Pajarito Solver for Mixed-Integer Convex Optimization

Chris Coey

Operations Research Center
Massachusetts Institute of Technology

JuMP Developers Meetup
June 12, 2017

Collaborators Miles Lubin & Juan Pablo Vielma (MIT),
Emre Yamangil & Russell Bent (LANL)

Repository `github.com/JuliaOpt/Pajarito.jl`

1. Mixed-integer convex optimization
2. Conic outer approximation
3. Pajarito mixed-integer conic solver
4. The future of Pajarito
5. An interactive look at the codebase

1. Mixed-integer convex optimization
2. Conic outer approximation
3. Pajarito mixed-integer conic solver
4. The future of Pajarito
5. An interactive look at the codebase

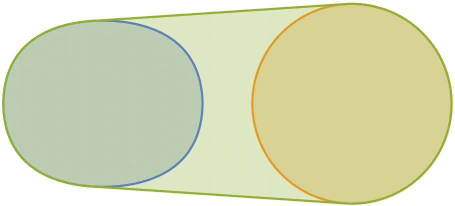
Mixed-integer convex optimization (MICP)

- a.k.a 'convex mixed-integer nonlinear programming' [BKL12]
- problems that are convex except for integrality constraints
- generalizes convex optimization and mixed-integer linear optimization

Mixed-integer convex optimization (MICP)

- a.k.a ‘convex mixed-integer nonlinear programming’ [BKL12]
- problems that are convex except for integrality constraints
- generalizes convex optimization and mixed-integer linear optimization

Many useful nonconvex sets are representable as feasible sets of MICPs, e.g. finite unions of compact convex sets [LZV16]



MICP general form and applications

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : \qquad \text{(linear objective)}$$

$$\boldsymbol{x} \in \mathcal{S} \qquad \text{(convex set constraints)}$$

$$x_i \in \mathbb{Z} \qquad \forall i \in [I] \qquad \text{(integrality constraints)}$$

MICP general form and applications

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : \qquad \text{(linear objective)}$$

$$\boldsymbol{x} \in \mathcal{S} \qquad \text{(convex set constraints)}$$

$$x_i \in \mathbb{Z} \qquad \forall i \in [I] \qquad \text{(integrality constraints)}$$

- quadratic facility location, stochastic service system design, cutting stock and constrained layout problems [BKL12]
- optimal discrete experimental design; see Appendix 6
- trajectory planning with spatial segmentation and sum-of-squares (SOS) control theory [DT15]
- portfolios with nonlinear risk measures and combinatorial constraints
- transistor gate-sizing for electrical circuit design [BKVH07]

A simple polyhedral outer approximation algorithm

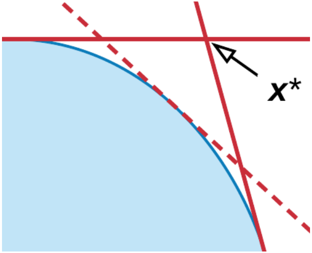
- mixed-integer linear optimization (MILP) solvers (such as **SCIP**, **Gurobi**, **CPLEX**) are mature, powerful, and numerically stable, enabling reliable cutting plane algorithms
- polyhedral outer approximation allows leveraging this power for MICP

A simple polyhedral outer approximation algorithm

- mixed-integer linear optimization (MILP) solvers (such as **SCIP**, **Gurobi**, **CPLEX**) are mature, powerful, and numerically stable, enabling reliable cutting plane algorithms
- polyhedral outer approximation allows leveraging this power for MICP

Build MILP OA model $\mathfrak{P}$ by replacing $\mathcal{S}$ with a polyhedral relaxation

1: solve $\mathfrak{P}$, let $\boldsymbol{x}^\star$ be optimal solution
2: **if** $\boldsymbol{x}^\star$ is 'close' to $\mathcal{S}$ **then**
3: return $\boldsymbol{x}^\star$
4: **else**
5: find separating hyperplane $(\boldsymbol{y}, z)$
6: update $\mathfrak{P}$ with cut $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \geq z$
7: **end if**



Conic extended formulations

- an extended formulation (EF) for $\boldsymbol{x} \in \mathcal{S}$ is an equivalent representation as a projection of a set in a higher dimensional space
- EFs can greatly accelerate OA algorithms [TS05, VDHL16]
- [LYBV16] noted that disciplined convex programming (DCP) implementations (such as **Convex.jl**) can automate the construction of convex conic extended formulations
- all 333 MICPs in MINLPLIB2 can be encoded with about 5 cone types
- with cones, we are not limited to sets defined by smooth, differentiable convex functions (many other MICP algorithms assume this)

Conic extended formulations

- an extended formulation (EF) for $\boldsymbol{x} \in \mathcal{S}$ is an equivalent representation as a projection of a set in a higher dimensional space
- EFs can greatly accelerate OA algorithms [TS05, VDHL16]
- [LYBV16] noted that disciplined convex programming (DCP) implementations (such as **Convex.jl**) can automate the construction of convex conic extended formulations
- all 333 MICPs in MINLPLIB2 can be encoded with about 5 cone types
- with cones, we are not limited to sets defined by smooth, differentiable convex functions (many other MICP algorithms assume this)

In [CLV17] we detail a conic framework for solving MICPs via OA and extended formulations, and implement our algorithms in **Pajarito**

Mixed-integer conic form

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : \tag{$\mathfrak{M}$}$$
$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \forall k \in [M]$$
$$x_i \in \mathbb{Z} \qquad \forall i \in [I]$$

- $\mathcal{C}_{K+1}, \ldots, \mathcal{C}_M$ are polyhedral cones, e.g. $\mathbb{R}_+$, $\mathbb{R}_-$, $\{0\}$
- $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are closed convex nonpolyhedral cones, e.g.

Mixed-integer conic form

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : \tag{$\mathfrak{M}$}$$

$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \forall k \in [M]$$

$$x_i \in \mathbb{Z} \qquad \forall i \in [I]$$

- $\mathcal{C}_{K+1}, \ldots, \mathcal{C}_M$ are polyhedral cones, e.g. $\mathbb{R}_+$, $\mathbb{R}_-$, $\{0\}$
- $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are closed convex nonpolyhedral cones, e.g.

  $\mathcal{L}$ second-order cone (epi $\|\cdot\|_2$)

  $\mathcal{E}$ exponential cone (epi cl per exp)

  $\mathcal{P}$ positive semidefinite cone ($\mathbb{S}_+$ on $\mathbb{S}$)

Mixed-integer conic form

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : \qquad (\mathfrak{M})$$

$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \forall k \in [M]$$

$$x_i \in \mathbb{Z} \qquad \forall i \in [I]$$

- $\mathcal{C}_{K+1}, \ldots, \mathcal{C}_M$ are polyhedral cones, e.g. $\mathbb{R}_+$, $\mathbb{R}_-$, $\{0\}$
- $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are closed convex nonpolyhedral cones, e.g.
  - $\mathcal{L}$ second-order cone (epi $\|\cdot\|_2$)
  - $\mathcal{E}$ exponential cone (epi cl per exp)
  - $\mathcal{P}$ positive semidefinite cone ($\mathbb{S}_+$ on $\mathbb{S}$)

Assume that if $\mathfrak{M}$ is feasible then its optimal value is attained

Outer approximation with $\mathcal{K}^*$ cuts

The dual cone of a closed convex cone is also a closed convex cone

$$\mathcal{K}^* = \{\boldsymbol{z} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{z} \rangle \geq 0, \forall \boldsymbol{y} \in \mathcal{K}\}$$

Outer approximation with $\mathcal{K}^*$ cuts

The dual cone of a closed convex cone is also a closed convex cone

$$\mathcal{K}^* = \{\boldsymbol{z} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{z} \rangle \geq 0, \forall \boldsymbol{y} \in \mathcal{K}\}$$

Thus we can reformulate nonpolyhedral conic constraint $k \in [K]$ as an infinite number of linear constraints - one for each dual cone point

$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \Leftrightarrow \qquad \langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \geq 0 \quad \forall \boldsymbol{z}_k \in \mathcal{C}_k^*$$

Outer approximation with $\mathcal{K}^*$ cuts

The dual cone of a closed convex cone is also a closed convex cone

$$\mathcal{K}^* = \{\boldsymbol{z} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{z} \rangle \geq 0, \forall \boldsymbol{y} \in \mathcal{K}\}$$

Thus we can reformulate nonpolyhedral conic constraint $k \in [K]$ as an infinite number of linear constraints - one for each dual cone point

$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \Leftrightarrow \qquad \langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \geq 0 \quad \forall \boldsymbol{z}_k \in \mathcal{C}_k^*$$

For OA, we instead choose a finite subset $\mathcal{Z}_k$ of the dual cone points

$$\langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \geq 0 \qquad \forall \boldsymbol{z}_k \in \mathcal{Z}_k \subset \mathcal{C}_k^*$$

Outer approximation with $\mathcal{K}^*$ cuts

The dual cone of a closed convex cone is also a closed convex cone

$$\mathcal{K}^* = \{\boldsymbol{z} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{z} \rangle \geq 0, \forall \boldsymbol{y} \in \mathcal{K}\}$$

Thus we can reformulate nonpolyhedral conic constraint $k \in [K]$ as an infinite number of linear constraints - one for each dual cone point

$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \Leftrightarrow \qquad \langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \geq 0 \quad \forall \boldsymbol{z}_k \in \mathcal{C}_k^*$$

For OA, we instead choose a finite subset $\mathcal{Z}_k$ of the dual cone points

$$\langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \geq 0 \qquad \forall \boldsymbol{z}_k \in \mathcal{Z}_k \subset \mathcal{C}_k^*$$

If added for each $k \in [K]$, these $\mathcal{K}^*$ cuts yield an MILP relaxation of $\mathfrak{M}$

Obtaining $\mathcal{K}^*$ cuts

There are various ways to choose $\mathcal{K}^*$ cuts (not unique)

Obtaining $\mathcal{K}^*$ cuts

There are various ways to choose $\mathcal{K}^*$ cuts (not unique)

If we solve continuous conic subproblems, we can get an algorithm with finite convergence guarantees, under some reasonable assumptions (see Appendix 7 for a detailed branch and bound algorithm)

Obtaining $\mathcal{K}^*$ cuts

There are various ways to choose $\mathcal{K}^*$ cuts (not unique)

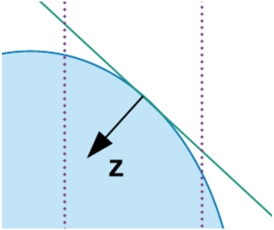
If we solve continuous conic subproblems, we can get an algorithm with finite convergence guarantees, under some reasonable assumptions (see Appendix 7 for a detailed branch and bound algorithm)

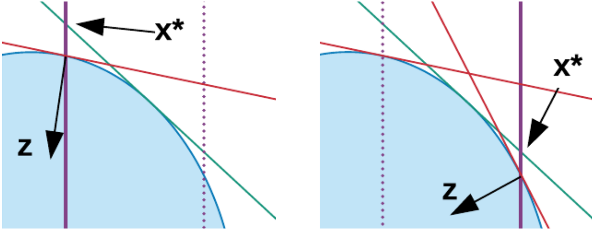
Define the following models

- $\mathfrak{M}$ the MICP problem
- $\mathfrak{P}$ the MILP OA model that we add $\mathcal{K}^*$ cuts to
- $\mathfrak{R}$ the continuous relaxation of $\mathfrak{M}$
- $\mathfrak{R}(\boldsymbol{x})$ a restriction of $\mathfrak{R}$ to the subspace in which the integer variables are fixed to $x_1, \ldots, x_I$

Geometric intuition

$\mathfrak{M}$: blue convex region intersected with purple dotted lines for integers
$\mathfrak{P}$: polyhedron under $\mathcal{K}^*$ cuts intersected with purple dotted lines





1. solve $\mathfrak{R}$ for dual $\bar{\boldsymbol{z}}$
2. add $\bar{\boldsymbol{z}}$ cut to $\mathfrak{P}$

1. solve $\mathfrak{P}$ for $\boldsymbol{x}^\star$
2. solve $\mathfrak{R}(\boldsymbol{x}^\star)$ for dual $\bar{\boldsymbol{z}}$
3. add $\bar{\boldsymbol{z}}$ cut to $\mathfrak{P}$
4. if $\mathfrak{R}(\boldsymbol{x}^\star)$ feasible, check if $\bar{\boldsymbol{x}}$ is new incumbent

The continuous relaxation

The continuous conic relaxation of $\mathfrak{M}$ is $\mathfrak{R}$

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \langle \boldsymbol{c}, \boldsymbol{x} \rangle : && && (\mathfrak{R}) \\ \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} &\in \mathcal{C}_k && \forall k \in [M] \\ \boldsymbol{x} &\in \mathbb{R}^N \end{aligned}$$

The continuous relaxation

The continuous conic relaxation of $\mathfrak{M}$ is $\mathfrak{R}$

$$\min_{\boldsymbol{x}} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : \tag{$\mathfrak{R}$}$$
$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \qquad \forall k \in [M]$$
$$\boldsymbol{x} \in \mathbb{R}^N$$

Using standard conic duality [BV04], the conic dual is $\mathfrak{R}^*$

$$\max_{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K} \quad - \sum_{k \in [M]} \langle \boldsymbol{b}_k, \boldsymbol{z}_k \rangle : \tag{$\mathfrak{R}^*$}$$
$$\boldsymbol{c} + \sum_{k \in [M]} \boldsymbol{A}_k^T \boldsymbol{z}_k \in \{0\}^N$$
$$\boldsymbol{z}_k \in \mathcal{C}_k^* \qquad \forall k \in [M]$$

Relaxation and subproblem $\mathcal{K}^*$ cuts

To obtain relaxation $\mathcal{K}^*$ cuts

- assume $\mathfrak{R}$ is bounded
- if $\mathfrak{R}$ is infeasible then $\mathfrak{M}$ is infeasible
- if $\mathfrak{R}$ is feasible, assume strong duality holds for $\mathfrak{R}, \mathfrak{R}^*$
  - exists primal-dual solutions with objective value C
  - from $\mathfrak{R}^*$ solution $(\bar{\boldsymbol{z}}_k)_{k \in [M]}$, we derive $\mathcal{K}^*$ cuts $\bar{\boldsymbol{z}}_k$ for $k \in [K]$
  - guarantee that $\mathfrak{P}$'s value is no worse than C

Relaxation and subproblem $\mathcal{K}^*$ cuts

To obtain relaxation $\mathcal{K}^*$ cuts

- assume $\mathfrak{R}$ is bounded
- if $\mathfrak{R}$ is infeasible then $\mathfrak{M}$ is infeasible
- if $\mathfrak{R}$ is feasible, assume strong duality holds for $\mathfrak{R}, \mathfrak{R}^*$
  - exists primal-dual solutions with objective value C
  - from $\mathfrak{R}^*$ solution $(\bar{\boldsymbol{z}}_k)_{k \in [M]}$, we derive $\mathcal{K}^*$ cuts $\bar{\boldsymbol{z}}_k$ for $k \in [K]$
  - guarantee that $\mathfrak{P}$'s value is no worse than C

To obtain subproblem $\mathcal{K}^*$ cuts given a feasible MILP solution $\boldsymbol{x}^\star$ for $\mathfrak{P}$

- note $\mathfrak{R}(\boldsymbol{x}^\star)$ is not unbounded
- if $\mathfrak{R}(\boldsymbol{x}^\star)$ is feasible, case is analogous to above for $\mathfrak{R}$
- if $\mathfrak{R}(\boldsymbol{x}^\star)$ is infeasible, we get a ray of $\mathfrak{R}^*(\boldsymbol{x}^\star)$ that defines $\mathcal{K}^*$ cuts excluding all $\boldsymbol{x}$ with the same integer assignment $x_1, \dots, x_I$

Algorithmic extensions

- scaling of subproblem $\mathcal{K}^*$ cuts to get convergence guarantees under realistic assumptions about MILP solver tolerances
- simple separation $\mathcal{K}^*$ cuts for infeasible OA solutions
- disaggregated extended formulation for $\mathcal{L}$ [VDHL16]
- initial fixed $\mathcal{K}^*$ cuts
  - for $\mathcal{L}$, based on the ℓ_1 and ℓ_∞ bounds for ℓ_2
  - for $\mathcal{P}$, an idea dual to the (scaled) diagonal dominance conditions for PSD matrices described by [AH15]
- extreme ray cuts, implemented for $\mathcal{P}$ based on eigendecompositions
- $\mathcal{L}$ subproblem cuts for $\mathcal{P}$, based on Schur complement [KKY03]

Algorithmic extensions

- scaling of subproblem $\mathcal{K}^*$ cuts to get convergence guarantees under realistic assumptions about MILP solver tolerances
- simple separation $\mathcal{K}^*$ cuts for infeasible OA solutions
- disaggregated extended formulation for $\mathcal{L}$ [VDHL16]
- initial fixed $\mathcal{K}^*$ cuts
  - for $\mathcal{L}$, based on the ℓ_1 and ℓ_∞ bounds for ℓ_2
  - for $\mathcal{P}$, an idea dual to the (scaled) diagonal dominance conditions for PSD matrices described by [AH15]
- extreme ray cuts, implemented for $\mathcal{P}$ based on eigendecompositions
- $\mathcal{L}$ subproblem cuts for $\mathcal{P}$, based on Schur complement [KKY03]

If using $\mathcal{L}$ cuts for $\mathcal{P}$, the OA MIP model is an MISOCP problem (can solve using **Pajarito**-in-**Pajarito**)

Pajarito mixed-integer conic solver

- open-source solver written in **Julia** and integrated with **JuliaOpt**
- uses the powerful **MathProgBase** abstraction layer
  - accepts mixed-integer conic input from multiple modeling packages
  - calls MIP and continuous conic solvers in a solver-independent way
- currently supports 3 nonpolyhedral cones: $\mathcal{L}, \mathcal{E}, \mathcal{P}$
- around 30 algorithmic options (including the extensions described)

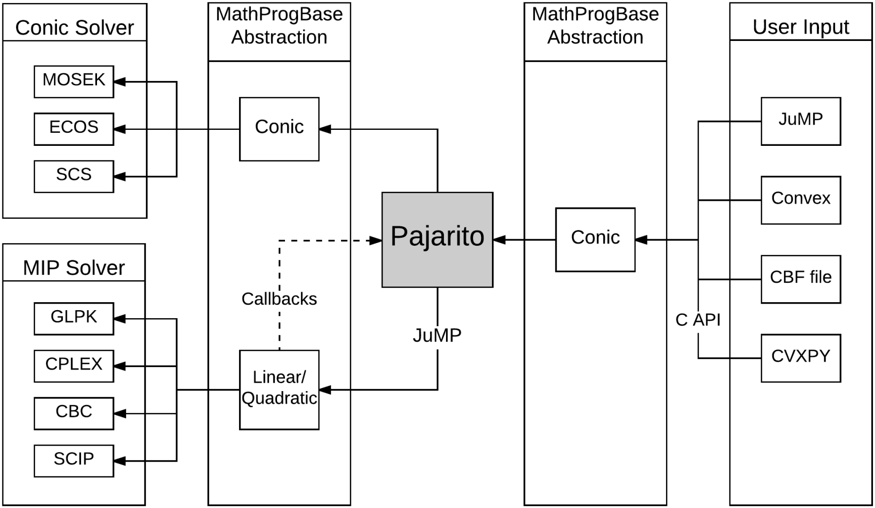
Pajarito mixed-integer conic solver

- open-source solver written in **Julia** and integrated with **JuliaOpt**
- uses the powerful **MathProgBase** abstraction layer
  - accepts mixed-integer conic input from multiple modeling packages
  - calls MIP and continuous conic solvers in a solver-independent way
- currently supports 3 nonpolyhedral cones: $\mathcal{L}, \mathcal{E}, \mathcal{P}$
- around 30 algorithmic options (including the extensions described)

See Appendix 8 for a summary of preliminary testing on 120 nontrivial MISOCP problems from CBLIB

- using **CPLEX**'s MILP solver and **MOSEK**'s SOCP solver, beat **CPLEX**'s specialized MISOCP solver
- using open-source sub-solvers, very convincingly beat **BONMIN**

Integration with MathProgBase



Accessing Pajarito: MathProgBase conic form

- traditional ‘convex MINLP’ solvers interact with the problem through oracles to query values and derivatives of constraints and objective
- this means complicated data structures and interfaces
- **Pajarito**’s conic algorithm takes the conic form problem $\mathfrak{M}$: two vectors $\boldsymbol{c}, \boldsymbol{b}$, a (sparse) matrix $\boldsymbol{A}$, and two lists of (predefined) cones
- this compact representation makes the solver interface very straightforward

Accessing Pajarito: MathProgBase conic form

Access **Pajarito** from algebraic modeling packages and conic formats

- **JuMP** - supports $\mathcal{L}$ and $\mathcal{P}$ cones only
- **Convex.jl** [UMZ+14] - a DCP implementation, does automatic conic extended formulations
- **CVXPY** [DB16] through the C API **cmpb**, thanks to Steven Diamond, Baris Ungun
- CBF proposed by Henrik Friberg [Fri16]
  - v2 support $\mathcal{L}, \mathcal{E}, \mathcal{P}$
  - encodes our benchmark and unit test problems
  - **ConicBenchmarkUtilties.jl** translates between conic format and CBF

Internal use of MathProgBase and JuMP

- uses the conic interface to interact with continuous conic solvers
- directly manipulates the conic data and uses `setbvec!`
- uses **JuMP** to manage the MIP OA model and interact with MIP solvers

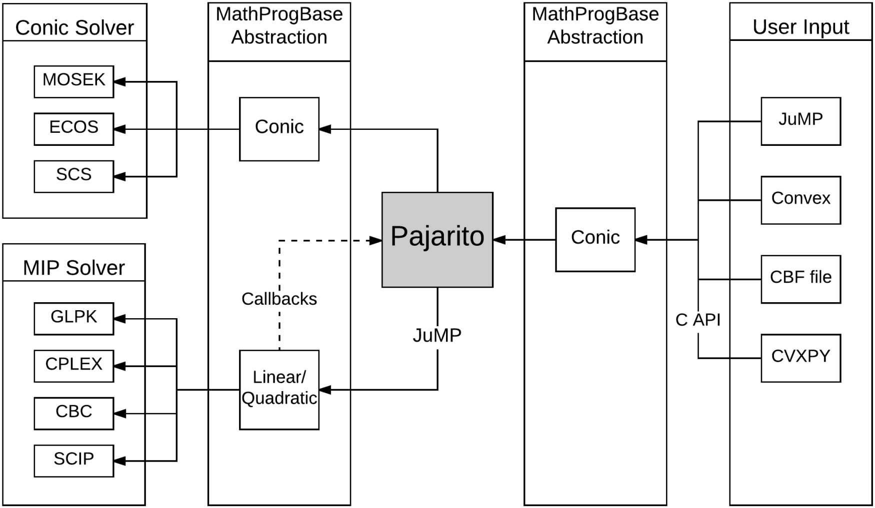
Internal use of MathProgBase and JuMP

- uses the conic interface to interact with continuous conic solvers
- directly manipulates the conic data and uses `setbvec!`
- uses **JuMP** to manage the MIP OA model and interact with MIP solvers

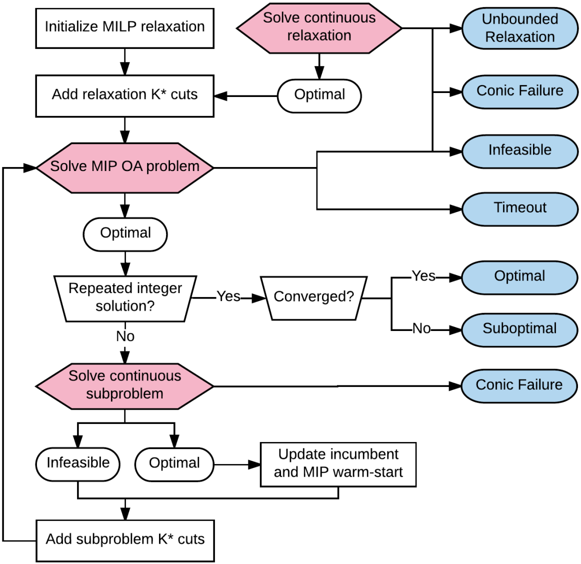
MathProgBase does not attempt to provide an abstraction for solver parameters like convergence tolerances

- user's responsibility to set tolerances on MIP and conic continuous solvers
- adjust the MILP solver's linear feasibility tolerance and integer feasibility tolerance for improved convergence behavior

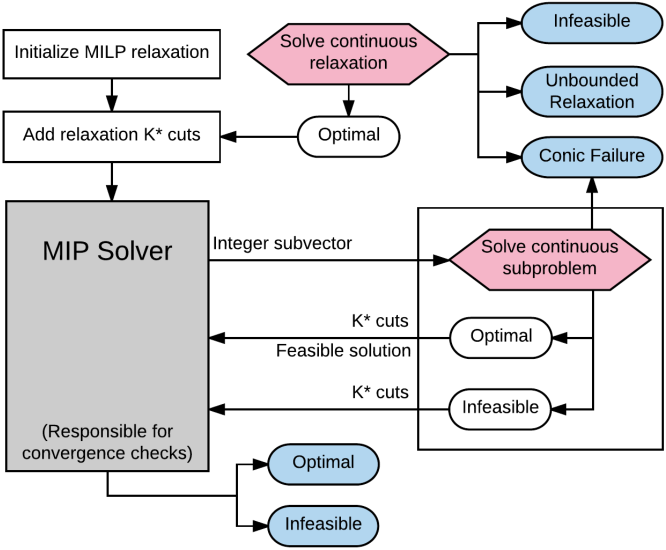
Integration with MathProgBase



Iterative OA algorithm



MIP-solver-driven OA algorithm



Issues with MIP callbacks in MSD algorithm

- the **MathProgBase** MIP callback abstraction was designed primarily around shared behavior between **CPLEX** and **Gurobi**
- MILP solver may choose to ignore lazy cuts for numerical reasons and accept its integer-feasible solution
- **CPLEX** and **SCIP** allow explicit rejection of the solution with incumbent callbacks, but these are not currently accessible
- through lazy callbacks we do not have the ability to provide new incumbents to the solver
- we use the heuristic callback, but no guarantees on when it is called

Issues with MIP callbacks in MSD algorithm

- the **MathProgBase** MIP callback abstraction was designed primarily around shared behavior between **CPLEX** and **Gurobi**
- MILP solver may choose to ignore lazy cuts for numerical reasons and accept its integer-feasible solution
- **CPLEX** and **SCIP** allow explicit rejection of the solution with incumbent callbacks, but these are not currently accessible
- through lazy callbacks we do not have the ability to provide new incumbents to the solver
- we use the heuristic callback, but no guarantees on when it is called

For nominal correctness of MSD we must be able to update the incumbent before the next node is processed

The future of Pajarito

- **MathProgBase** status changes will fix some internal issues and necessitate rethinking **Pajarito** return statuses
- **Convex.jl** does not yet support **Julia** 0.6
- **MathProgBase** set-inclusion models may allow a much-improved future version of **Pajarito**
  - atom-based modeling package extension for **JuMP** ('**AtomicJuMP**')?
  - automated extended formulations of set-inclusion models?
  - a continuous set-inclusion model solver? (primal-dual? certificates?)
- examples folder - please submit PRs with applications, after changes
- some basic documentation with **Documenter.jl**
- eventually **MathProgBase** callbacks should be rethought

Solver and model objects

Look at cardinality constrained least squares example (cardls.jl), which has both NLP and conic models

Solver and model objects

Look at cardinality constrained least squares example (cardls.jl), which has both NLP and conic models

- `using Pajarito`
- `s = PajaritoSolver()` - instantiate **Pajarito** solver `s`
- `m = Model(s)` - instantiate **Pajarito** model `m`, either:
  - `PajaritoConicModel<:AbstractConicModel`
  - `PajaritoNonlinearModel<:AbstractNonlinearModel`

Loading and solving a conic model

Use the following basic **MathProgBase** functions

- `loadproblem!(m, c, A, b, cone_con, cone_var)`
- `setvartype!(m, var_types)`
- `optimize!(m)`
- miscellaneous getters

Pajarito's unit tests

See the test folder

- define MILP/MISOCP and conic and NLP solvers
- use `@testset` to define nested sets of tests, iterating over combinations of solvers and options
- the conic tests use CBF instances in the CBF folder
- the CBF instances are created from **JuMP** models by calling a function from `ConicBenchmarkUtilities.jl`

Pajarito's unit tests

See the test folder

- define MILP/MISOCP and conic and NLP solvers
- use `@testset` to define nested sets of tests, iterating over combinations of solvers and options
- the conic tests use CBF instances in the CBF folder
- the CBF instances are created from **JuMP** models by calling a function from `ConicBenchmarkUtilities.jl`

Tests are fragile

- we rely on numerical solvers
- tolerances matter, but it's not always clear how
- we have identified incorrect solutions/statuses from all MIP solvers

Thank you

References I

Amir Ali Ahmadi and Georgina Hall, *Sum of squares basis pursuit with linear and second order cone programming*, arXiv preprint arXiv:1510.01597 (2015).

Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization **5** (2008), no. 2, 186–204.

Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth, *Algorithms and software for convex mixed integer nonlinear programs*, Mixed Integer Nonlinear Programming (Jon Lee and Sven Leyffer, eds.), The IMA Volumes in Mathematics and its Applications, vol. 154, Springer New York, 2012, pp. 1–39 (English).

Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi, *A tutorial on geometric programming*, Optimization and engineering **8** (2007), no. 1, 67.

References II

- Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski, *Robust optimization*, Princeton University Press, 2009.
- A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization*, Society for Industrial and Applied Mathematics, 2001.
- Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.
- Chris Coey, Miles Lubin, and Juan Pablo Vielma, *A conic framework for solving mixed-integer convex problems via outer approximation*, in preparation, 2017.
- Steven Diamond and Stephen Boyd, *Cvxpy: A python-embedded modeling language for convex optimization*, Journal of Machine Learning Research **17** (2016), no. 83, 1–5.

References III

Robin Deits and Russ Tedrake, *Efficient mixed-integer planning for uavs in cluttered environments*, Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2015, pp. 42–49.

Henrik A. Friberg, *CBLIB 2014: a benchmark library for conic mixed-integer and continuous optimization*, Mathematical Programming Computation **8** (2016), no. 2, 191–214.

Sunyoung Kim, Masakazu Kojima, and Makoto Yamashita, *Second order cone programming relaxation of a positive semidefinite constraint*, Optimization Methods and Software **18** (2003), no. 5, 535–541.

M. Lubin, E. Yamangil, R. Bent, and Juan Pablo Vielma, *Polyhedral approximation in mixed-integer convex optimization*, ArXiv e-prints (2016).

M. Lubin, I. Zadik, and J. P. Vielma, *Mixed-integer convex representability*, ArXiv e-prints (2016).

References IV

I. Quesada and I.E. Grossmann, *An lp/nlp based branch and bound algorithm for convex minlp optimization problems*, Computers & Chemical Engineering **16** (1992), no. 10, 937–947.

Mohit Tawarmalani and Nikolaos V. Sahinidis, *A polyhedral branch-and-cut approach to global optimization*, Mathematical Programming **103** (2005), no. 2, 225–249 (English).

Madeleine Udell, Karanveer Mohan, David Zeng, Jenny Hong, Steven Diamond, and Stephen Boyd, *Convex optimization in julia*, Proceedings of HPTCDL 14 (Piscataway, NJ, USA), IEEE Press, 2014, pp. 18–28.

Juan Pablo Vielma, Iain Dunning, Joey Huchette, and Miles Lubin, *Extended formulations in mixed integer conic quadratic programming*, Mathematical Programming Computation (2016), 1–50.

6. Example: experimental design
7. Branch and bound algorithm
8. Computational testing

Experimental design optimization

From [BV04]

- estimate a vector $\boldsymbol{x} \in \mathbb{R}^Q$
- budget of M experiments selected from a menu $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_P \in \mathbb{R}^Q$
- let m_p be the number times experiment $\boldsymbol{u}_p$ is run
- assume a linear model $\boldsymbol{u}'\boldsymbol{x}$ with Gaussian noise
- to maximize informativeness, 'minimize' the error covariance matrix (a function of the experiment choices $\boldsymbol{m}$), denoted $\boldsymbol{E}(\boldsymbol{m})$

Experimental design optimization

From [BV04]

- estimate a vector $\boldsymbol{x} \in \mathbb{R}^Q$
- budget of M experiments selected from a menu $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_P \in \mathbb{R}^Q$
- let m_p be the number times experiment $\boldsymbol{u}_p$ is run
- assume a linear model $\boldsymbol{u}'\boldsymbol{x}$ with Gaussian noise
- to maximize informativeness, 'minimize' the error covariance matrix (a function of the experiment choices $\boldsymbol{m}$), denoted $\boldsymbol{E}(\boldsymbol{m})$

$$\boldsymbol{E}(\boldsymbol{m}) \equiv \left(\sum_{p \in [P]} m_p \boldsymbol{u}_p \boldsymbol{u}_p' \right)^{-1}$$

Mixed-integer convex formulation

If $f : \mathbb{S}_+^Q \to \mathbb{R}$ measures the 'size' of the error covariance matrix $\boldsymbol{E}(\boldsymbol{m})$

$$
\begin{aligned}
\min_{\boldsymbol{m} \in \mathbb{R}^P} \quad & f(\boldsymbol{E}(\boldsymbol{m})) : && \text{minimize error covariance} \\
& \mathbf{1}'\boldsymbol{m} \leq M && \text{budget of experiments} \\
& \boldsymbol{m} \in \mathbb{Z}_+^P && \text{integrality restriction}
\end{aligned}
$$

Mixed-integer convex formulation

If $f : \mathbb{S}_+^Q \to \mathbb{R}$ measures the 'size' of the error covariance matrix $\boldsymbol{E}(\boldsymbol{m})$

$$\begin{aligned} \min_{\boldsymbol{m} \in \mathbb{R}^P} \quad & f(\boldsymbol{E}(\boldsymbol{m})) : && \text{minimize error covariance} \\ & \mathbf{1}'\boldsymbol{m} \leq M && \text{budget of experiments} \\ & \boldsymbol{m} \in \mathbb{Z}_+^P && \text{integrality restriction} \end{aligned}$$

If $\mathcal{E}$ is a confidence ellipsoid for $\boldsymbol{x}$ given $\boldsymbol{E}$, there are many choices for $f(\boldsymbol{E})$

- E-opt minimizes the diameter of $\mathcal{E}$: $\min \|\boldsymbol{E}\|_2$
- A-opt minimizes mean squared error: $\min \operatorname{tr} \boldsymbol{E}$
- D-opt minimizes the volume of $\mathcal{E}$: $\min \log\det \boldsymbol{E}$, or by [BTN01], maximizes scaled geomean eigenvalues of $\sum_{p \in [P]} m_p \boldsymbol{u}_p \boldsymbol{u}_p'$

A branch and bound OA algorithm

- a conic analogue of [QG92] (convex MINLP)
- assume we have explicit bounds $\boldsymbol{l}^0, \boldsymbol{u}^0$ on the integer variables $(x_i)_{i \in [I]}$
- recursively partition the possible assignments of integer variables by lower and upper bound vectors $\boldsymbol{l}, \boldsymbol{u}$
- add subproblem $\mathcal{K}^*$ cuts when we get integer solutions for $x_1, \ldots, x_I$ - globally valid and, if added to the LP relaxation, contain enough information to properly process the node
- solve linear programming relaxations with reliable (dual) simplex
  - requires few pivots after adding cuts
  - achieve very tight feasibility and optimality tolerances
- finite convergence if there is a finite number of integer assignments
  - finite number of nodes, each examined a finite number of times
  - if we add subproblem cuts at every node, assuming strong duality
  - then the optimal objective value of the final polyhedral OA model will equal that of the MICP problem

Processing nodes

Suppose we are at a node $(\boldsymbol{l}, \boldsymbol{u}, L)$ of the branch and bound tree

- $\boldsymbol{l}, \boldsymbol{u}$ are the node's lower, upper variable bounds for $\hat{\boldsymbol{x}} = (x_1, \ldots, x_I)$
- L is a lower objective bound for $\mathfrak{M}$ restricted to $x_i \in [l_i, u_i], \forall i \in [I]$
- so if $L > U$ then we can discard the node
- otherwise we try to tighten L by solving a polyhedral OA relaxation

Processing nodes

Suppose we are at a node $(\boldsymbol{l}, \boldsymbol{u}, L)$ of the branch and bound tree

- $\boldsymbol{l}, \boldsymbol{u}$ are the node's lower, upper variable bounds for $\hat{\boldsymbol{x}} = (x_1, \ldots, x_I)$
- L is a lower objective bound for $\mathfrak{M}$ restricted to $x_i \in [l_i, u_i], \forall i \in [I]$
- so if $L > U$ then we can discard the node
- otherwise we try to tighten L by solving a polyhedral OA relaxation

Given current $\mathcal{K}^*$ cut sets $(\mathcal{Z}_k)_{k \in [K]}$, we solve the LP $\mathfrak{P}\left((\mathcal{Z}_k)_{k \in [K]}, \boldsymbol{l}, \boldsymbol{u}\right)$

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle : & \qquad \left(\mathfrak{P}\left((\mathcal{Z}_k)_{k \in [K]}, \boldsymbol{l}, \boldsymbol{u}\right)\right) \\
\langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \in \mathbb{R}_+ & \qquad \forall \boldsymbol{z}_k \in \mathcal{Z}_k, k \in [K] \\
\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k & \qquad \forall k \in [M] \backslash [K] \\
x_i \in [l_i, u_i] & \qquad \forall i \in [I]
\end{aligned}
$$

Branch and bound algorithm

initialize global upper bound U to ∞
solve $\mathfrak{R}$ for optimal value $C_{\mathfrak{R}}$ and dual solution $(\bar{\boldsymbol{z}}_k)_{k \in [M]}$
initialize $\mathcal{K}^*$ cut sets $(\mathcal{Z}_k)_{k \in [K]}$ with relaxation cuts $(\bar{\boldsymbol{z}}_k)_{k \in [K]}$
initialize node list $\mathcal{N}$ with most relaxed node $(\boldsymbol{l}^0, \boldsymbol{u}^0, C_{\mathfrak{R}})$
while $\mathcal{N}$ contains nodes **do**
    remove a node $(\boldsymbol{l}, \boldsymbol{u}, L)$ from $\mathcal{N}$
    if node's lower bound $L \leq U$ **then**
        solve LP $\mathfrak{P}\left((\mathcal{Z}_k)_{k \in [K]}, \boldsymbol{l}, \boldsymbol{u}\right)$ and update U, $(\mathcal{Z}_k)_{k \in [K]}$, $\mathcal{N}$
    end if
end while

LP procedure at a node

if $\mathfrak{P}\left((\mathcal{Z}_k)_{k\in[K]}, \boldsymbol{l}, \boldsymbol{u}\right)$ is feasible **&** optimal value $C_{\mathfrak{P}} < U$ **then**
  let $\bar{\boldsymbol{x}}^\star$ be the integer variable subvector of an optimal solution
  if integrality $\bar{\boldsymbol{x}}^\star \in \mathbb{Z}^I$ is satisfied **then**
    solve $\mathfrak{R}(\bar{\boldsymbol{x}}^\star, \bar{\boldsymbol{x}}^\star)$ for an optimal dual solution or ray $(\bar{\boldsymbol{z}}_k)_{k\in[M]}$
    add $\mathcal{K}^*$ cuts $(\bar{\boldsymbol{z}}_k)_{k\in[K]}$ to $(\mathcal{Z}_k)_{k\in[K]}$
    if $\mathfrak{R}(\bar{\boldsymbol{x}}^\star, \bar{\boldsymbol{x}}^\star)$ is feasible **&** optimal value $C_{\mathfrak{R}}(\bar{\boldsymbol{x}}^\star, \bar{\boldsymbol{x}}^\star) < U$ **then**
      update U to new best feasible value $C_{\mathfrak{R}}(\bar{\boldsymbol{x}}^\star, \bar{\boldsymbol{x}}^\star)$
    end if
    add node $(\boldsymbol{l}, \boldsymbol{u}, C_{\mathfrak{P}})$ to $\mathcal{N}$ for reprocessing
  else
    choose a fractional variable $i : x_i^\star \notin \mathbb{Z}$ to branch on
    add left branch node $\left(\boldsymbol{l}, \left(u_1, \ldots, \lfloor x_i^\star \rfloor, \ldots, u_I\right), C_{\mathfrak{P}}\right)$ to $\mathcal{N}$
    add right branch node $\left(\left(l_1, \ldots, \lceil x_i^\star \rceil, \ldots, l_I\right), \boldsymbol{u}, C_{\mathfrak{P}}\right)$ to $\mathcal{N}$
  end if
end if

A continuous subproblem

Consider restricting the (relaxed) integer variables of $\mathfrak{R}$ to a box $(\boldsymbol{l}, \boldsymbol{u})$

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & \langle \boldsymbol{c}, \boldsymbol{x} \rangle : && (\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})) \\
\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} &\in \mathcal{C}_k && \forall k \in [M] \\
x_i &\in [l_i, u_i] && \forall i \in [I] \\
\boldsymbol{x} &\in \mathbb{R}^N
\end{aligned}
$$

A continuous subproblem

Consider restricting the (relaxed) integer variables of $\mathfrak{R}$ to a box $(\boldsymbol{l}, \boldsymbol{u})$

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \langle \boldsymbol{c}, \boldsymbol{x} \rangle : & & (\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})) \\ \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} &\in \mathcal{C}_k & \forall k \in [M] & \\ x_i &\in [l_i, u_i] & \forall i \in [I] & \\ \boldsymbol{x} &\in \mathbb{R}^N & & \end{aligned}$$

After encoding the box constraints conically, the conic dual is

$$\begin{aligned} \max_{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K, \boldsymbol{\alpha}, \boldsymbol{\beta}} \quad & \sum_{i \in [I]} (l_i \alpha_i - u_i \beta_i) - \sum_{k \in [M]} \langle \boldsymbol{b}_k, \boldsymbol{z}_k \rangle : & & (\mathfrak{R}^*(\boldsymbol{l}, \boldsymbol{u})) \\ \boldsymbol{c} + \sum_{i \in [I]} (\beta_i - \alpha_i) \boldsymbol{e}(i) + \sum_{k \in [M]} \boldsymbol{A}_k^T \boldsymbol{z}_k &\in \{0\}^N & & \\ \boldsymbol{z}_k &\in \mathcal{C}_k^* & \forall k \in [M] & \\ \boldsymbol{\alpha}, \boldsymbol{\beta} &\in \mathbb{R}_+^I & & \end{aligned}$$

Subproblem $\mathcal{K}^*$ cuts: feasible primal case

Assume $\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})$ is feasible and bounded, and strong duality holds, thus we have an optimal primal-dual solution $(\boldsymbol{x}^\star, \boldsymbol{z}_1^\star, \ldots, \boldsymbol{z}_K^\star, \boldsymbol{\alpha}^\star, \boldsymbol{\beta}^\star)$

Subproblem $\mathcal{K}^*$ cuts: feasible primal case

Assume $\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})$ is feasible and bounded, and strong duality holds, thus we have an optimal primal-dual solution $(\boldsymbol{x}^\star, \boldsymbol{z}_1^\star, \ldots, \boldsymbol{z}_K^\star, \boldsymbol{\alpha}^\star, \boldsymbol{\beta}^\star)$

From the dual solution subvector $(\bar{\boldsymbol{z}}_k)_{k \in [M]}$, we derive $\mathcal{K}^*$ cuts

Subproblem $\mathcal{K}^*$ cuts: feasible primal case

Assume $\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})$ is feasible and bounded, and strong duality holds, thus we have an optimal primal-dual solution $(\boldsymbol{x}^\star, \boldsymbol{z}_1^\star, \ldots, \boldsymbol{z}_K^\star, \boldsymbol{\alpha}^\star, \boldsymbol{\beta}^\star)$

From the dual solution subvector $(\bar{\boldsymbol{z}}_k)_{k \in [M]}$, we derive $\mathcal{K}^*$ cuts

$$\begin{aligned} \langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \bar{\boldsymbol{z}}_k \rangle &\geq 0 && \forall k \in [K] \\ \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} &\in \mathcal{C}_k && \forall k \in [M] \backslash [K] \\ x_i &\in [l_i, u_i] && \forall i \in [I] \end{aligned}$$

Any $\boldsymbol{x}$ satisfying these linear constraints satisfies an objective bound

$$\langle \boldsymbol{c}, \boldsymbol{x} \rangle \geq \langle \boldsymbol{c}, \boldsymbol{x}^\star \rangle$$

Subproblem $\mathcal{K}^*$ cuts: infeasible primal case

Assume now $\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})$ is infeasible, so we have a certificate of infeasibility i.e. a ray $\big((\boldsymbol{z}_k)_{k \in [M]}, \boldsymbol{\alpha}, \boldsymbol{\beta}\big)$ of $\mathfrak{R}^*(\boldsymbol{l}, \boldsymbol{u})$ satisfying

$$\sum_{i \in [I]} (\beta_i - \alpha_i) \boldsymbol{e}(i) + \sum_{k \in [K]} \boldsymbol{A}_k^T \boldsymbol{z}_k \in \{0\}^N$$

$$\sum_{i \in [I]} (u_i \beta_i - l_i \alpha_i) + \sum_{k \in [M]} \langle \boldsymbol{b}_k, \boldsymbol{z}_k \rangle < 0$$

Subproblem $\mathcal{K}^*$ cuts: infeasible primal case

Assume now $\mathfrak{R}(\boldsymbol{l}, \boldsymbol{u})$ is infeasible, so we have a certificate of infeasibility i.e. a ray $\big((\boldsymbol{z}_k)_{k\in[M]}, \boldsymbol{\alpha}, \boldsymbol{\beta}\big)$ of $\mathfrak{R}^*(\boldsymbol{l}, \boldsymbol{u})$ satisfying

$$\sum_{i\in[I]}(\beta_i - \alpha_i)\boldsymbol{e}(i) + \sum_{k\in[K]} \boldsymbol{A}_k^T \boldsymbol{z}_k \in \{0\}^N$$

$$\sum_{i\in[I]}(u_i\beta_i - l_i\alpha_i) + \sum_{k\in[M]} \langle \boldsymbol{b}_k, \boldsymbol{z}_k \rangle < 0$$

From the ray subvector $(\bar{\boldsymbol{z}}_k)_{k\in[M]}$, we derive $\mathcal{K}^*$ cuts that exclude all solutions for the bounds $(\boldsymbol{l}, \boldsymbol{u})$

Subproblem $\mathcal{K}^*$ cuts: infeasible primal case

For all $\boldsymbol{x}$ satisfying

$$\begin{aligned} \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} &\in \mathcal{C}_k & \forall k &\in [M] \backslash [K] \\ x_i &\in [l_i, u_i] & \forall i &\in [I] \end{aligned}$$

there exists a $k \in [K]$ such that $\langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle < 0$

Subproblem $\mathcal{K}^*$ cuts: infeasible primal case

For all $\boldsymbol{x}$ satisfying

$$\begin{aligned} \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} &\in \mathcal{C}_k && \forall k \in [M] \backslash [K] \\ x_i &\in [l_i, u_i] && \forall i \in [I] \end{aligned}$$

there exists a $k \in [K]$ such that $\langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle < 0$

$$\begin{aligned} &\sum_{k \in [K]} \langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle \\ &\leq \sum_{k \in [M]} \langle \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}, \boldsymbol{z}_k \rangle + \sum_{i \in [I]} (-l_i + x_i) \alpha_i + \sum_{i \in [I]} (u_i - x_i) \beta_i \\ &= \left\langle \boldsymbol{x}, \sum_{i \in [I]} (\alpha_i - \beta_i) \boldsymbol{e}(i) - \sum_{k \in [M]} \boldsymbol{A}_k^T \boldsymbol{z}_k \right\rangle + \sum_{k \in [M]} \langle \boldsymbol{b}_k, \boldsymbol{z}_k \rangle + \sum_{i \in [I]} (u_i \beta_i - l_i \alpha_i) \\ &= \sum_{k \in [M]} \langle \boldsymbol{b}_k, \boldsymbol{z}_k \rangle + \sum_{i \in [I]} (u_i \beta_i - l_i \alpha_i) < 0 \end{aligned}$$

Comparing subproblem and separation cuts

Termination statuses and shifted geomean of solve time and iteration count (for iterative algorithm only) on 120 MISOCPs, using **Pajarito** with **CPLEX** and **MOSEK**

options		termination status counts				conv only stats	
alg	cuts	conv	wrong	not conv	limit	time(s)	iterations
iter	sep	96	1	0	23	55.23	6.76
iter	subp	95	1	3	21	39.59	4.07
MSD	sep	95	1	0	24	20.86	–
MSD	subp	100	0	1	19	17.56	–

Comparing subproblem and separation cuts

Termination statuses and shifted geomean of solve time and iteration count (for iterative algorithm only) on 120 MISOCPs, using **Pajarito** with **CPLEX** and **MOSEK**

options		termination status counts				conv only stats	
alg	cuts	conv	wrong	not conv	limit	time(s)	iterations
iter	sep	96	1	0	23	55.23	6.76
iter	subp	95	1	3	21	39.59	4.07
MSD	sep	95	1	0	24	20.86	–
MSD	subp	100	0	1	19	17.56	–

Subproblem cuts should be used always, and separation cuts should be invoked when necessary for convergence

Comparisons with specialized MISOCP solvers

Termination statuses and shifted geometric mean of solve time on 120 MISOCPs, for **SCIP** and **CPLEX** MISOCP solvers, and default MSD and iterative **Pajarito** solvers using **CPLEX** and **MOSEK**

	termination status counts				
solver	conv	wrong	not conv	limit	time(s)
SCIP	78	1	0	41	43.36
CPLEX	96	3	5	16	14.30
Paj-iter	96	1	0	23	38.70
Paj-MSD	101	0	0	19	18.12

Comparisons with specialized MISOCP solvers

Termination statuses and shifted geometric mean of solve time on 120 MISOCPs, for **SCIP** and **CPLEX** MISOCP solvers, and default MSD and iterative **Pajarito** solvers using **CPLEX** and **MOSEK**

	termination status counts				
solver	conv	wrong	not conv	limit	time(s)
SCIP	78	1	0	41	43.36
CPLEX	96	3	5	16	14.30
Paj-iter	96	1	0	23	38.70
Paj-MSD	101	0	0	19	18.12

Pajarito's MSD algorithm solves more instances in the time limit and has no incorrect solutions

Open-source solver comparisons for MISOCP

Termination statuses and shifted geomean of solve time on 120 MISOCPs for **BONMIN** [BBC+08] with **Cbc** and **IPOPT**, and **Pajarito** using **Cbc** or **GLPK** and **ECOS** (iterative algorithm with default options)

	termination status counts				
solver	conv	wrong	not conv	limit	time(s)
BONMIN-BB	37	27	10	46	82.95
BONMIN-OA	30	8	29	53	72.12
BONMIN-OA-D	35	8	29	48	64.25
Paj-CBC-ECOS	81	8	0	31	51.48
Paj-GLPK-ECOS	68	0	2	50	42.75

Testing with portfolio optimization

Using covariance estimates from real data, we generate cardinality constrained multi-portfolio problems with convex risk constraints

Testing with portfolio optimization

Using covariance estimates from real data, we generate cardinality constrained multi-portfolio problems with convex risk constraints

- $\mathcal{L}$ ℓ_2 norm - standard in Markowitz model
- $\mathcal{P}$ robust ℓ_2 norm [BTEGN09]
- $\mathcal{E}$ entropic ball [BTEGN09]

Testing with portfolio optimization

Using covariance estimates from real data, we generate cardinality constrained multi-portfolio problems with convex risk constraints

$\mathcal{L}$ ℓ_2 norm - standard in Markowitz model

$\mathcal{P}$ robust ℓ_2 norm [BTEGN09]

$\mathcal{E}$ entropic ball [BTEGN09]

On instances with 20 portfolios and up to 100 stocks per portfolio, running **Pajarito**'s MSD algorithm using default options and **CPLEX**

- with ℓ_2 norm, using **MOSEK**, several minutes
- with ℓ_2 norm and entropic ball, using **ECOS**, 5-10 minutes
- with ℓ_2 norm and robust norm, using **MOSEK**, 20-30 minutes
- with all three risk constraints, using **SCS**, hours

Testing with portfolio optimization

Using covariance estimates from real data, we generate cardinality constrained multi-portfolio problems with convex risk constraints

$\mathcal{L}$ ℓ_2 norm - standard in Markowitz model

$\mathcal{P}$ robust ℓ_2 norm [BTEGN09]

$\mathcal{E}$ entropic ball [BTEGN09]

On instances with 20 portfolios and up to 100 stocks per portfolio, running **Pajarito**'s MSD algorithm using default options and **CPLEX**

- with ℓ_2 norm, using **MOSEK**, several minutes
- with ℓ_2 norm and entropic ball, using **ECOS**, 5-10 minutes
- with ℓ_2 norm and robust norm, using **MOSEK**, 20-30 minutes
- with all three risk constraints, using **SCS**, hours

Problems with $\mathcal{P}$ scale poorly - no disaggregated extended formulation