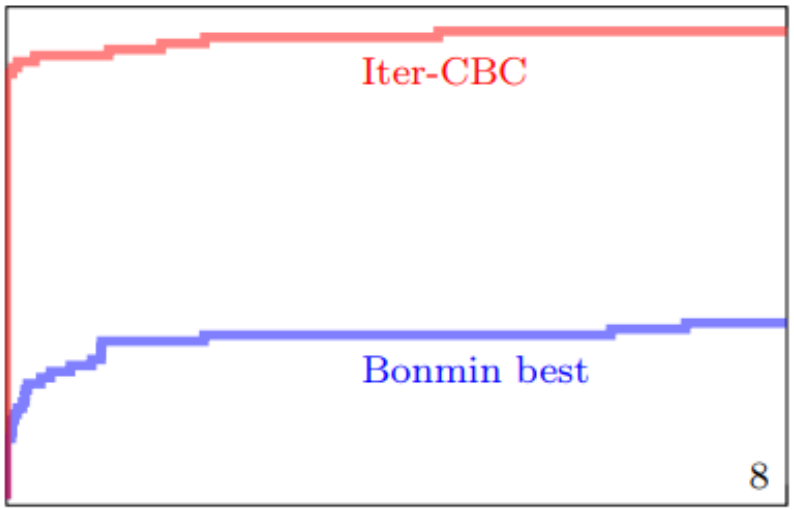
Numerical experiments

- Selection of 120 MISOCP instances from CBLIB library
- Solutions with large feasibility violations marked as ‘excluded’
- ‘conv‘ means converged, ‘error’ means failed to converge, and ‘limit’ means timeout
- Time summaries presented as shifted geometric means $(\prod_{i=1}^{n}(t_i + s))^{\frac{1}{n}} - s$ with $s = 10$ seconds

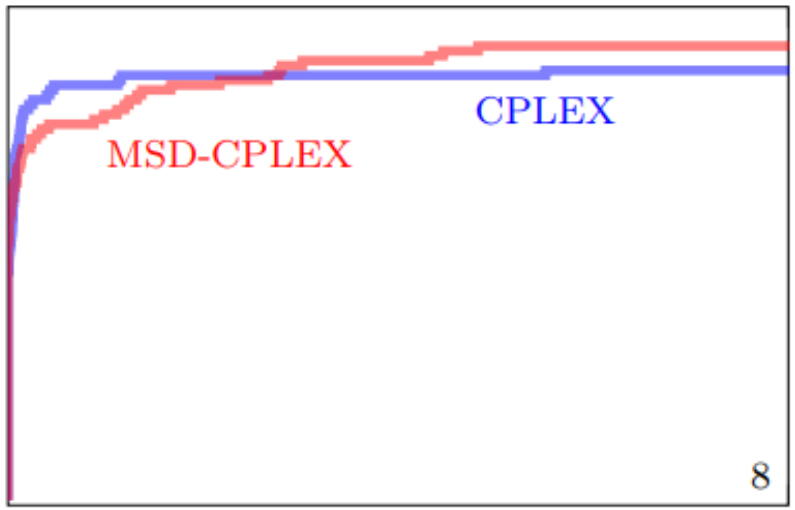
Status summary

		statuses				time (s)
	solver	conv	limit	error	excluded	all conv
open source	Bonmin-BB	34	44	11	31	38.0
	Bonmin-OA	25	53	29	13	64.2
	Bonmin-OA-D	30	48	29	13	15.1
	Iter-GLPK	56	60	3	1	2.0
	Iter-CBC	78	30	3	9	1.6
restricted	SCIP	74	35	8	3	3.2
	CPLEX	90	16	5	9	0.9
	Iter-CPLEX	86	26	0	8	0.4
	MSD-CPLEX	97	20	2	1	0.4

Performance profiles



(a) Open source Bonmin (instance-wise best of 3) and Pajarito iterative solvers.



(b) CPLEX MISOCP and Pajarito MSD solvers.

Test of algorithic variants on 95 MICP instances with a variety of cones:

- Experimental design: PSD cone and exponential cone
- Portfolio problems with risk constraints: exponential cone (entropy risk), second-order cone (norm risk), PSD cone (robust norm risk)
- Retrofit-synthesis of process networks: exponential cone, from MINLPLIB2
- Representative MISOCP instances from CBLIB

Effect of certificate cut scaling

	scale	statuses			
		conv	limit	error	excluded
Iter	off	63	1	28	3
	on	69	1	22	3
MSD	off	60	0	30	5
	on	67	0	26	2

Significant improvement in reliability. Effect on solve time (not shown) is hard to tell.

Thanks!

Joint work with Emre Yamangil, Chris Coey, Russell Bent, and Juan Pablo Vielma.

- “Extended Formulations in Mixed-integer Convex Programming”, IPCO 2016
- “Polyhedral Approximation in Mixed-integer Convex Optimization”, Math Programming B, 2017
- “Outer Approximation With Conic Certificates For Mixed-Integer Convex Problems”, arXiv:1808.05290