Mathematical Optimization using JuMP

Juan Pablo Vielma

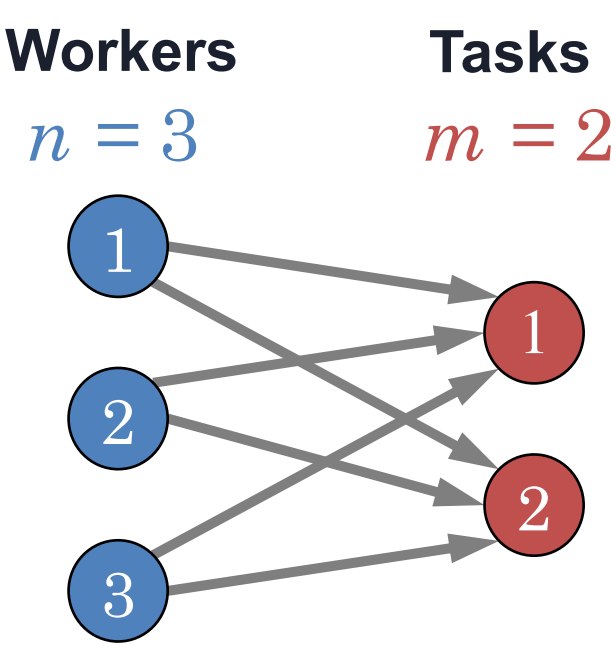
Massachusetts Institute of Technology

MIT 18.337/6.338 Modern Numerical Computing

Cambridge , MA, November, 2018.

Assignment Problem = Bipartite Matching

- Assign n workers to m tasks to complete all tasks
- At most one task per worker
- Minimize total time worked
  - Worker i takes $t_{i,j}$ hours to complete task j
- Variables:
  - $x_{i,j} = 1$ if worker i is assigned to task j and 0 o.w.



$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{m} t_{i,j} x_{i,j}$$

$s.t.$

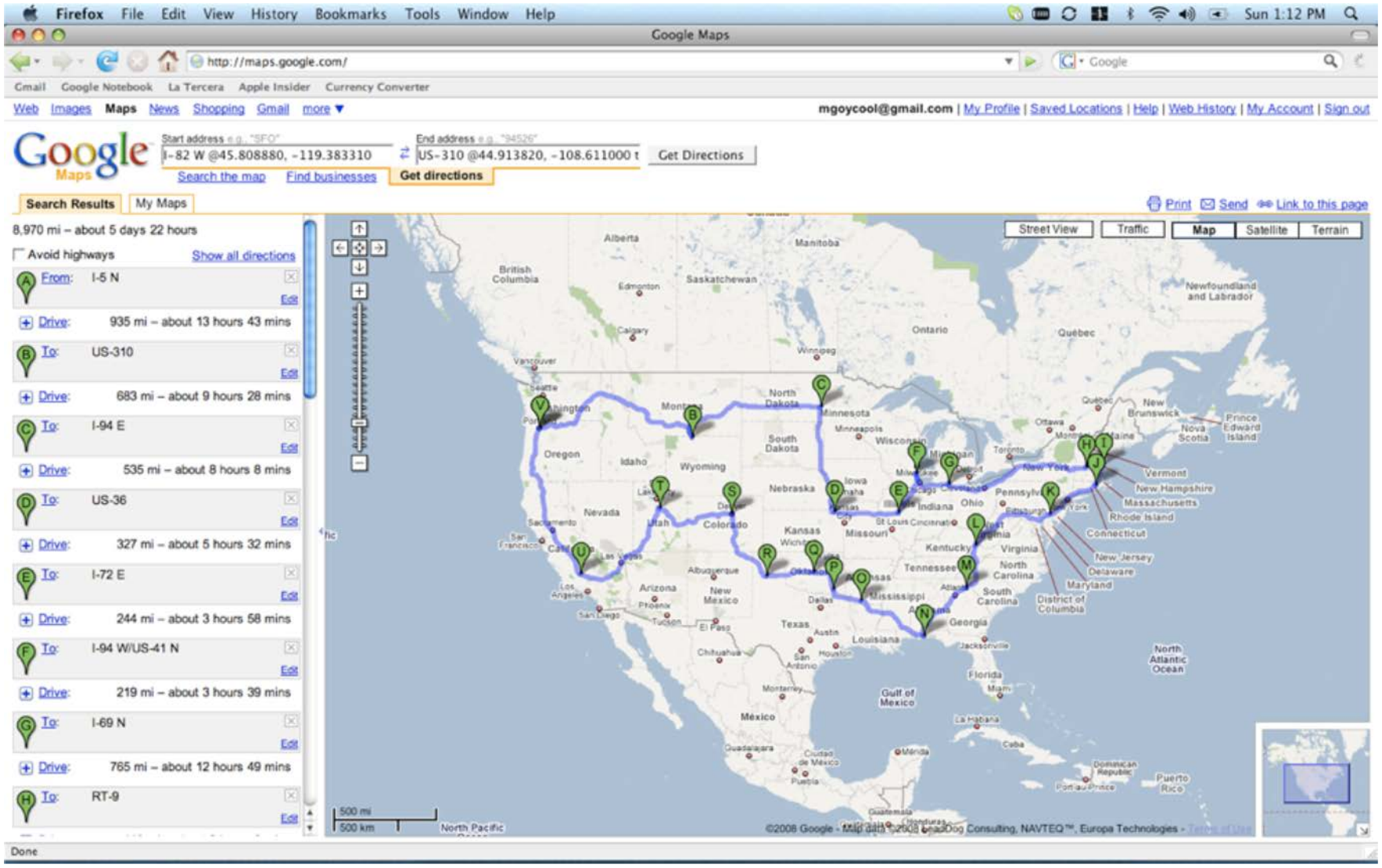
Can change to "= " if n=m

$$\sum_{j=1}^{m} x_{i,j} \leq 1 \qquad \forall i \in \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} x_{i,j} \geq 1 \qquad \forall j \in \{1, \ldots, m\}$$

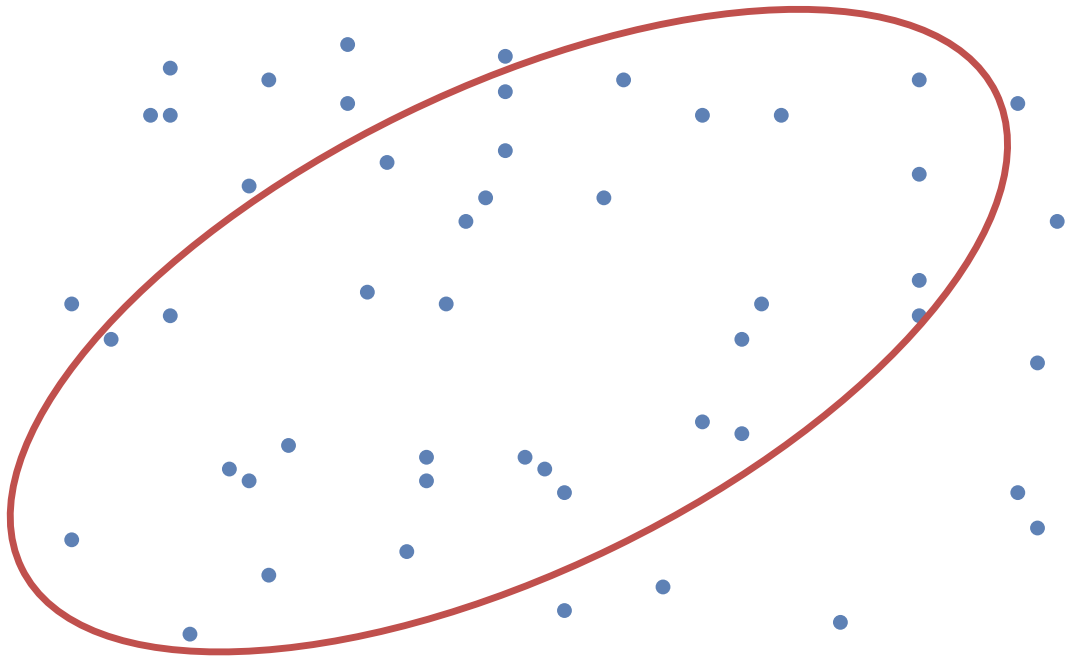
$$x_{i,j} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n\}, \; j \in \{1, \ldots, m\}$$

Traveling Salesman Problem : Visit all Cities Once



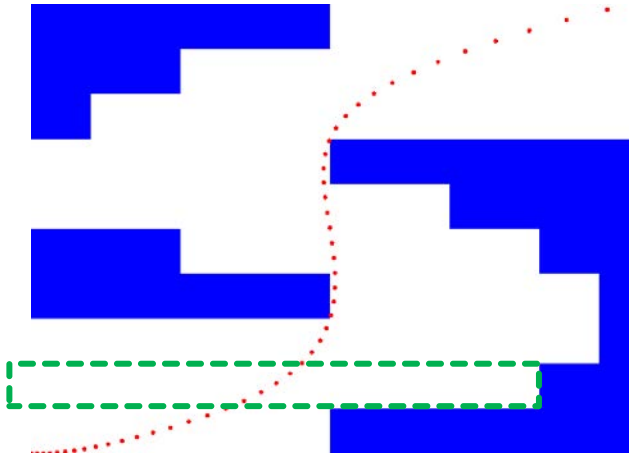
Mixed Integer Programming (MIP)

- Discrete and continuous variables or combinatorial constraints on continuous variables.
- Example: Find minimum volume ellipsoid that contains 90% of data points



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

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

$$\text{s.t.} \quad p_1(0) = X_0,\ p'(0) = X_0',\ p''(0) = X_0'' \qquad \text{Initial/Terminal Conditions}$$

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

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

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

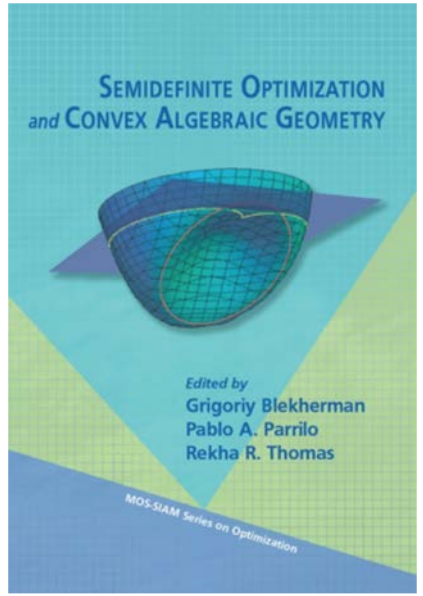
$$p_i''(T_{i+1}) = p_{i+1}''(T_{i+1}) \quad \forall i \in \{1, \ldots, N-1\}$$

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

Avoid Collision = Remain in Safe Regions

MIP

+



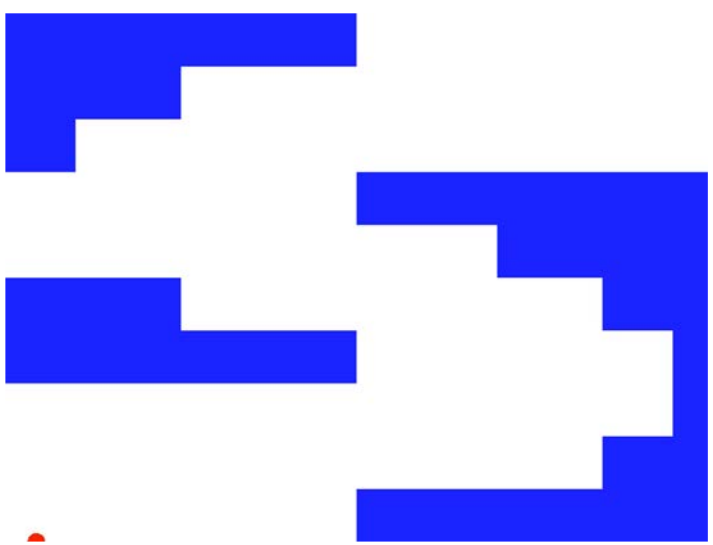
… Mixed Integer Semidefinite Programming

+

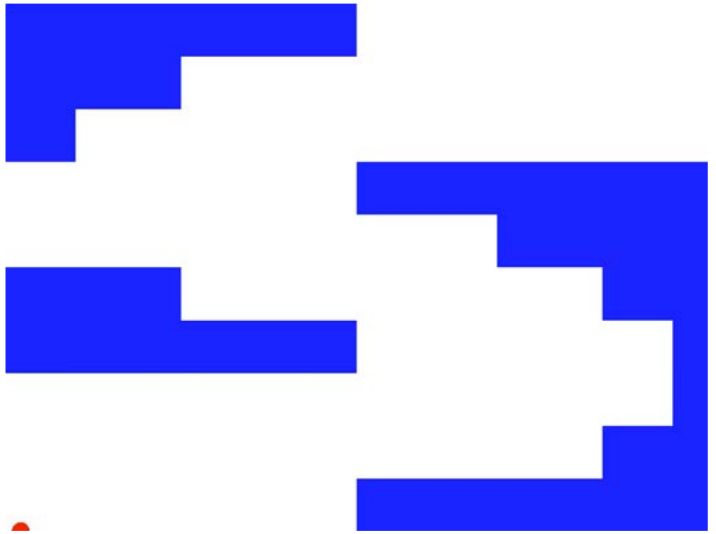

```
model = SOSModel(solver=PajaritoSolver())

@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

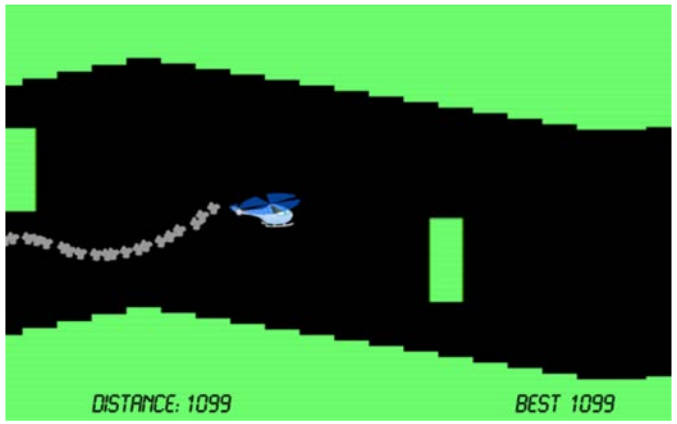


First Feasible Solution:
58 seconds

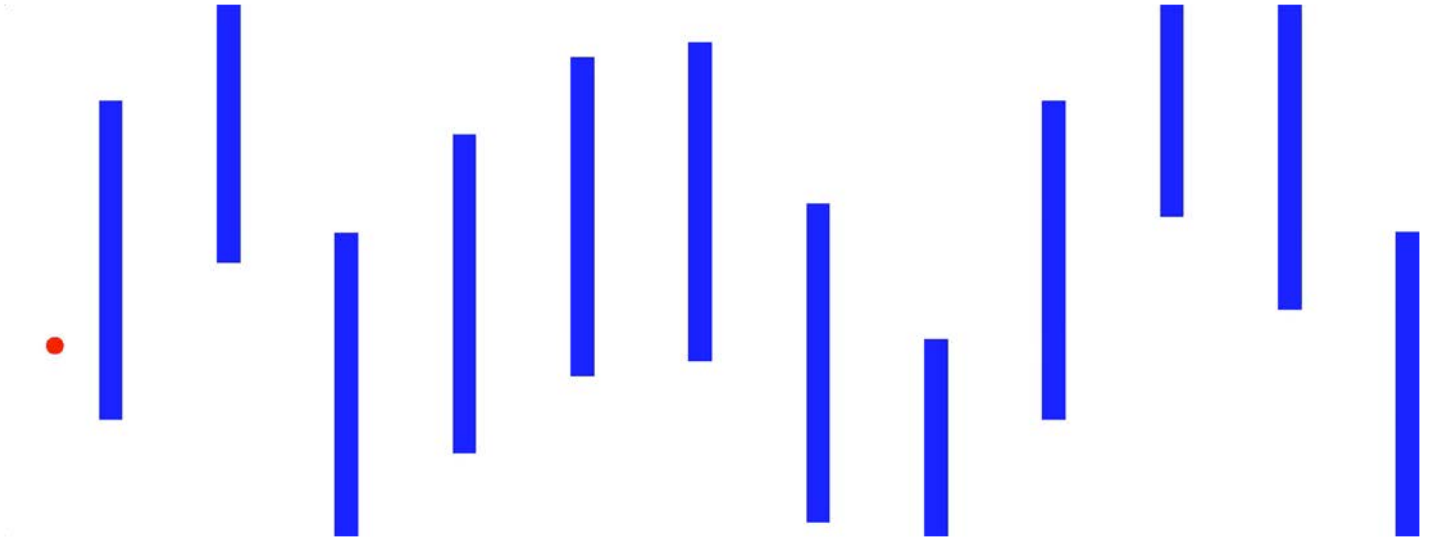


Optimal Solution:
651 seconds

Helicopter Game / Flappy Bird



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



MIP & Daily Fantasy Sports

The Greater Boston FOOD BANK

> $15K

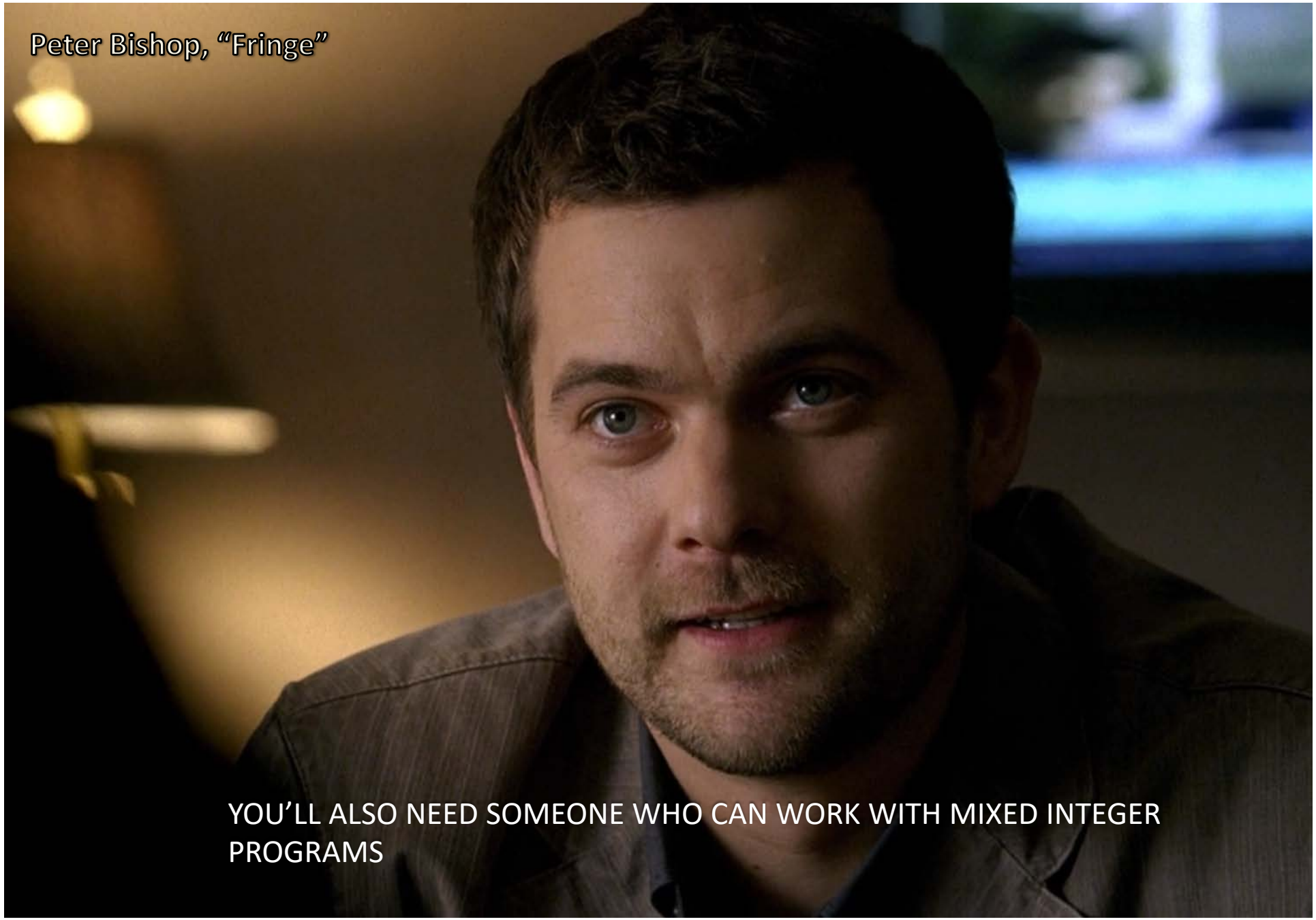
Download Code from Github:

https://github.com/dscotthunter/Fantasy-Hockey-IP-Code

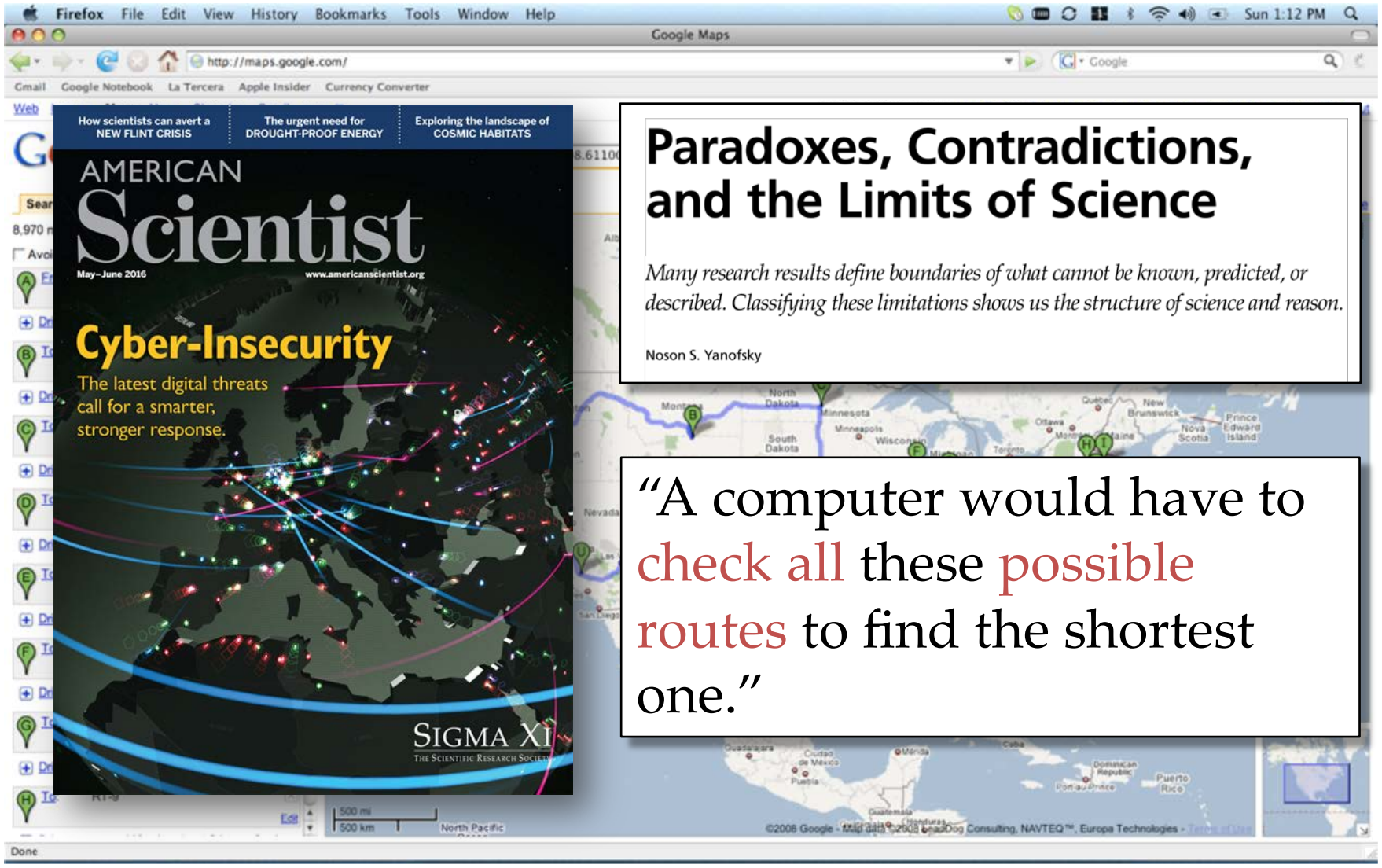
http://arxiv.org/pdf/1604.01455v1.pdf

Many Practical Applications

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


Peter Bishop, “Fringe”
YOU’LL ALSO NEED SOMEONE WHO CAN WORK WITH MIXED INTEGER PROGRAMS

How hard is MIP: Traveling Salesman Problem ?



MIP = Avoid (Complete) Enumeration

- Number of tours for 49 cities $= 48!/2 \approx 10^{60}$
- Fastest supercomputer $\approx 10^{17}\text{flops}$
- Assuming one floating point operation per tour:

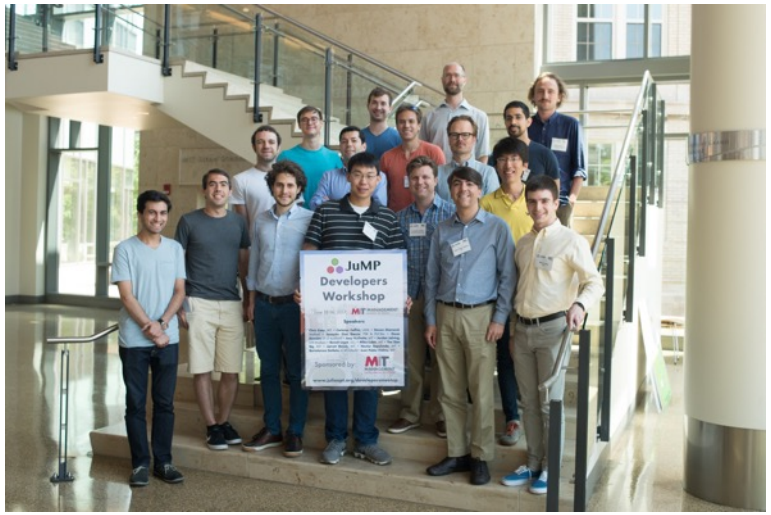
$$> 10^{35} \text{ years} \approx 10^{25} \text{ times the age of the universe!}$$

- How long does it take on an iphone?
  - < 1 sec ! Dantzig, Fulkerson and Johnson ✍️ in 54'
  - Even theoretically hard MIPs "can" be solved:
    - Open-source solvers: GLPK, CBC, etc.
    - Commercial: Gurobi, CPLEX, etc.
    - Modeling Language:

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!

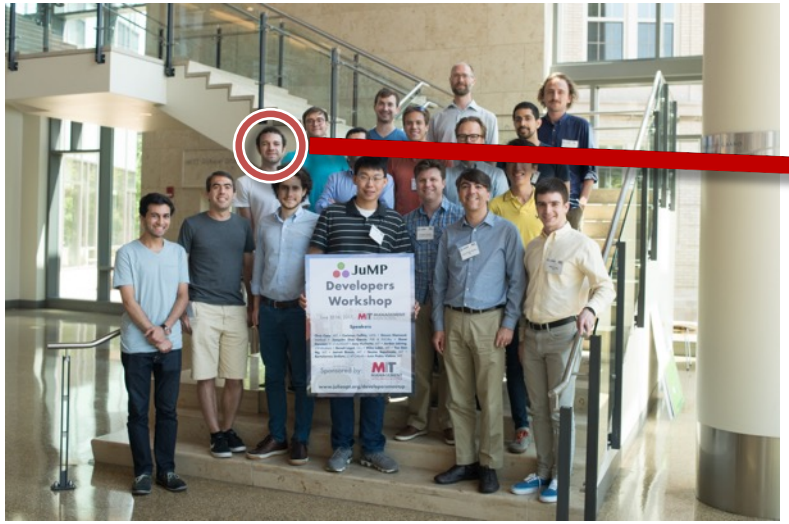
Modeling Tool	Linear / Quadratic	Convex		Nonconvex	Integer
		Conic	Smooth		
JuMP	✔	✔	✔	✔	✔
Convex.jl	✔	✔			✔
Solver					
CDD (.jl)	✔				
Clp (.jl)	✔				
OSQP (.jl)	✔				
Cbc (.jl)	✔				✔
GLPK (.jl)	✔				✔cb
CSDP (.jl)	✔	✔			
ECOS (.jl)	✔	✔			
SCS (.jl)	✔	✔			
SDPA (.jl)	✔	✔			
CPLEX (.jl)	✔	✔			✔cb
Gurobi (.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

Julia and JuMP In Production Environments

PSR

Peruvian Energy Ministry

The Nature Conservancy

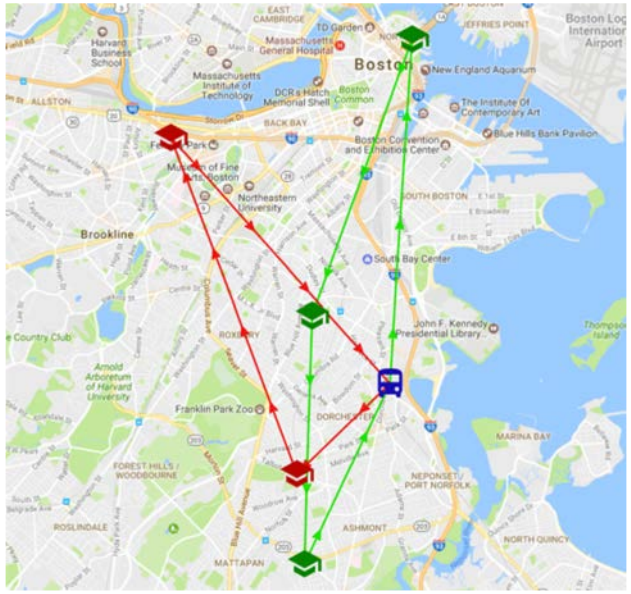


BID
Banco Interamericano de Desarrollo

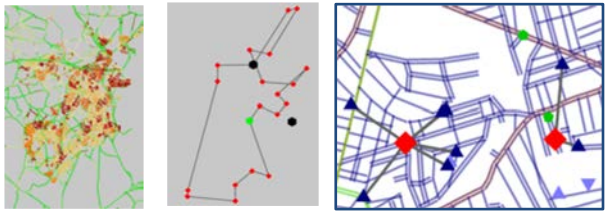
Northwest Power and Conservation Council

Joaquim Dias Garcia

Saving $3M – $5M in BPS Transportation Challenge



BOSTON Public Schools
Focus on Children



MIT MANAGEMENT SLOAN SCHOOL

Dimitris Bertsimas

OPERATIONS RESEARCH CENTER

Arthur Delarue

OPERATIONS RESEARCH CENTER

Sebastien Martin

Efficient Decarbonization of the Electrical Grid

Nestor Sepulveda

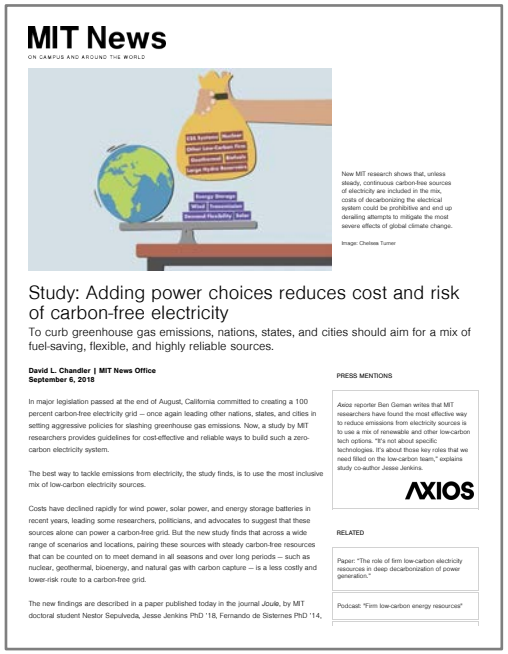
MITei
MIT Energy Initiative

Jesse Jenkins

IDSS
MIT INSTITUTE FOR DATA, SYSTEMS, AND SOCIETY

Richard Lester

NSE
Nuclear Science & Engineering at MIT
SCIENCE : SYSTEMS : SOCIETY



MIT News
ON CAMPUS AND AROUND THE WORLD

New MIT research shows that, unless steady, continuous carbon-free sources of electricity are included in the mix, costs of decarbonizing the electrical system could be prohibitive and end up derailing attempts to mitigate the most severe effects of global climate change.

Image: Chelsea Turner

Study: Adding power choices reduces cost and risk of carbon-free electricity

To curb greenhouse gas emissions, nations, states, and cities should aim for a mix of fuel-saving, flexible, and highly reliable sources.

David L. Chandler | MIT News Office
September 6, 2018

In major legislation passed at the end of August, California committed to creating a 100 percent carbon-free electricity grid -- once again leading other nations, states, and cities in setting aggressive policies for slashing greenhouse gas emissions. Now, a study by MIT researchers provides guidelines for cost-effective and reliable ways to build such a zero-carbon electricity system.

The best way to tackle emissions from electricity, the study finds, is to use the most inclusive mix of low-carbon electricity sources.

Costs have declined rapidly for wind power, solar power, and energy storage batteries in recent years, leading some researchers, politicians, and advocates to suggest that these sources alone can power a carbon-free grid. But the new study finds that across a wide range of scenarios and locations, pairing these sources with steady carbon-free resources that can be counted on to meet demand in all seasons and over long periods -- such as nuclear, geothermal, bioenergy, and natural gas with carbon capture -- is a less costly and lower-risk route to a carbon-free grid.

The new findings are described in a paper published today in the journal Joule, by MIT doctoral student Nestor Sepulveda, Jesse Jenkins PhD '18, Fernando de Sisternes PhD '14,

PRESS MENTIONS

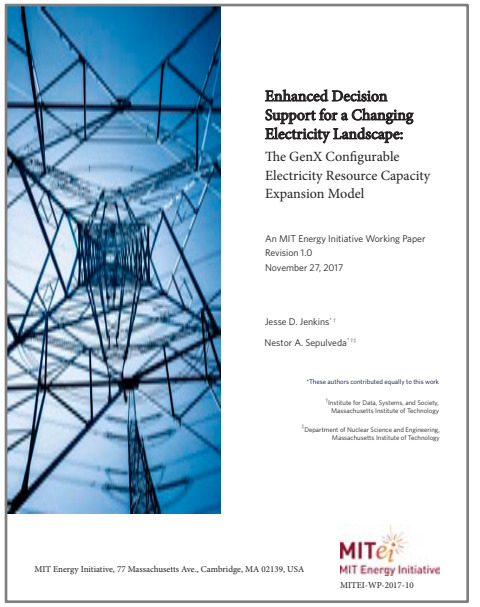
Axios reporter Ben Geman writes that MIT researchers have found the most effective way to reduce emissions from electricity sources is to use a mix of renewable and other low-carbon tech options. "It's not about specific technologies. It's about those key roles that we need filled on the low-carbon team," explains study co-author Jesse Jenkins.

AXIOS

RELATED

Paper: "The role of firm low-carbon electricity resources in deep decarbonization of power generation."

Podcast: "Firm low-carbon energy resources"



Enhanced Decision Support for a Changing Electricity Landscape:
The GenX Configurable Electricity Resource Capacity Expansion Model

An MIT Energy Initiative Working Paper
Revision 1.0
November 27, 2017

Jesse D. Jenkins[*†]
Nestor A. Sepulveda[*†‡]

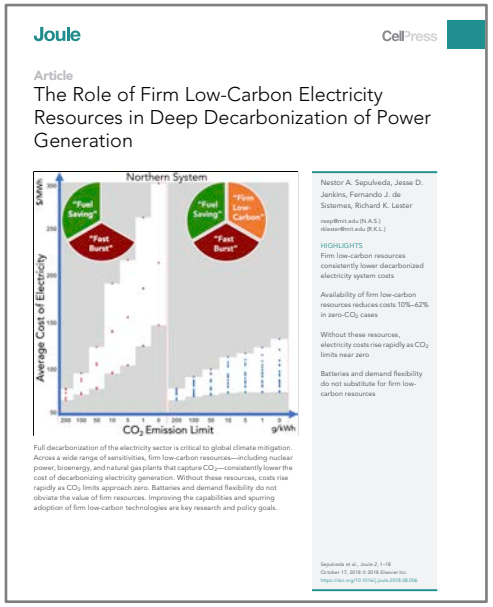
[*]These authors contributed equally to this work

[†]Institute for Data, Systems, and Society, Massachusetts Institute of Technology

[‡]Department of Nuclear Science and Engineering, Massachusetts Institute of Technology

MIT Energy Initiative, 77 Massachusetts Ave., Cambridge, MA 02139, USA

MITei
MIT Energy Initiative
MITEI-WP-2017-10



Joule
CellPress

Article
The Role of Firm Low-Carbon Electricity Resources in Deep Decarbonization of Power Generation

Nestor A. Sepulveda, Jesse D. Jenkins, Fernando J. de Sisternes, Richard K. Lester

HIGHLIGHTS

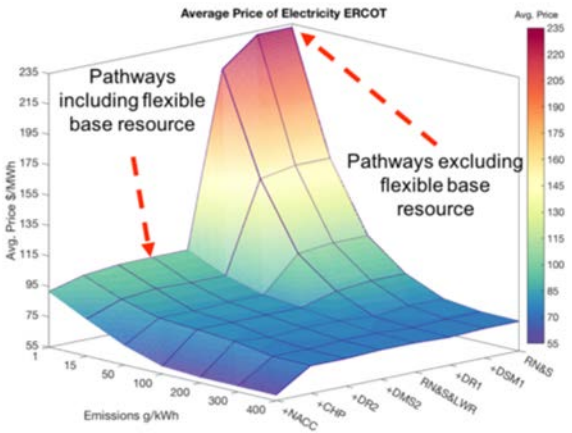
Firm low-carbon resources consistently lower decarbonized electricity system costs

Availability of firm low-carbon resources reduces costs 10%–62% in zero-CO_2 cases

Without these resources, electricity costs rise rapidly as CO_2 limits near zero

Batteries and demand flexibility do not substitute for firm low-carbon resources

Full decarbonization of the electricity sector is critical to global climate mitigation. Across a wide range of sensitivities, firm low-carbon resources—including nuclear power, bioenergy, and natural gas plants that capture CO_2—consistently lower the cost of decarbonizing electricity generation. Without these resources, costs rise rapidly as CO_2 limits approach zero. Batteries and demand flexibility do not obviate the value of firm resources. Improving the capabilities and spurring adoption of firm low-carbon technologies are key research and policy goals.



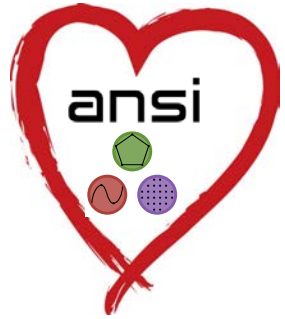
Advanced Network Science Initiative

Los Alamos NATIONAL LABORATORY

- 34 repositories (https://github.com/lanl-ansi)
  - PowerModels.jl
  - GraphicalModelLearning.jl
  - Juniper.jl (MINLP solver)
  - GasGridModels.jl
  - …

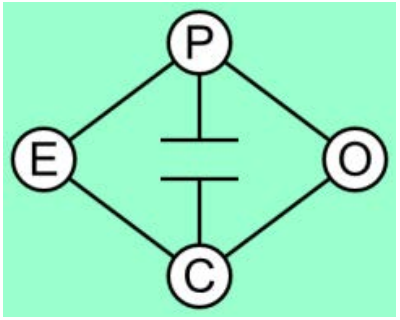
A Bit About LANL

ANSI
LOVES
JuliaOpt



Carleton Coffrin

Milk Output Optimizer, or MOO



THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

- From hydroelectric power to dairy farms:
  - rain, price of (milk/electricity) and substitutes (coal/corn)

Oscar Dowson

Optimal Control Using Sum-of-Squares Optimization

- EntropicCone.jl
- SwitchOnSafety.jl
- **SumOfSquares.jl**
- Polyhedra.jl
- MultivariatePolynomials.jl

UCL Université catholique de Louvain

Benoît Legat

Sum-of-Squares Programming in Julia with JuMP

Benoît Legat*, Chris Coey†, Robin Deits†, Joey Huchette† and Amelia Perry†
* UCLouvain, † MIT

Sum-of-Squares (SOS) Programming

Nonnegative quadratic forms into sum of squares

$$(x_1,x_2,x_3) \xrightarrow{p(x)=x^\top Q x} \text{unique}$$

$$x_1^2+2x_1x_2+5x_2^2+4x_2x_3+x_3^2 = x^\top \begin{bmatrix} 1 & 1 & 0 \\ 1 & 5 & 2 \\ 0 & 2 & 1 \end{bmatrix} x$$

$$p(x) \geq 0 \; \forall x \iff Q \succeq 0 \quad \text{(cholesky)}$$

$$(x_1+x_2)^2+(2x_2+x_3)^2 \longleftarrow x^\top \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}^\top \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} x$$

Nonnegative polynomial into sum of squares

$$(x_1,x_2,x_3) \xrightarrow{p(x)=X^\top Q X} (x_1,x_1x_2,x_2) \text{ not unique}$$

$$x_1^4+2x_1^3x_2+5x_1^2x_3^2+4x_1x_2^2+x_2^2 = X^\top \begin{bmatrix} 1 & 1 & 0 \\ 1 & 5 & 2 \\ 0 & 2 & 1 \end{bmatrix} X$$

$$p(x) \geq 0 \; \forall x \impliedby Q \succeq 0 \quad \text{(cholesky)}$$

$$(x_1+x_1x_2)^2+(2x_1x_2+x_2)^2 \longleftarrow X^\top \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}^\top \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} X$$

When is nonnegativity equivalent to sum of squares ?

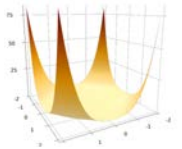
Determining whether a polynomial is nonnegative is NP-hard.

Hilbert 1888 Nonnegativity of $p(x)$ of n variables and degree $2d$ is equivalent to sum of squares in the following three cases:
- $n = 1$: Univariate polynomials
- $2d = 2$: Quadratic polynomials
- $n = 2$, $2d = 4$: Bivariate quartics

Motzkin 1967 First explicit example:

$$x_1^4x_2^2+x_1^2x_2^4+1-3x_1^2x_2^2 \geq 0 \quad \forall x$$

but is not a sum of squares.



Manipulating Polynomials

Two implementations: `TypedPolynomials.jl` and `DynamicPolynomials.jl`.
One common independent interface: `MultivariatePolynomials.jl`.

```
@polyvar y # one variable
@polyvar x[1:2] # tuple/vector
```

Build a vector of monomials:
- (x_1^2, x_1x_2, x_2^2): `X = monomials(x, 2)`
- $(x_1^2, x_1x_2, x_2^2, x_1, x_2, 1)$: `X = monomials(x, 0:2)`

Polynomial variables

By hand, with an integer decision variable `a` and real decision variable `b`:

```
@variable(model, a, Int)
@variable(model, b)
p = a*x^2 + (a+b)*y^2*x + b*y^3
```

From a polynomial basis, e.g. the *scaled monomial* basis, with integer decision variables as coefficients:

```
@variable(model,
          Poly(ScaledMonomialBasis(X)),
          Int)
```

Polynomial constraints

Constrain $p(x,y) \geq q(x,y)$ $\forall x, y$ such that $x \geq 0, y \geq 0, x+y \geq 1$ using the scaled monomial basis:

```
S = @set x >= 0 && y >= 0 && x + y >= 1
@constraint(model, p >= q, domain = S,
            basis = ScaledMonomialBasis)
```

Interpreted as:

```
@constraint(model, p - q in SOSCone(),
            domain = S,
            basis = ScaledMonomialBasis)
```

To use DSOS or SDSOS (Ahmadi, Majumdar 2017):

```
@constraint(model, p - q in DSOSCone())
@constraint(model, p - q in SDSOSCone())
```

SOS on algebraic domain

The domain S is defined by equalities forming an *algebraic variety* V and inequalities q_i. We search for Sum-of-Squares polynomials s_i such that

$$p(x) - q(x) \equiv s_0(x) + s_1(x)q_1(x) + \cdots \pmod{V}$$

The Gröbner basis of V is computed the equation is reduced modulo V.

Dual value

The dual of the constraint is a positive semidefinite (PSD) matrix of moments μ. The `extractatoms` function attempts to find an *atomic* measure with these moments by solving an algebraic system.

Sum-of-Squares extension

MathOptInterface.jl (MOI)

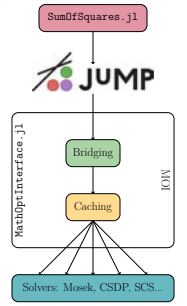
MOI is an abstraction layer for mathematical optimization solvers. A constraint is defined by a "function" ∈ "set" pair.

MOI extension: `AbstractVectorFunction` ∈ `SOS(X)` (resp. `WSOS(X)`): SOS constraint without (resp. with) `domain` equipped with a bridge to `AbstractVectorFunction` ∈ `PSD` (resp. `SOS(X)`).

JuMP

JuMP is a domain-specific modeling language for mathematical optimization. It stores the problem directly (a cache can optionally be used) in the solver using MOI.

JuMP extension: $p(x) \geq q(x)$ and $p(x) \in$ `SOS()` are rewritten into MOI SOS or WSOS constraints, e.g. $x^2+y^2 \geq 2xy$ is rewritten into $[1,-2,1] \in \text{SOS}(x^2, xy, y^2)$. $p(x) \in$ `DSOS()` (resp. `SDSOS()`) is rewritten into linear (resp. second-order cone) constraints.



Bridging Automatic reformulation of a constraint into an equivalent form supported by the solver, e.g. quadratic constraint into second-order cone constraint. In particular, reformulates SOS/WSOS constraints into PSD constraints. An interior-point solver that natively supports SOS and WSOS without reformulation to SDP using the approach of (Papp, Yildz 2017) is under development.

Caching Cache of the problem data in case the solver do not support a modification (can be disabled). For instance, Mosek provides many modification capabilities in the API but CSDP only support pre-allocating and then loading the whole problem at once.

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

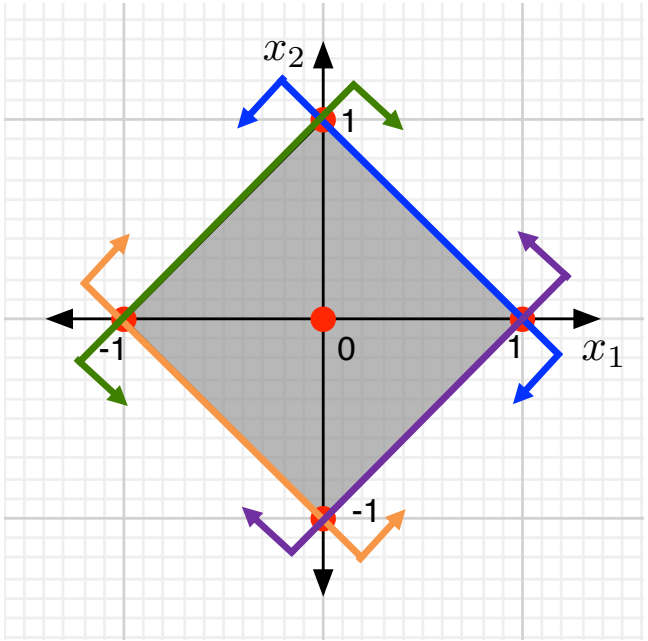
- Mature: Linear and Quadratic (Conic Quadratic/SOCP)
  - Commercial:

  - "Open Source"

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

Solving MIPs: Step 1 = Linear Programming

$$\begin{aligned} \max \quad & x_2 \\ s.t. \quad & x_1 + x_2 \leq 1 \\ & -x_1 - x_2 \leq 1 \\ & +x_1 - x_2 \leq 1 \\ & -x_1 + x_2 \leq 1 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$



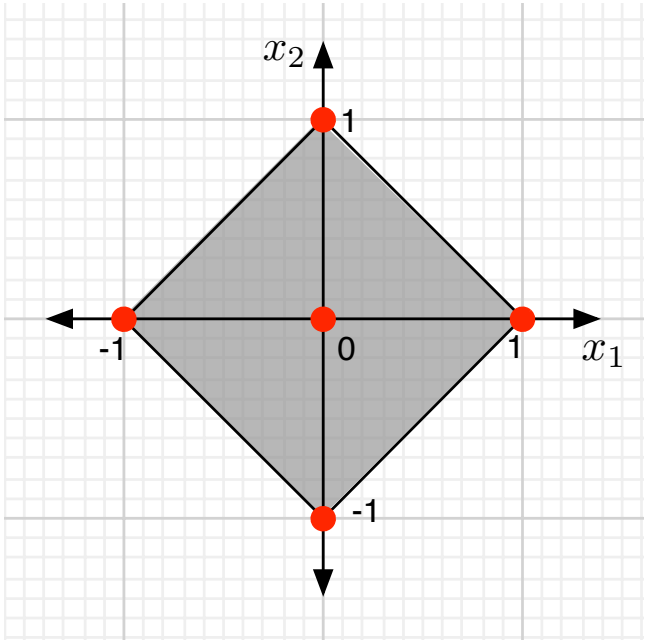
← Linear Programming (LP) Relaxation

- Solving LPs is easy in theory and practice.
- One reason = LP duality
  - Suppose I guess optimum $x_1 = 0$ and $x_2 = 1$.
  - How do I prove that for all solutions of LP $x_2 \leq 1$?

$$\begin{aligned} & (1/2) \times (\quad x_1 + x_2 \leq 1) \\ + \quad & (1/2) \times (-x_1 + x_2 \leq 1) \\ \hline & x_2 \leq 1 \end{aligned}$$

Solving MIPs: Step 1 = Linear Programming

$$
\begin{aligned}
\max \quad & x_2 \\
s.t. \quad & x_1 + x_2 \leq 1 \\
& -x_1 - x_2 \leq 1 \\
& +x_1 - x_2 \leq 1 \\
& -x_1 + x_2 \leq 1 \\
& x_1, x_2 \in \mathbb{Z}
\end{aligned}
$$



← Linear Programming (LP) Relaxation

- LP relaxation always gives a (upper) bound on the MIP:
  - If solution of LP is "integer" then you solved the MIP
  - LP solvers return "corner" solution, which fixes "multiple optima" (e.g. $\max x_1 + x_2$)
  - Solve LP relaxation of assignment problem with JuMP. Is solution integer?

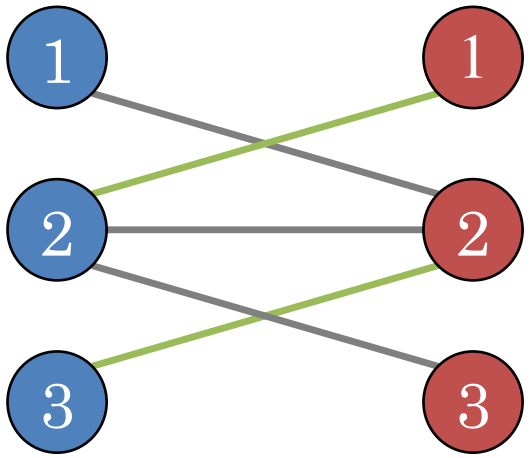
Easy Problems : LP Relaxation Always Integral

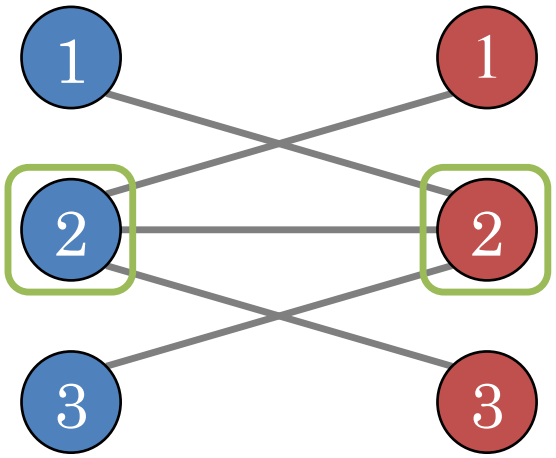
Consequence of LP duality: Kőnig's theorem

- Largest Matching
  - Pick edges, at most one edge per node

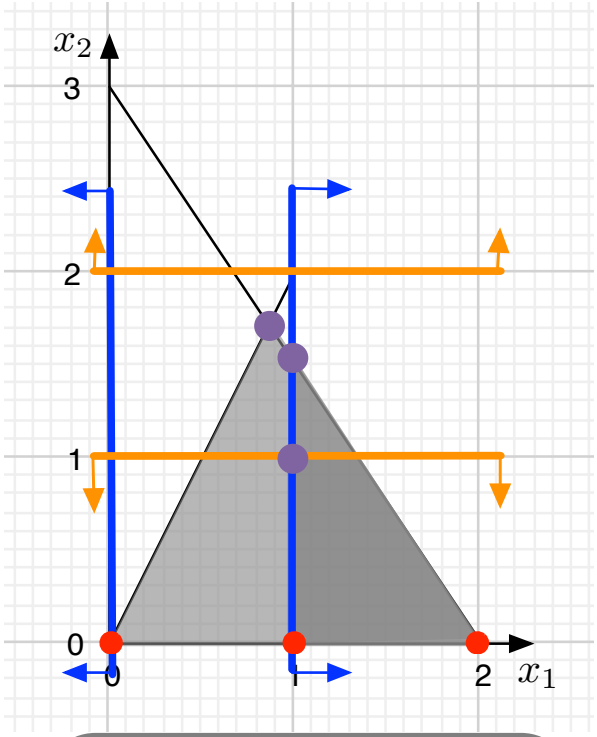
=

- Smallest Node Cover
  - Pick nodes that touch all edges



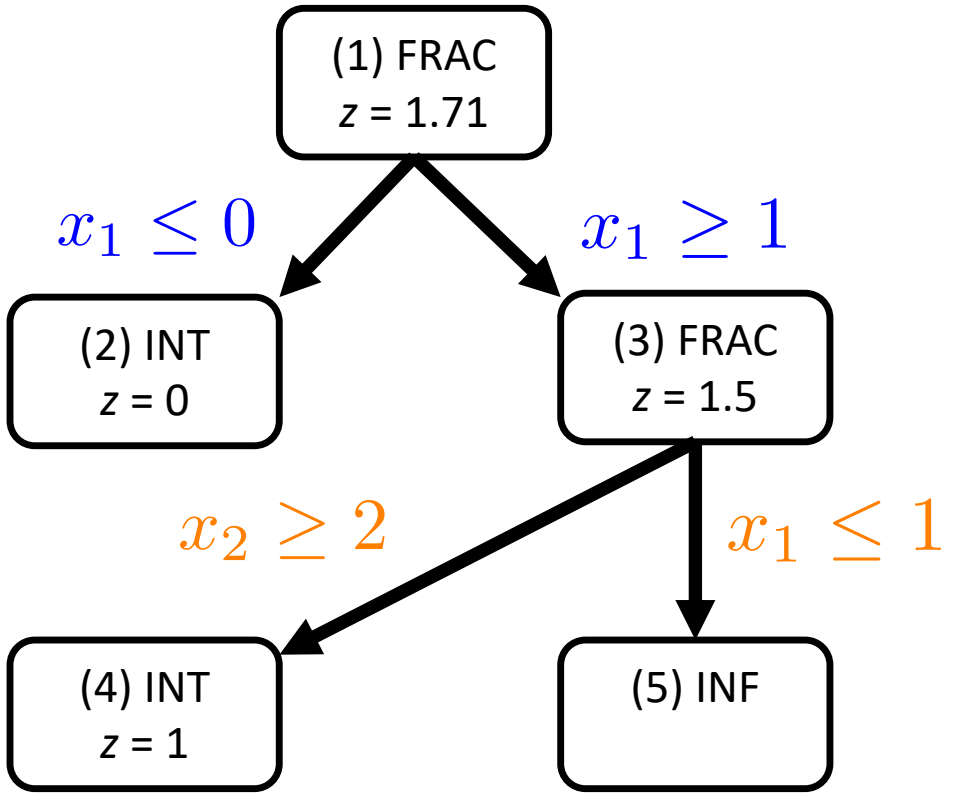


Solving MIPs: Step 2 = Branch-and-Bound



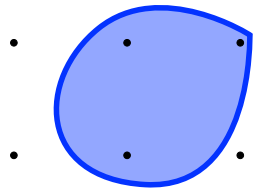
$$\begin{aligned} \max z := x_2 \\ 3x_1 + 2x_2 &\le 6 \\ -2x_1 + x_2 &\le 0 \\ x_1,\ x_2 &\ge 0 \\ x_1,\ x_2 &\in \mathbb{Z} \end{aligned}$$

← Linear Programming (LP) Relaxation

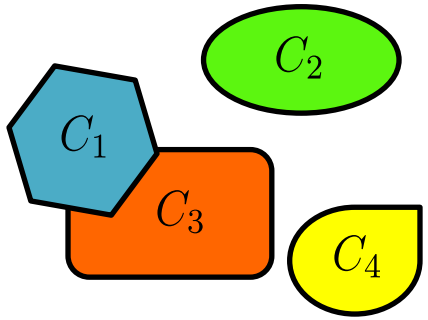


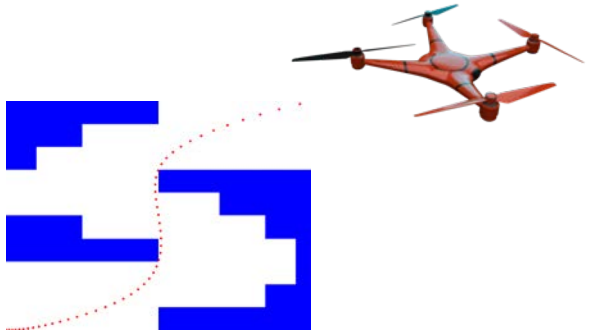
Mixed Integer Convex Optimization (MICONV)

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



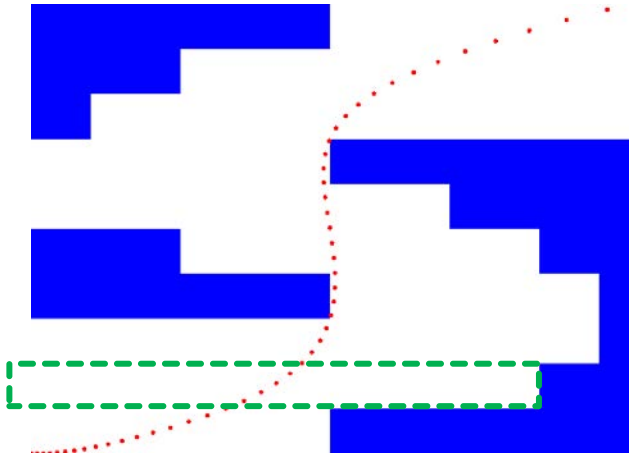
convex f and C.





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

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

$$\text{s.t.} \quad p_1(0) = X_0,\ p'(0) = X_0',\ p''(0) = X_0'' \qquad \text{Initial/Terminal Conditions}$$

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

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

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

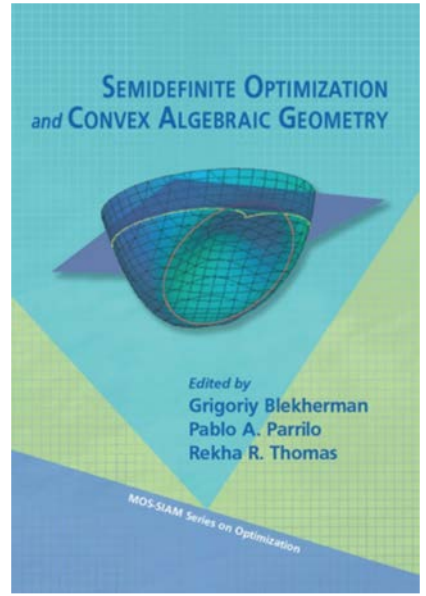
$$p_i''(T_{i+1}) = p_{i+1}''(T_{i+1}) \quad \forall i \in \{1, \ldots, N-1\}$$

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

Avoid Collision = Remain in Safe Regions

MIP

+



… Mixed Integer Semidefinite Programming

~~Disiunctive~~ *Polynomial* Optimization Formulation

Mixed-Integer

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

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

$$\text{s.t.} \quad p_1(0) = X_0,\ p'(0) = X_0',\ p''(0) = X_0''$$

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

Initial/Terminal Conditions

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

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

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

Interstitial Smoothing Conditions

$$b_j^r + M_j^r(1 - z_{i,r}) - A_j^r p_i(t) \geq 0 \quad \text{for } t \in [T_i, T_{i+1}] \quad \forall i, j, r$$

$$\sum_{r=1}^{R} z_{i,r} = 1 \quad \forall i, \quad z \in \{0,1\}^{N \times R}$$

Avoid Collision = Remain in Safe Regions

Mixed-Integer Semidefinite Programming Formulation

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

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

$$\text{s.t.} \quad p_1(0) = X_0,\ p'(0) = X_0',\ p''(0) = X_0''$$

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

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

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

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

Interstitial Smoothing Conditions

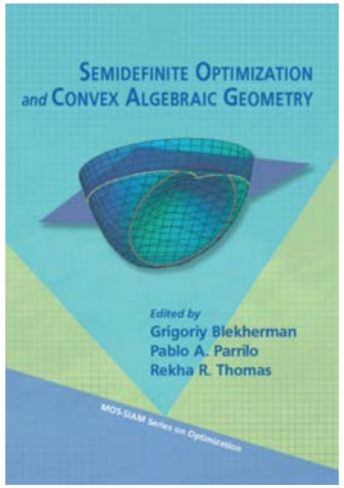
$$b_j^r + M_j^r(1 - z_{i,r}) - A_j^r p_i(t) \text{ is SOS for } t \in [T_i, T_{i+1}] \quad \forall i, j, r$$

← SDP

$$\sum_{r=1}^{R} z_{i,r} = 1 \quad \forall i, z \in \{0,1\}^{N \times R}$$

← MIP

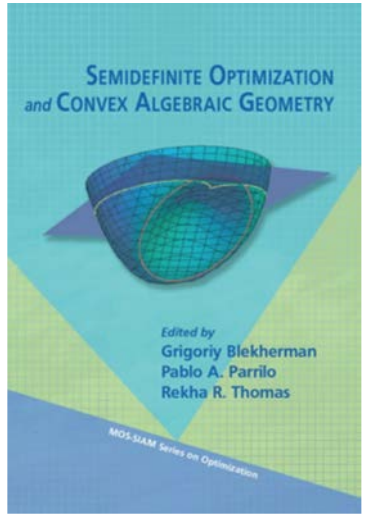
Avoid Collision = Remain in Safe Regions



From Sum of Squares to Semidefinite Programming

- Sufficient condition for non-negative polynomial:
  - Sum of Squares : $f(x) = \sum_i g_i^2(x)$
  - SDP representable for fixed degree:

$$\text{degree} \leq k \quad \rightarrow \quad (k-1) \times (k-1) \text{ matrices}$$



- MI-SOS:
  - Low degree polynomials (≤3):
    - MI-SOCP: solvable by Gurobi/CPLEX
    - Deits/Tedrake:2015
  - Higher degree polynomials:
    - MI-SDP: solvable by Pajarito

GUROBI OPTIMIZATION

IBM CPLEX

optimized by Xpress FICO

MICONV B&B Algorithms

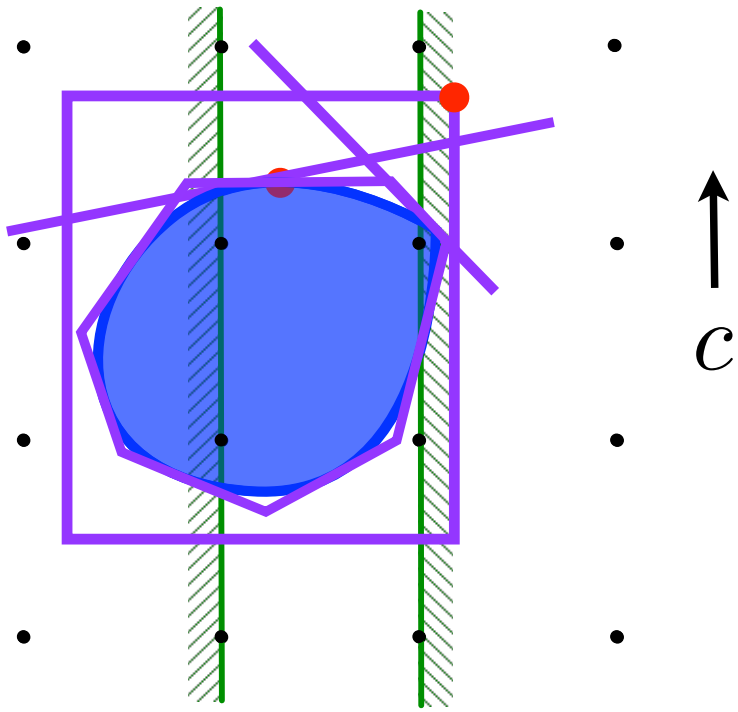
- NLP (QCP) Based B&B
- (Dynamic) LP Based B&B
  - Few cuts = high speed.
  - Possible slow convergence.
- Lifted LP B&B
  - Extended or Lifted relaxation.
  - Static relaxation
    - Mimic NLP B&B.
  - Dynamic relaxation
    - Standard LP B&B

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

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

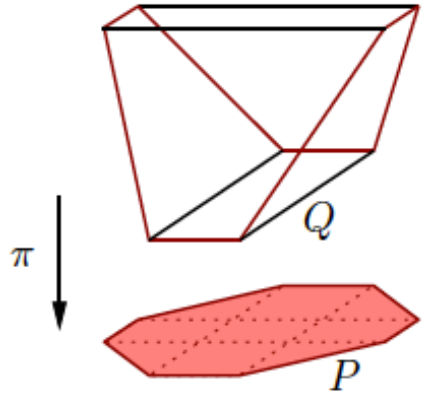
$$g_i(x) \leq 0,\ i \in I, \quad x \in \mathbb{Z}^n$$

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



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)



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

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

$$\|y\|_2 \le y_0$$

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

Not MICONV but, Mixed Integer Conic Programming (MICP)

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

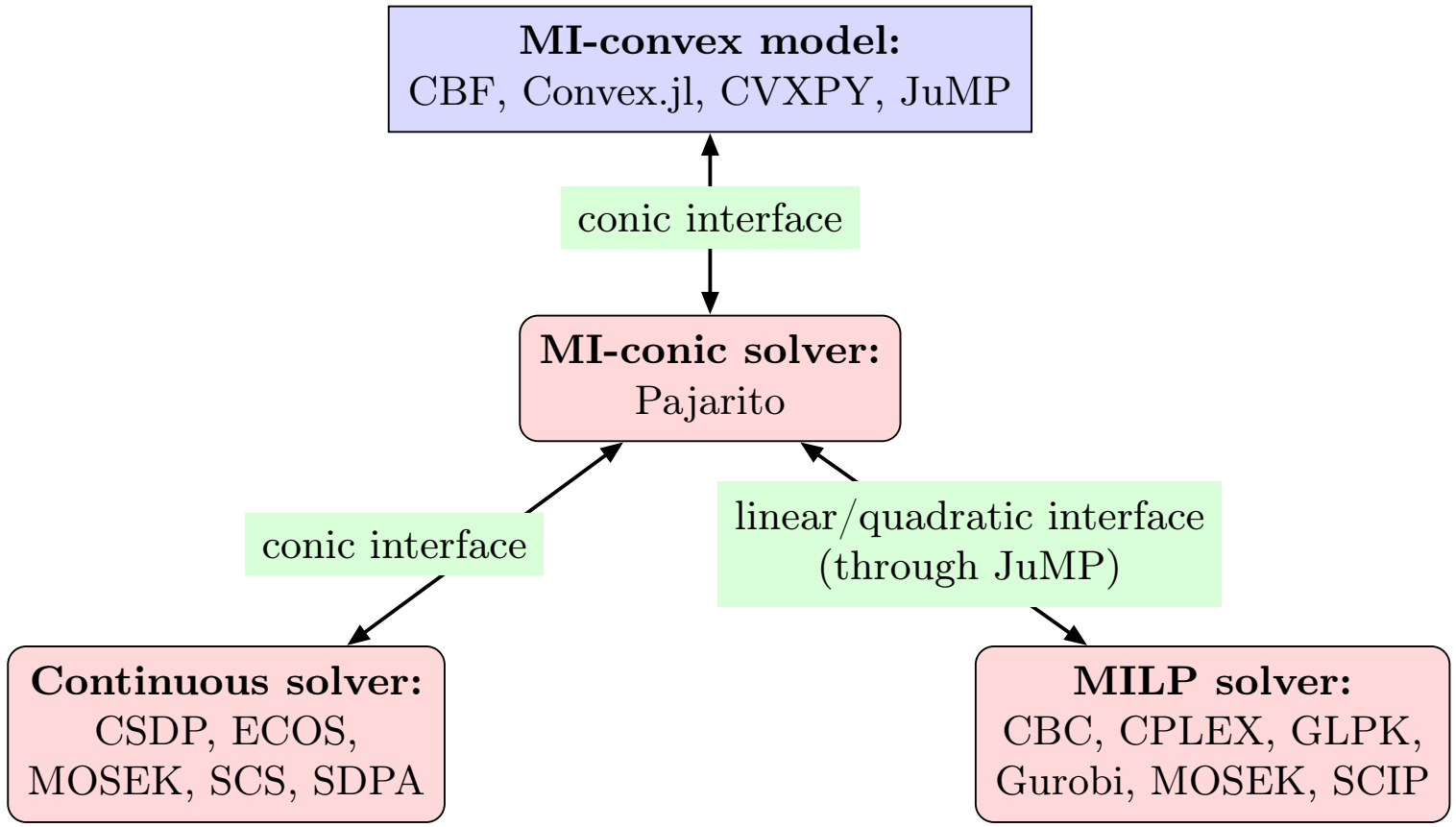
$$\boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \in \mathcal{C}_k \quad \forall k \in [M]$$

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

- $\mathcal{C}_k$ 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)

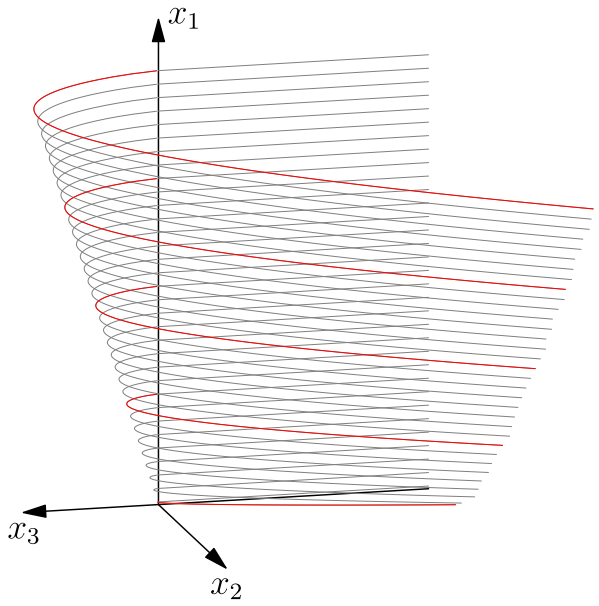
	solver	statuses				time (s)
		ok	limit	error	wrong	
open source	Bonmin-BB	34	44	11	31	463
	Bonmin-OA	25	53	29	13	726
	Bonmin-OA-D	30	48	29	13	610
	Pajarito-GLPK-ECOS	56	60	3	1	377
	Pajarito-CBC-ECOS	78	30	3	9	163
restricted	SCIP (4.0.0)	74	35	8	3	160
	CPLEX (12.7.0)	90	16	5	9	50
	Pajarito-CPLEX-MOSEK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

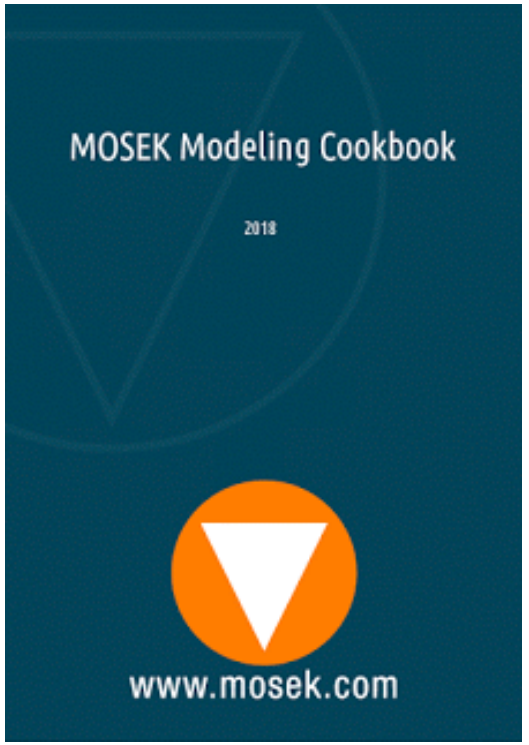
$$x_3 \leq x_2 \log(x_1/x_2), \;\; x_1, x_2 > 0.$$



- 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)

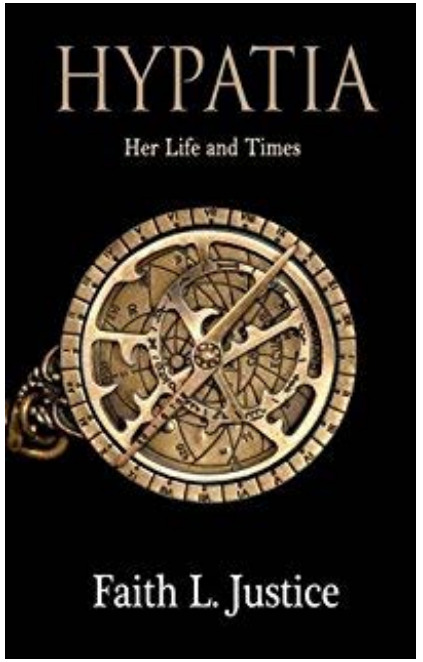


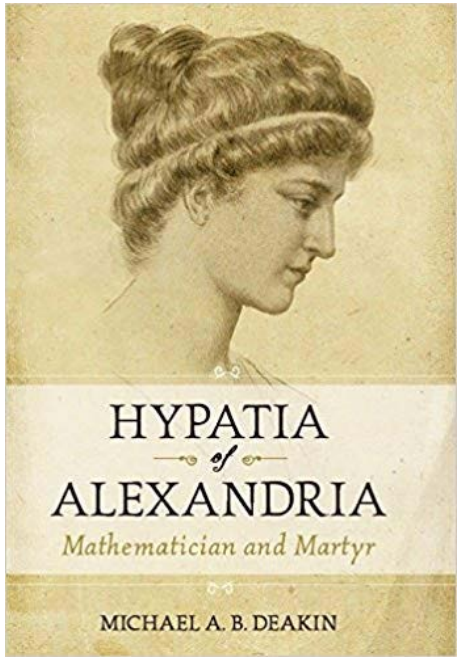
https://themosekblog.blogspot.com/2018/05/new-modeling-cookbook.html

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

Linear Optimization

Polynomial Envelope

Polynomial Minimization

			Aug 5	Aug 19	Aug 23
test	iters	Matlab	75cba5f	c9f1eb5	133b422
dense lp	65	5.8	4.1	2.03	1.25
envelope	30	0.085	0.043	0.020	x
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!