Modeling Power of Mixed Integer Convex Optimization Problems And Their Effective Solution with Julia and JuMP

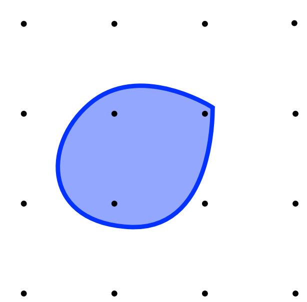
Juan Pablo Vielma

Massachusetts Institute of Technology

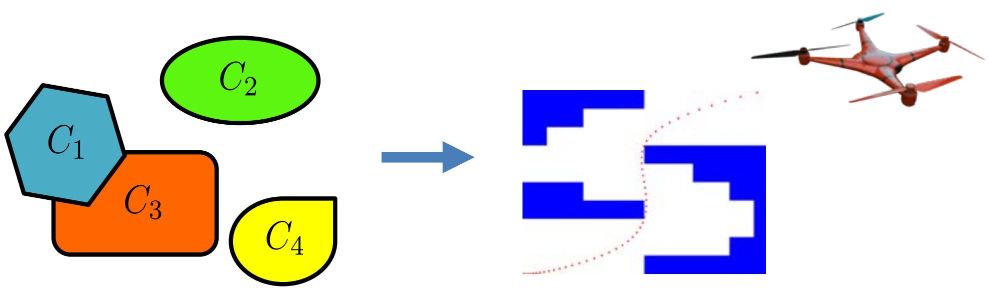
Mechanical and Industrial Engineering Department, Northeastern University,
Boston, MA, October, 2018.

Mixed Integer Convex Optimization (MICONV)

$$
\begin{aligned}
\min \quad & f(x) \\
s.t. \quad & \\
& x \in C \\
& x_i \in \mathbb{Z} \quad i \in I
\end{aligned}
$$



convex f and C.



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

Overview

- What can we model with MICONV
- How can we solve MICONVs
  - How can we access solvers
  - How solvers work

What can we model with MICONV?

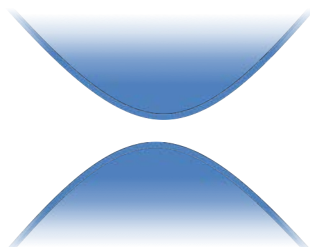
Joint work with Miles Lubin and Ilias Zadik

What Can MICONV Model?

- Optimal discrete experimental design
- Obstacle avoidance and trajectory planning in optimal control
- Portfolio optimization with nonlinear risk measures and combinatorial constraints
- …

No, Really. What Can MICONV Model?

Two sheet hyperbola?



$$\{x \in \mathbb{R}^2 \ : \ 1 + x_1^2 \leq x_2^2\}$$

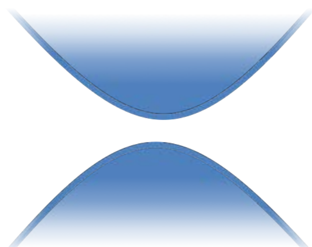
Spherical shell?

$$\{x \in \mathbb{R}^2 \ : \ 1 \leq \|x\| \leq 2\}$$

- Integer points in parabola $\{(x, x^2) : x \in \mathbb{Z}\}$?
- The set of $n \times n$ matrices with $\mathrm{rank} \leq k$?
- Set of prime numbers?

No, Really. What Can MICONV Model?

Two sheet hyperbola?

✓ 

Simple MICONV Formulation

$$\{x \in \mathbb{R}^2 \; : \; 1 + x_1^2 \leq x_2^2\}$$

Spherical shell?

$$\{x \in \mathbb{R}^2 \; : \; 1 \leq \|x\| \leq 2\}$$

- Integer points in parabola $\{(x, x^2) : x \in \mathbb{Z}\}$?
- The set of $n \times n$ matrices with rank $\leq k$?
- Set of prime numbers?

A Simple Obstruction for MICONV Formulations

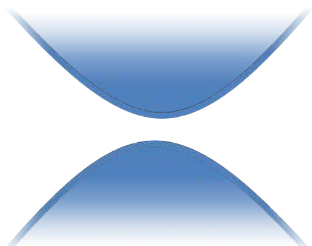
- S cannot have a MICONV formulation if there exists:
  - There exist infinite $R \subseteq S \quad \text{s.t.}$

$$\frac{u+v}{2} \notin S \quad \forall u, v \in R,\ u \neq v$$

✗ **Spherical shell** $\{x \in \mathbb{R}^2 \,:\, 1 \leq \|x\| \leq 2\}$

No, Really. What Can MICONV Model?

Two sheet hyperbola?

✓ 

$$\{x \in \mathbb{R}^2 \ : \ 1 + x_1^2 \leq x_2^2\}$$

Spherical shell?

✗ 

$$\{x \in \mathbb{R}^2 \ : \ 1 \leq \|x\| \leq 2\}$$

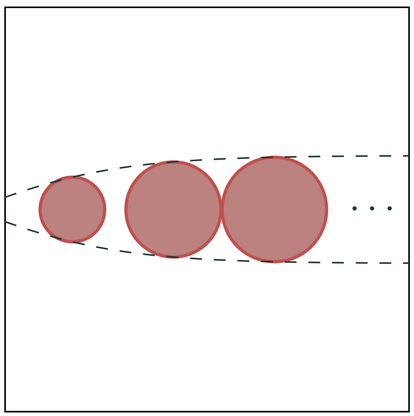
✗ Integer points in parabola $\{(x, x^2) : x \in \mathbb{Z}\}$?

✗ The set of $n \times n$ matrices with rank $\leq k$?

✗ Set of prime numbers?

Does have non-convex polynomial MIP formulation

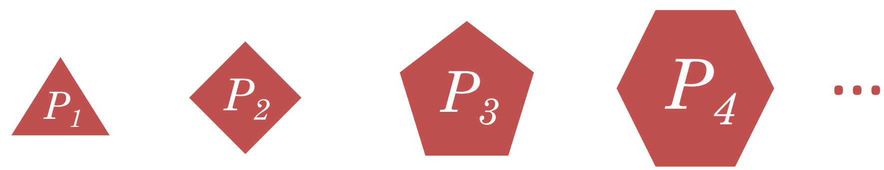
MICONV = **Structured** ***Countably Infinite*** Unions of Convex Sets



$$\sqrt{(x_1 - 2z)^2 + x_2^2} \le 1 - 1/z,$$
$$z \ge 1, \quad z \in \mathbb{Z}$$

Unbounded Integer Variables

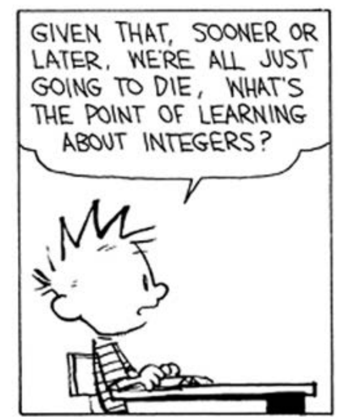
- Can be "strange" unions, e.g. :
  - Infinite number of shapes



- Can be REALLY strange:
  - Dense discrete set

$$\left\{ \sqrt{2}x - \left\lfloor \sqrt{2}x \right\rfloor \; : \; x \in \mathbb{N} \right\} \subseteq [0,1]$$

$$\|(z_1, z_1)\|_2 \le z_2 + 1, \quad \|(z_2, z_2)\|_2 \le 2z_1, \quad x_1 = y_1 - z_2,$$
$$\|(z_1, z_1)\|_2 \le y_1, \quad \|(y_1, y_1)\|_2 \le 2z_1, \quad z \in \mathbb{Z}^2$$



MICONV with

julia & JuMP

50+ Years of MIP = Significant Solver Speedups

- Algorithmic Improvements (**Machine Independent**):
  - CPLEX → ILOG → IBM
    - v1.2 (1991) – v11 (2007): **29,000 x** speedup
  - GUROBI OPTIMIZATION
    - v1 (2009) – v6.5 (2015): **48.7 x** speedup

  ≈ **1.9 x / year**

- Also convex nonlinear:
  - GUROBI OPTIMIZATION
    - v6.0 (2014) – v6.5 (2015) quadratic: **4.43 x**

      (V., Dunning, Huchette, Lubin, 2015)

State of MIP Solvers

- Mature: Linear and Quadratic (Conic Quadratic/SOCP)
  - Commercial:
  - "Open Source"

CBC GLPK

- Emerging: Convex Nonlinear (e.g. SDP)
  - Open-Source + Commercial linear MIP Solver > Commercial

Accessing MIP Solvers = Modelling Languages

- User-friendly algebraic modelling languages (AML):

Standalone and Fast | Based on General Language and Versatile

- Fast and Versatile, but complicated (and possibly proprietary)
  - Low-level C/C++ solver or Coin-OR interphases & frameworks
- 21st Century AMLs:

21st Century Programming/Modelling Languages

julia

- Open-source and free!
- Developed at MIT
- "Floats like python/matlab, stings like C/Fortran"
- Easy to use and wide library ecosystem (specialized and frontend)
- Only language besides C/C++/Fortran to scale to 1 Petaflop!

JUMP

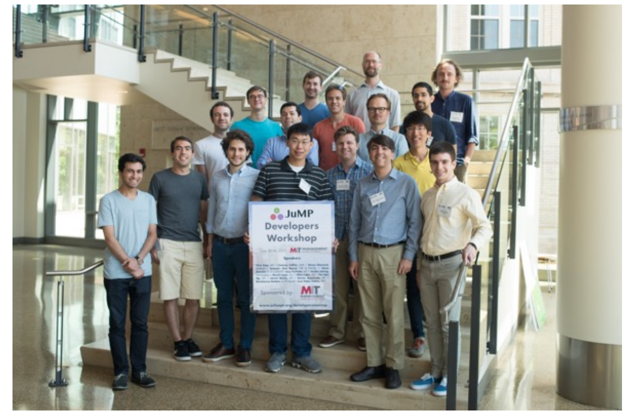
- Open-source and free!
- Modelling language, interface and software ecosystem for optimization
- Easy to use and advanced
- Integrated into Julia
- Created at MIT and beyond…

Large Software Stack and Vibrant Community

Large Software Stack and Vibrant Community

2016

Iain Dunning, Miles Lubin and Joey Huchette



JUMP Not Just a Modeling Language / Interphase

- JuMP domain specific language (DSL)
- Solve abstraction layers:
  - MathProgBase / MathOptInterface
- Solver interfaces
- Solvers: Pajarito.jl, Pavito.jl
- Extensions: SumOfSquares.jl, PolyJuMP.jl
- Now a NumFOCUS Sponsored project!

Modeling Tool	Linear / Quadratic	Convex: Conic	Convex: Smooth	Nonconvex	Integer
JuMP	✔	✔	✔	✔	✔
Convex.jl	✔	✔			✔
Solver					
CDD (.jl)	✔				
Clp (.jl)	✔				
OSQP (.jl)	✔				
Cbc (.jl)	✔				✔
GLPK (.jl)	✔				✔cb
CSDP (.jl)	✔	✔			
ECOS (.jl)	✔	✔			
SCS (.jl)	✔	✔			
SDPA (.jl)	✔	✔			
CPLEX (.jl)	✔	✔			✔cb
Gurobi (.jl)	✔	✔			✔cb
FICO Xpress (.jl)	✔	✔			✔
Mosek (.jl)	✔	✔	✔		✔
Pajarito.jl	✔	✔	✔		✔
NLopt (.jl)			✔	✔	
Ipopt (.jl)	✔		✔	✔	
Bonmin (via AmplNLWriter.jl)	✔		✔	✔	✔
Couenne (via AmplNLWriter.jl)	✔		✔	✔	✔
Artelys Knitro (.jl)	✔		✔	✔	✔
SCIP (.jl)	✔	✔	✔	✔	✔cb

More julia Packages

(Based on https://www.flickr.com/photos/153311384@N03/)

BioJulia
EcoJulia
JuliaArchive
JuliaArrays
JuliaAstro
JuliaDB
JuliaApproximation

JuliaAstrodynamics
JuliaAudio
JuliaBerry
JuliaCI
JuliaCN
JuliaCloud
JuliaCollections

JuliaDSP
JuliaData
JuliaDiff
JuliaDiffEq
JuliaDocs
JuliaDynamics
JuliaEditorSupport

JuliaGPU
JuliaGeo
JuliaGeometry
JuliaGraphics
JuliaGraphs
JuliaIO
JuliaImages

And More julia Packages

(Based on https://www.flickr.com/photos/153311384@N03/)

JuliaFEM JuliaFinMetrix JuliaML JuliaMath JuliaNeuro SeismicJulia JuliaPOMDP

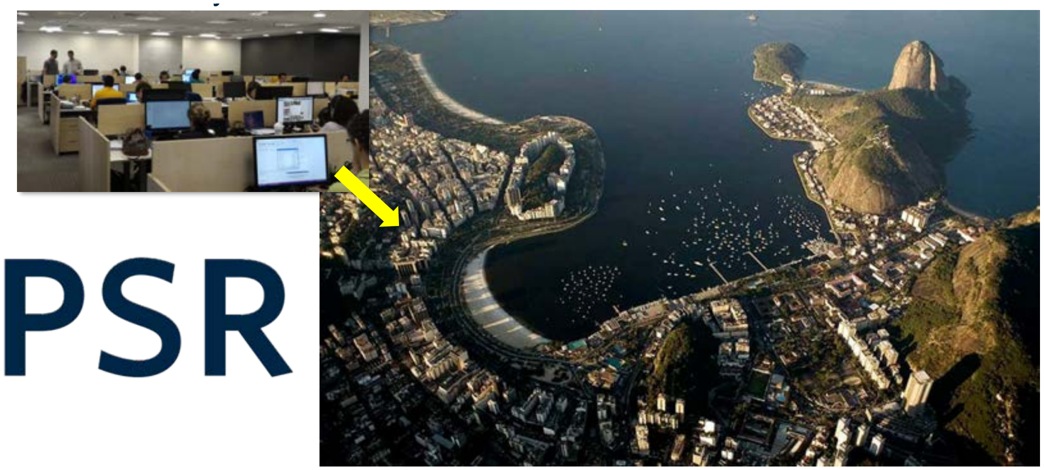
JuliaPlots JuliaPraxis JuliaPy JuliaQuant JuliaQuantum JuliaSmoothOptimizers JuliaSparse

JuliaGL JuliaWeb JuliaInterop JuliaInv JuliaPackaging JuliaParallel JuliaPhysics

JuliaStatistics JuliaTime

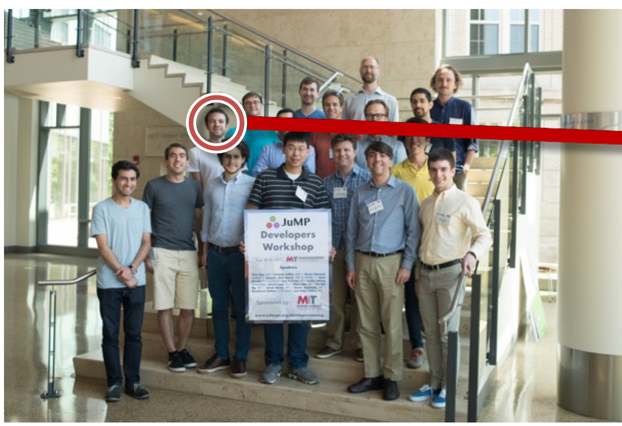
Julia and JuMP In Production Environments

PSR



Peruvian Energy Ministry

The Nature Conservancy



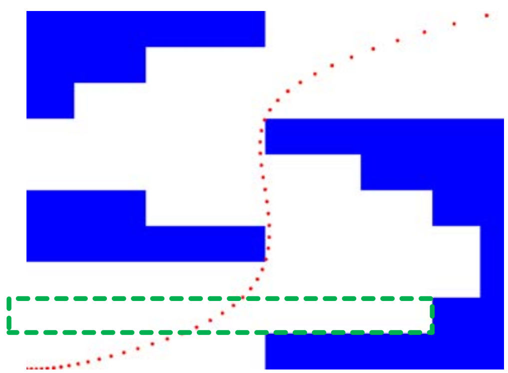
Joaquim Dias Garcia

BID
Banco Interamericano de Desarrollo

Northwest Power and Conservation Council

An MICONV Example

- Problem: Steer a quadcopter through obstacles [Deits/Tedrake:2015]
  - ~2 week of work by Joey Huchette for SIOPT '17
- Position described by polynomials:



$$(p^x(t), p^y(t))_{t\in[0,1]}$$

$$\{p_i : [T_i, T_{i+1}] \to \mathbb{R}^2\}_{i=1}^N$$

$$0 = T_1 < T_2 < \ldots < T_N = 1$$

- Solution approach:
  – split domain into "safe polyhedrons" + discretize time into intervals

Disjunctive *Polynomial* Optimization Formulation

Variables = Polynomials : $\{p_i : [T_i, T_{i+1}] \rightarrow \mathbb{R}^2\}_{i=1}^N$

$$\min_{p} \sum_{i=1}^{N} ||p_i'''(t)||^2$$

s.t.

$$p_1(0) = X_0,\ p'(0) = X_0',\ p''(0) = X_0''$$

$$p_N(1) = X_f,\ p_N'(1) = X_f',\ p_N''(1) = X_f''$$

Initial/Terminal Conditions

$$p_i(T_{i+1}) = p_{i+1}(T_{i+1}) \quad \forall i \in \{1, \ldots, N-1\}$$

$$p_i'(T_{i+1}) = p_{i+1}'(T_{i+1}) \quad \forall i \in \{1, \ldots, N-1\}$$

$$p_i''(T_{i+1}) = p_{i+1}''(T_{i+1}) \quad \forall i \in \{1, \ldots, N-1\}$$

Interstitial Smoothing Conditions

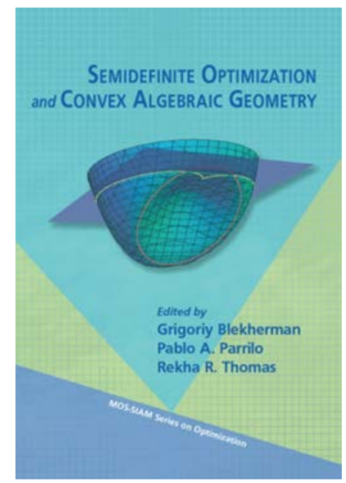
$$\bigvee_{r=1}^{R} [A^r p_i(t) \leq b^r] \text{ for } t \in [T_i, T_{i+1}] \quad \forall i \in \{1, \ldots, N-1\}$$

Avoid Collision = Remain in Safe Regions

… Mixed Integer Semidefinite Programming

MIP

+

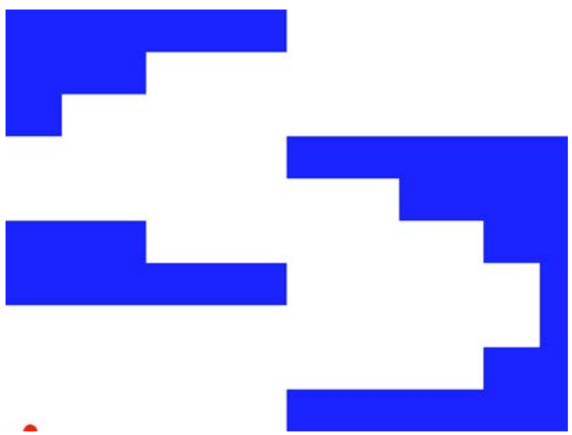


+

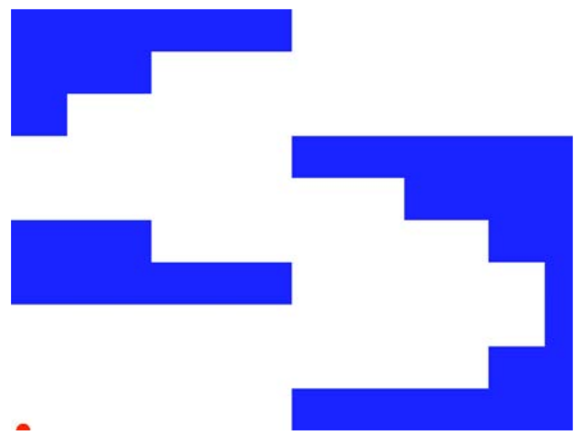

```
model = SOSModel(solver=PajaritoSolver())

@polyvar(t)
Z = monomials([t], 0:r)

@variable(model, H[1:N,boxes], Bin)

p = Dict()
for j in 1:N
   @constraint(model, sum(H[j,box] for box in boxes) == 1)
   p[(:x,j)] = @polyvariable(model, _, Z)
   p[(:y,j)] = @polyvariable(model, _, Z)
   for box in boxes
       xl, xu, yl, yu = box.xl, box.xu, box.yl, box.yu
       @polyconstraint(model, p[(:x,j)] >= Mxl + (xl-Mxl)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
       @polyconstraint(model, p[(:x,j)] <= Mxu + (xu-Mxu)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
       @polyconstraint(model, p[(:y,j)] >= Myl + (yl-Myl)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
       @polyconstraint(model, p[(:y,j)] <= Myu + (yu-Myu)*H[j,box], domain = (t >= T[j] && t <= T[j+1]))
   end
end

for ax in (:x,:y)
   @constraint(model,                p[(ax,1)       ]([0], [t]) == X₀[ax])
   @constraint(model, differentiate(p[(ax,1)], t   )([0], [t]) == X₀′[ax])
   @constraint(model, differentiate(p[(ax,1)], t, 2)([0], [t]) == X₀′′[ax])
   for j in 1:N-1
       @constraint(model,                p[(ax,j)     ]([T[j+1]],[t]) ==                p[(ax,j+1)     ]([T[j+1]],[t]))
       @constraint(model, differentiate(p[(ax,j)],t  )([T[j+1]],[t]) == differentiate(p[(ax,j+1)],t  )([T[j+1]],[t]))
       @constraint(model, differentiate(p[(ax,j)],t,2)([T[j+1]],[t]) == differentiate(p[(ax,j+1)],t,2)([T[j+1]],[t]))
   end
   @constraint(model,                p[(ax,N)       ]([1], [t]) == X₁[ax])
   @constraint(model, differentiate(p[(ax,N)], t   )([1], [t]) == X₁′[ax])
   @constraint(model, differentiate(p[(ax,N)], t, 2)([1], [t]) == X₁′′[ax])
end

@variable(model, γ[keys(p)] ≥ 0)
for (key,val) in p
   @constraint(model, γ[key] ≥ norm(differentiate(val, t, 3)))
end
@objective(model, Min, sum(γ))
```

```
function eval_poly(r)
    for i in 1:N
        if T[i] <= r <= T[i+1]
            return PP[(:x,i)]([r], [t]), PP[(:y,i)]([r], [t])
            break
        end
    end
end
```

Results for 9 Regions and 8 time steps

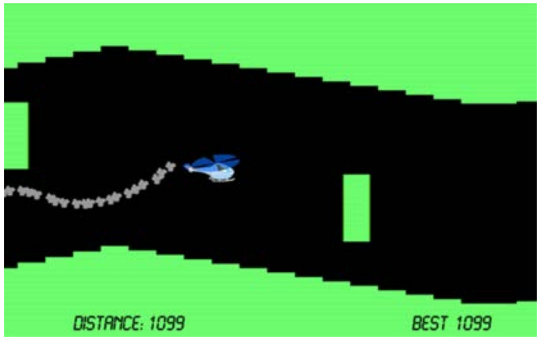


First Feasible Solution:
58 seconds

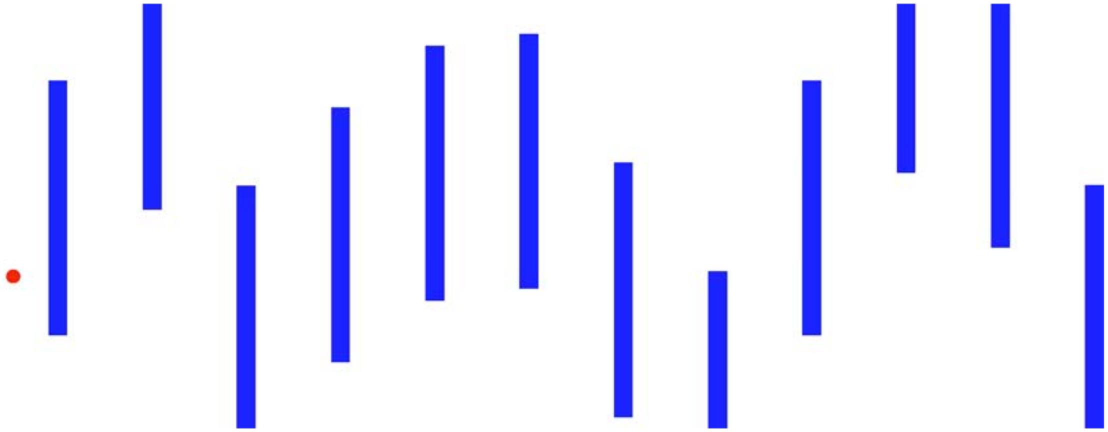


Optimal Solution:
651 seconds

Helicopter Game / Flappy Bird



- 60 horizontal segments, obstacle every 5 = 80 sec. to opt.



How can we solve MICONV?

Joint work with Russell Bent, Chris Coey, Iain Dunning, Joey Huchette, Lea Kapelevich, Miles Lubin, Emre Yamangil, …

MICONV B&B Algorithms

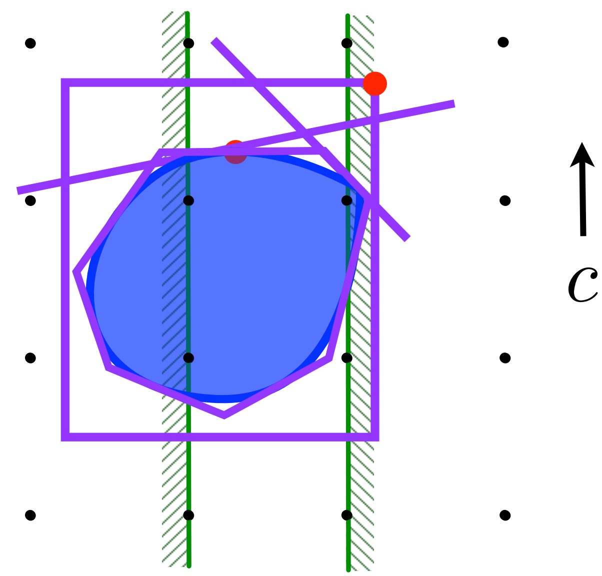
- NLP (QCP) Based B&B
- (Dynamic) LP Based B&B
  - Few cuts = high speed.
  - Possible slow convergence.
- Lifted LP B&B
  - Extended or Lifted relaxation.
  - Static relaxation
    - Mimic NLP B&B.
  - Dynamic relaxation
    - Standard LP B&B

$$\max \quad \sum_{i=1}^{n} c_i x_i$$

$$s.t. \quad Ax + Dz \leq b,$$

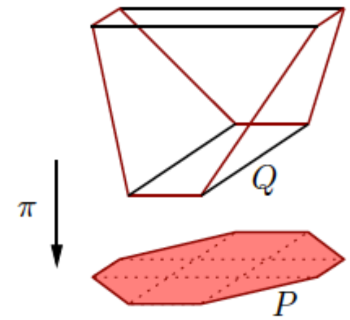
$$g_i(x) \leq 0,\ i \in I, \quad x \in \mathbb{Z}^n$$

$$x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}$$



Lifted or Extended Approximations

- Projection = multiply constraints.
- V., Ahmed. and Nemhauser 2008:
  - Extremely accurate, but static and complex approximation by Ben-Tal and Nemirovski
- V., Dunning, Huchette and Lubin 2015: Simple, dynamic and good approximation:
  - First talks: May ‘14 (SIOPT), Dec ‘14 IBM
  - Paper in arxive, May ’15
  - Adopted in CPLEX v12.6.2, Jun 15’
  - Gurobi (Oct ‘15), Xpress (May ‘16), SCIP (Mar’ 17)



$$y_i^2 \leq z_i \cdot y_0 \quad \forall i \in [n]$$

$$\sum_{i=1}^{n} z_i \leq y_0$$

$$\|y\|_2 \leq y_0$$

Image from Lipton and Regan, https://rjlipton.wordpress.com

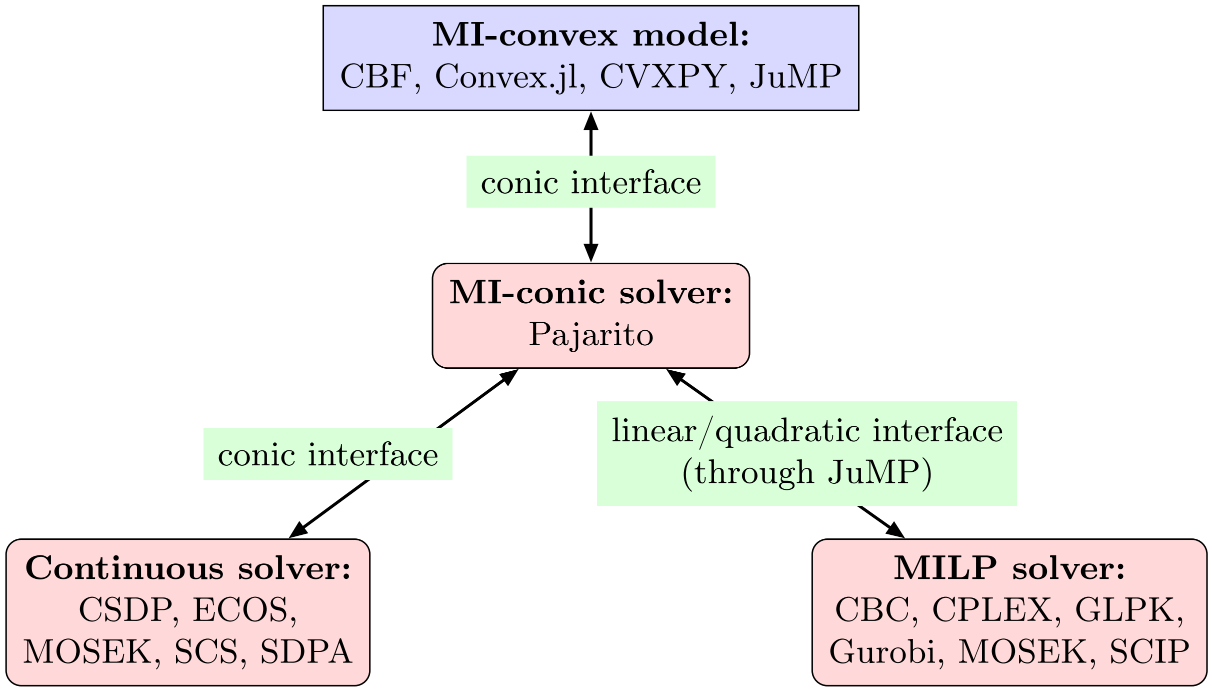
Not MICONV but, Mixed Integer Conic Programming (MICP)

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

- $\mathcal{C}_k$ closed convex cones
  - Linear, SOCP, rotated SOCP, SDP
  - Exponential cone, power cone, …
  - Spectral norm, relative entropy, sum-of-squares, …

- Fast and stable interior point algorithms for continuous relaxation
- Geometrically intuitive conic duality guides linear inequality selection
- Conic formulation techniques usually lead to extended formulations
  - MINLPLIB2 instances unsolved since 2001 solved by re-write to MISOCP

Pajarito: A Julia-based MICP Solver



- Early version solved gams01, tls5 and tls6 (MINLPLIB2)

Performance for MISOCP Instances (120 from CBLIB)

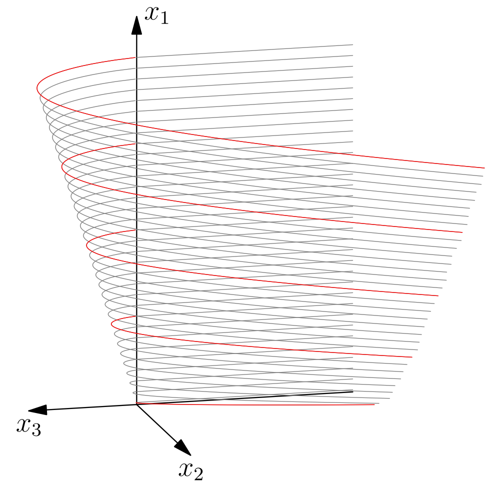
	solver	statuses				
		ok	limit	error	wrong	time (s)
open source	Bonmin-BB	34	44	11	31	463
	Bonmin-OA	25	53	29	13	726
	Bonmin-OA-D	30	48	29	13	610
	Pajarito-GLPK-ECOS	56	60	3	1	377
	Pajarito-CBC-ECOS	78	30	3	9	163
restricted	SCIP (4.0.0)	74	35	8	3	160
	CPLEX (12.7.0)	90	16	5	9	50
	Pajarito-CPLEX-MOSEK (9.0.0.29-alpha)	97	20	2	1	56

Also Exponential Cone + LP / SOCP / SDP

$$x_1 \geq x_2 e^{x_3/x_2}, \;\; x_1, x_2 > 0.$$

or

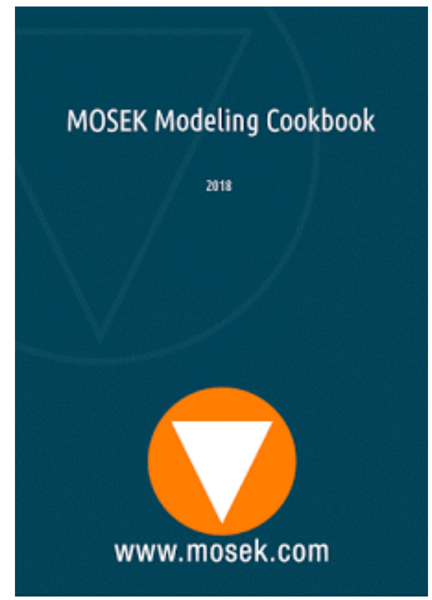
$$x_3 \leq x_2 \log(x_1/x_2), \;\; x_1, x_2 > 0.$$



- Discrete experimental design

$$x \longrightarrow \log\det\left(\sum_{i=1}^{n} x_i \mathbf{u}_i \mathbf{u}_i^T\right)$$

- Portfolio Optimization with entropic risk constraints
- All 333 MICONVs from MINLPLIB2
- Pajarito with SCS or Mosek (version 7.5.2)

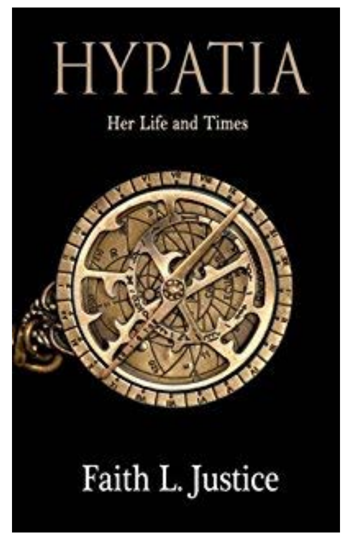


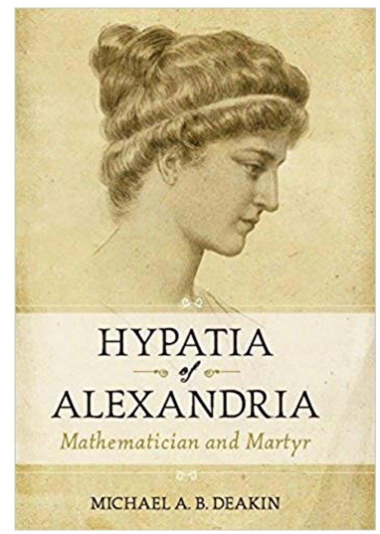
https://themosekblog.blogspot.com/2018/05/new-modeling-cookbook.html

Hypatia: **Pure Julia-based** IPM Beyond “Standard” Cones

- Extension of methods in CVXOPT and Alfonso
  - A customizable homogeneous interior-point solver for nonsymmetric convex
  - Skajaa and Ye ‘15, Papp and Yıldız ‘17, Andersen, Dahl, and Vandenberghe ‘04-18
- Cones: LP, dual Sum-of-Squares, SOCP, RSOCP, 3-dim exponential cone, PSD, L_∞, n-dim power cone (using AD), spectral norm, …
- Potential:
  - flexible number types and linear algebra
  - BOB: bring your own barrier (in ~50 lines of code)
  - Alternative prediction steps (Runge–Kutta)

Chris Coey





Early Comparison with Alfonso for LP and SOS

First Hypatia commit : Jul 15

Category	test	iters	Matlab	`75cba5f` (Aug 5)	`c9f1eb5` (Aug 19)	`133b422` (Aug 23)
Linear Optimization	dense lp	65	5.8	4.1	2.03	1.25
Polynomial Envelope	envelope	30	0.085	0.043	0.020	x
Polynomial Minimization	butcher	32/30	0.63	0.41	0.357	0.136
	caprasse	31/30	1.38	1.87	1.80	0.530
	lotka-volt	31/30	0.47	0.38	0.37	0.104
	motzkin	41/42	0.35	0.24	x	0.054
	reac-diff	29/30	0.32	0.23	0.19	0.075
	robinson	29	0.34	0.23	0.17	0.034

- First Batch of Tests on CBLIB Instances (SDP/SOCP): Only 2 – 10K times slower than Mosek 8!

Summary

- MICONV can model many problems (but not all)
- How to solve MICONVs? Don't solve MICONVs, solve MICPs
- Easy access to optimization modeling and solvers with JuMP
- Advanced solver development with Julia
- Disclaimers:
  - Julia just reached version 1 (Yay!)
  - … JuMP is undergoing a major redesign
    - Try in Julia 1.0 through "] add JuMP#v0.19-alpha"