Systematically building mixed-integer programming formulations using JuMP and Julia

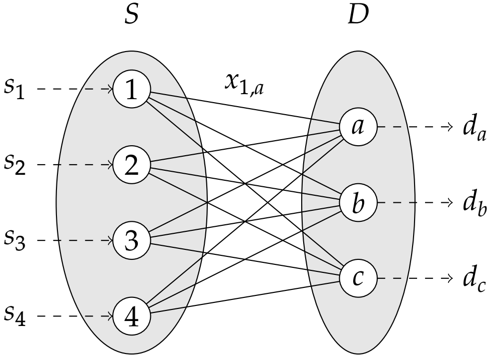
Joey Huchette

MIT (three weeks ago)
Google (in three weeks)
??? (right now)

June 27, 2018

Motivating example: The transportation problem

- *How do I route natural gas from processing facilities* (S) *to distribution centers* (D) *while minimizing transportation costs?*



- Network flow problem on a bipartite graph

Motivating example: The transportation problem

- Cost = linear function over flow on each arc (fixed unit costs)

$$
\begin{aligned}
\min_{x} \quad & \sum_{i \in S} \sum_{j \in D} c_{i,j} x_{i,j} & \\
\text{s.t.} \quad & \sum_{j \in D} x_{i,j} = s_i & \forall i \in S \\
& \sum_{i \in S} x_{i,j} = d_j & \forall j \in D \\
& x_{i,j} \geq 0 & \forall i \in S, j \in D
\end{aligned}
$$

- Linear optimization problem (with specialized algorithms)

Motivating example: The transportation problem

- Cost $=$ concave function over flow on each arc (*economies of scale*)

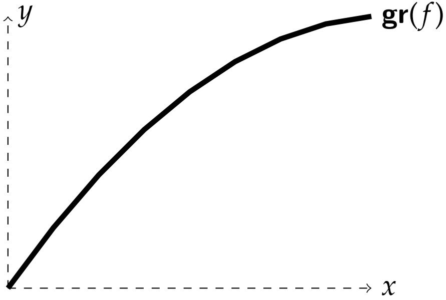
$$\begin{aligned}
\min_{x} \quad & \sum_{i \in S} \sum_{j \in D} f_{i,j}(x_{i,j}) & \\
\text{s.t.} \quad & \sum_{j \in D} x_{i,j} = s_i & \forall i \in S \\
& \sum_{i \in S} x_{i,j} = d_j & \forall j \in D \\
& x_{i,j} \geq 0 & \forall i \in S, j \in D
\end{aligned}$$

- How do we solve this *nonconvex optimization problem*?

Univariate piecewise linear functions

Want to optimize over the *graph* of a nonconvex function:

$$\mathbf{gr}(f) = \left\{(x, f(x)) : x \in D\right\}$$



Univariate piecewise linear functions

Want to optimize over the *graph* of a nonconvex function:

$$\mathbf{gr}(f) = \Big\{(x, f(x)) : x \in D\Big\}$$

$$\begin{aligned}
\min_{x} \quad & \sum_{i \in S} \sum_{j \in D} f_{i,j}(x_{i,j}) & \\
\text{s.t.} \quad & \sum_{j \in D} x_{i,j} = s_i & \forall i \in S \\
& \sum_{i \in S} x_{i,j} = d_j & \forall j \in D \\
& x_{i,j} \geq 0 & \forall i \in S, j \in D
\end{aligned}$$

Univariate piecewise linear functions

Want to optimize over the *graph* of a nonconvex function:

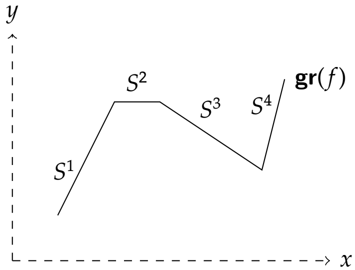
$$\mathbf{gr}(f) = \left\{(x, f(x)) : x \in D\right\}$$

$$\begin{aligned}
\min_{x} \quad & \sum_{i \in S} \sum_{j \in D} y_{i,j} \\
\text{s.t.} \quad & \sum_{j \in D} x_{i,j} = s_i && \forall i \in S \\
& \sum_{i \in S} x_{i,j} = d_j && \forall j \in D \\
& x_{i,j} \geq 0 && \forall i \in S, j \in D \\
& (x_{i,j}, y_{i,j}) \in \mathbf{gr}(f_{i,j}) && \forall i \in S, j \in D
\end{aligned}$$

Nonconvex optimization using mixed-integer programming

1. Write as a *disjunctive constraint*:

$$x \in \mathbf{gr}(f) = \bigcup_{i=1}^{d} S^i \subseteq \mathbb{R}^n$$

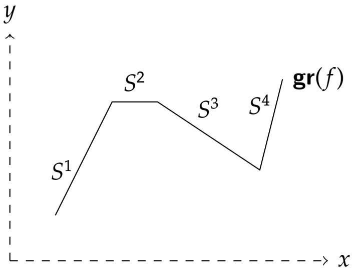


Nonconvex optimization using mixed-integer programming

1. Write as a *disjunctive constraint*:

$$x \in \mathbf{gr}(f) = \bigcup_{i=1}^{d} S^i \subseteq \mathbb{R}^n$$

2. Introduce integer variables $z \in \mathbb{Z}^r$



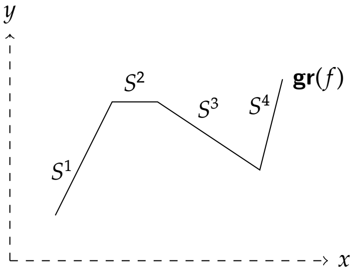
Nonconvex optimization using mixed-integer programming

1. Write as a *disjunctive constraint*:

$$x \in \mathbf{gr}(f) = \bigcup_{i=1}^{d} S^i \subseteq \mathbb{R}^n$$

2. Introduce integer variables $z \in \mathbb{Z}^r$
3. Build LP relaxation $Q \subseteq \mathbb{R}^{n+r}$ so:

$$\mathrm{Proj}_x \left(Q \cap (\mathbb{R}^n \times \mathbb{Z}^r)\right) = \bigcup_{i=1}^{d} S^i$$



Nonconvex optimization using mixed-integer programming

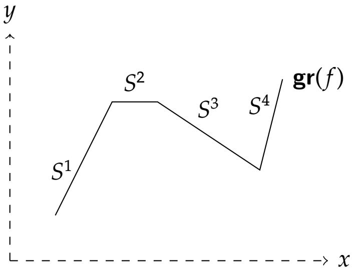
1. Write as a *disjunctive constraint*:

$$x \in \mathbf{gr}(f) = \bigcup_{i=1}^{d} S^i \subseteq \mathbb{R}^n$$

2. Introduce integer variables $z \in \mathbb{Z}^r$
3. Build LP relaxation $Q \subseteq \mathbb{R}^{n+r}$ so:

$$\mathrm{Proj}_x \left(Q \cap (\mathbb{R}^n \times \mathbb{Z}^r)\right) = \bigcup_{i=1}^{d} S^i$$

? How do we choose Q?



The right formulation matters!

N	Metric	MC	CC	DLog	Stencil
4	Mean (s)	1.4	1.5	0.9	0.4
	Win	0	0	0	100
8	Mean (s)	39.3	97.2	12.6	2.7
	Win	0	0	0	100
16	Mean (s)	1370.9	1648.1	352.8	24.6
	Fail	53	66	6	0
	Win	0	0	0	80
32	Mean (s)	1800.0	1800.0	1499.6	133.5
	Fail	80	80	50	0
	Win	0	0	0	80

Solve time (in seconds, with CPLEX v12.7.0). Functions have N^2 pieces, fixed network $|S| = |D| = 5$.

- Advanced `Stencil` formulation is the fastest on every instance
- $>$10x speedup on average for medium/large instances
- Previous approaches could not solve 50 of 80 largest instances

The right formulation matters!

N	Metric	`MC`	`CC`	`DLog`	`Stencil`
4	Mean (s)	1.4	1.5	0.9	0.4
	Win	0	0	0	100
8	Mean (s)	39.3	97.2	12.6	2.7
	Win	0	0	0	100
16	Mean (s)	1370.9	1648.1	352.8	24.6
	Fail	53	66	6	0
	Win	0	0	0	80
32	Mean (s)	1800.0	1800.0	1499.6	133.5
	Fail	80	80	50	0
	Win	0	0	0	80

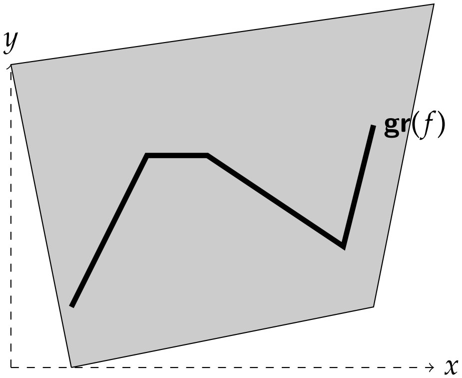
Solve time (in seconds, with CPLEX v12.7.0). Functions have N pieces, fixed network $|S| = |D| = 5$.

- Advanced `Stencil` formulation is the fastest on every instance
- $>$10x speedup on average for medium/large instances
- Previous approaches could not solve 50 of 80 largest instances

What do we want in a MIP formulation?

1 **Strength** How tight is the LP relaxation?

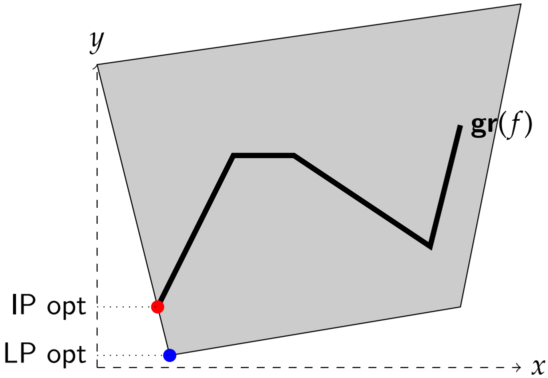
✗ Not sharp = bad bounds from LP



What do we want in a MIP formulation?

1 **Strength** How tight is the LP relaxation?

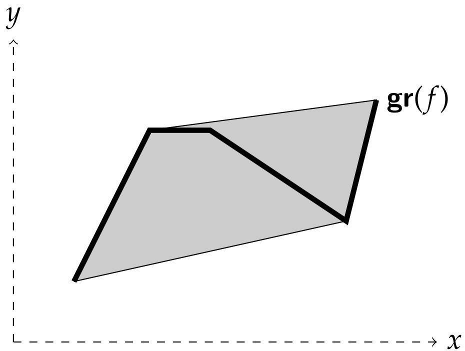
✗ Not sharp = bad bounds from LP



What do we want in a MIP formulation?

1 **Strength** How tight is the LP relaxation?

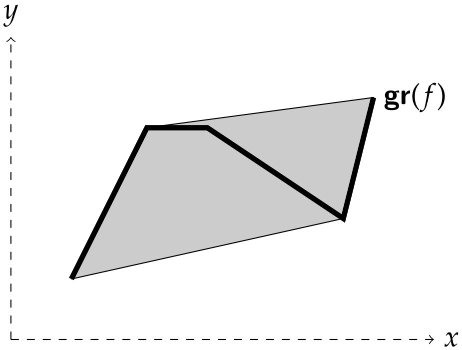
✓ **Sharp** = good bounds from LP



What do we want in a MIP formulation?

1 **Strength** How tight is the LP relaxation?

✓ **Sharp** = good bounds from LP

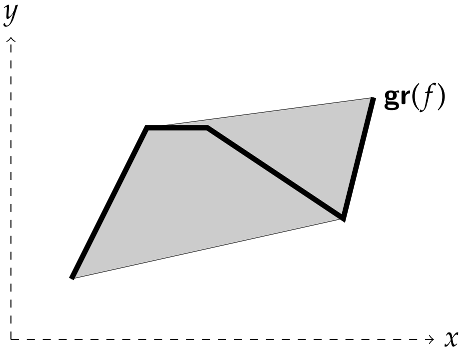


✓ **Ideal** = Sharp + $\mathrm{ext}(Q) \subseteq \mathbb{R}^n \times \mathbb{Z}^r$

What do we want in a MIP formulation?

1 **Strength** How tight is the LP relaxation?

✓ **Sharp** = good bounds from LP



✓ **Ideal** = Sharp + $\text{ext}(Q) \subseteq \mathbb{R}^n \times \mathbb{Z}^r$
= strongest possible relaxation!

What do we want in a MIP formulation?

2 **Size** How many additional variables and constraints?

$$x \in \bigcup_{i=1}^{d} S^i \iff \text{exists } z \in \mathbb{Z}^r \text{ such that } (x, z) \in Q$$

What do we want in a MIP formulation?

2 **Size** How many additional variables and constraints?

$$x \in \bigcup_{i=1}^{d} S^i \iff \text{ exists } z \in \mathbb{Z}^r \text{ such that } (x, z) \in Q$$

$$Q = \left\{ (x, z) \;\middle|\; A \begin{pmatrix} x \\ z \end{pmatrix} \leq b \right\}, \text{ where } A \in \mathbb{R}^{m \times (n+r)}$$

What do we want in a MIP formulation?

2 **Size** How many additional variables and constraints?

$$x \in \bigcup_{i=1}^{d} S^i \iff \text{ exists } z \in \mathbb{Z}^r \text{ such that } (x,z) \in Q$$

$$Q = \left\{ (x,z) \;\middle|\; A \begin{pmatrix} x \\ z \end{pmatrix} \leq b \right\}, \text{ where } A \in \mathbb{R}^{m \times (n+r)}$$

- How big is...
  - r? (# of integer variables)
  - m? (# of constraints)

What do we want in a MIP formulation?

\# 2 **Size** How many additional variables and constraints?

$$x \in \bigcup_{i=1}^{d} S^i \iff \text{ exists } z \in \mathbb{Z}^r \text{ such that } (x, z) \in Q$$

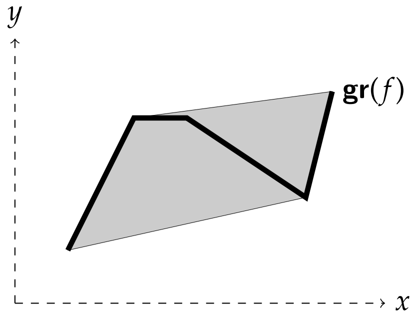
$$Q = \left\{ (x, z) \;\middle|\; A \begin{pmatrix} x \\ z \end{pmatrix} \leq b \right\}, \text{ where } A \in \mathbb{R}^{m \times (n+r)}$$

- How big is...
  - r? (# of integer variables)
  - m? (# of constraints)
- The smaller m^* and r, the quicker to optimize over LP relaxation

*(We really only care about general inequality constraints, we get variable bounds, e.g. $x \geq 0$, for free)

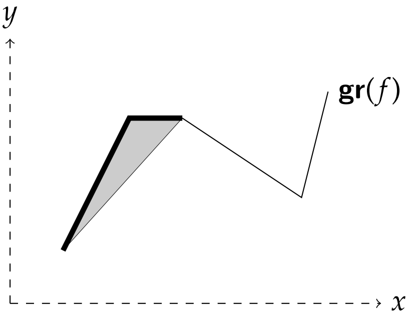
What do we want in a MIP formulation?

3 **Branching** How does formulation change in branch-and-bound?

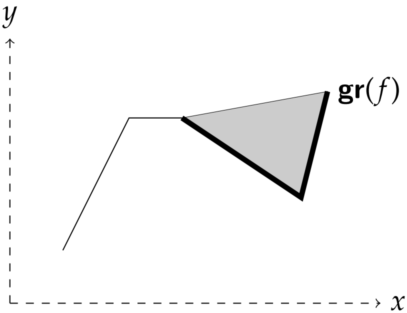


What do we want in a MIP formulation?

3 **Branching** How does formulation change in branch-and-bound?



$z_1 \leq 0$

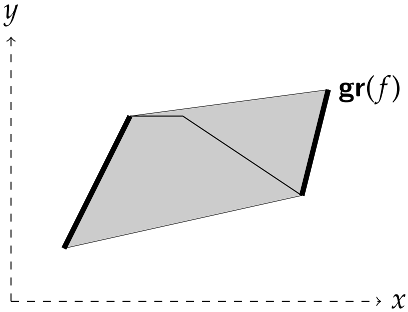


$z_1 \geq 1$

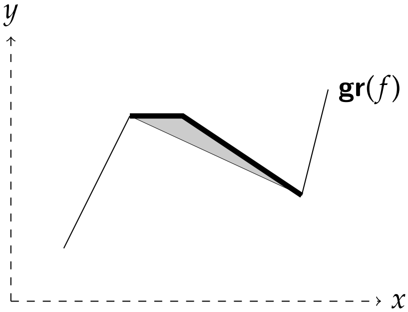
Branching with Formulation A

What do we want in a MIP formulation?

3 **Branching** How does formulation change in branch-and-bound?



$z_1 \leq 0$



$z_1 \geq 1$

Branching with Formulation B

How can we build MIP formulations?

Approach #1: Ad-hoc formulations

Ad-hoc formulations for trained neural networks

- Just reason it out by hand!

Ad-hoc formulations for trained neural networks

- Just reason it out by hand!
- Simple example:

$$\mathtt{MAX} = \left\{ (x, y) \in \mathbb{R}^d \times \mathbb{R} \,\middle|\, L \leq x \leq U,\ y = \max\{0, w \cdot x + b\} \right\}$$

Ad-hoc formulations for trained neural networks

- Just reason it out by hand!
- Simple example:

$$\texttt{MAX} = \left\{ (x, y) \in \mathbb{R}^d \times \mathbb{R} \;\middle|\; L \leq x \leq U,\ y = \max\{0, w \cdot x + b\} \right\}$$

- $\texttt{MAX} \equiv$ ReLu activation unit in trained neural network

Ad-hoc formulations for trained neural networks

- Just reason it out by hand!
- Simple example:

$$\texttt{MAX} = \left\{ (x, y) \in \mathbb{R}^d \times \mathbb{R} \,\middle|\, L \leq x \leq U,\ y = \max\{0, w \cdot x + b\} \right\}$$

- $\texttt{MAX} \equiv$ ReLu activation unit in trained neural network
- Big-M formulation:

$$\begin{gathered} y + L(1 - z) \leq w \cdot x + b \leq y \\ y \leq Uz \\ (x, y, z) \in [L, U] \times \mathbb{R}_{\geq 0} \times \{0, 1\} \end{gathered}$$

Ad-hoc formulations for trained neural networks

- Just reason it out by hand!
- Simple example:

$$\texttt{MAX} = \left\{ (x, y) \in \mathbb{R}^d \times \mathbb{R} \,\middle|\, L \le x \le U,\ y = \max\{0, w \cdot x + b\} \right\}$$

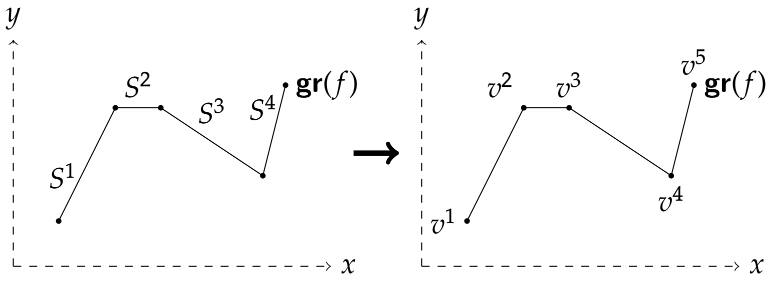
- $\texttt{MAX} \equiv$ ReLu activation unit in trained neural network
- Big-M formulation:

$$\begin{gathered} y + L(1 - z) \le w \cdot x + b \le y \\ y \le Uz \\ (x, y, z) \in [L, U] \times \mathbb{R}_{\ge 0} \times \{0, 1\} \end{gathered}$$

- Not ideal or sharp

Approach #2: Combinatorial construction framework

Univariate piecewise linear functions

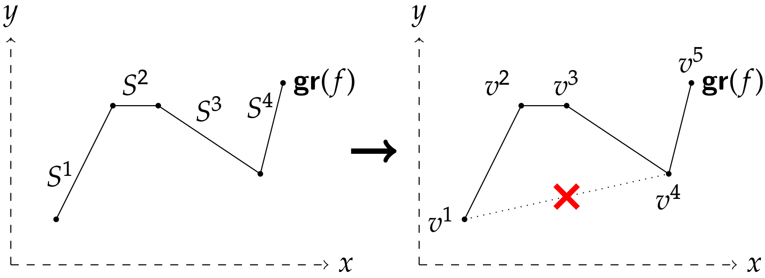


- Introduce λ_i variable for each breakpoint v^i

$$(x, y) \in \mathbf{gr}(f) \Longleftrightarrow (x, y) = \sum_{i=1}^{d+1} v^i \lambda_i \text{ and } \lambda \text{ is SOS2}$$

- λ is SOS2 if: [Beale 1970, 1976]
  1. they are convex multipliers ($\lambda \in \Delta^{d+1}$ = unit simplex)
  2. $\text{support}(\lambda) \subseteq \{j, j+1\}$ for some j

Univariate piecewise linear functions

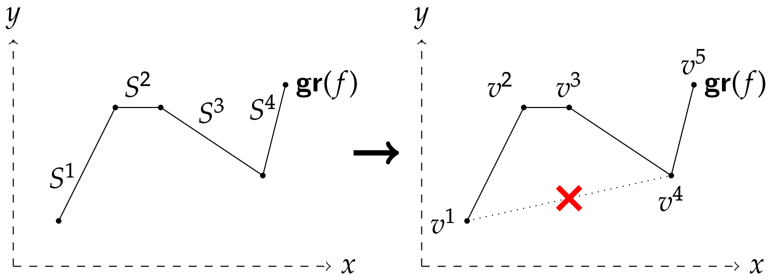


- Introduce λ_i variable for each breakpoint v^i

$$(x,y) \in \mathbf{gr}(f) \iff (x,y) = \sum_{i=1}^{d+1} v^i \lambda_i \text{ and } \lambda \text{ is SOS2}$$

- λ is SOS2 if: [Beale 1970, 1976]
  1. they are convex multipliers ($\lambda \in \Delta^{d+1}$ = unit simplex)
  2. $\text{support}(\lambda) \subseteq \{j, j+1\}$ for some j

Univariate piecewise linear functions



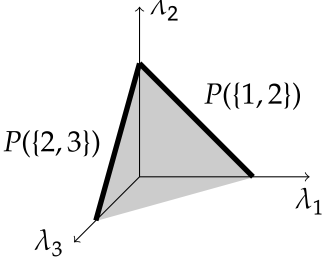
- Introduce λ_i variable for each breakpoint v^i

$$(x, y) \in \bigcup_{i=1}^{d} S^i \Longleftrightarrow (x, y) = \sum_{i=1}^{d+1} v^i \lambda_i \text{ and } \lambda \in \bigcup_{i=1}^{d} P(\{i, i+1\})$$

- $P(T) = \{\lambda \in \Delta^{d+1} : \text{support}(\lambda) \subseteq T\}$ (*face of the simplex*)

The SOS2 constraint

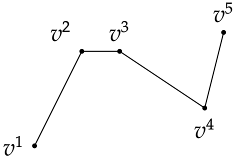
$$\lambda \in \bigcup_{i=1}^{d} P(\{i, i+1\})$$



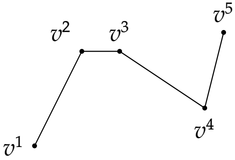
1. Strip away problem data (values of v^i)
2. Formulate the SOS2 constraint on λ over the unit simplex Δ^{d+1}
3. Apply linear transformation $(x, y) = \sum_{i=1}^{d+1} v^i \lambda_i$

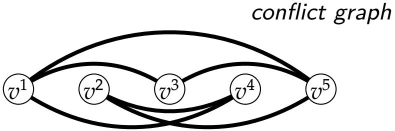
$$P(T) = \{\lambda \in \Delta^{d+1} : \text{support}(\lambda) \subseteq T\} (\textit{face of the simplex})$$

A combinatorial way to build formulations

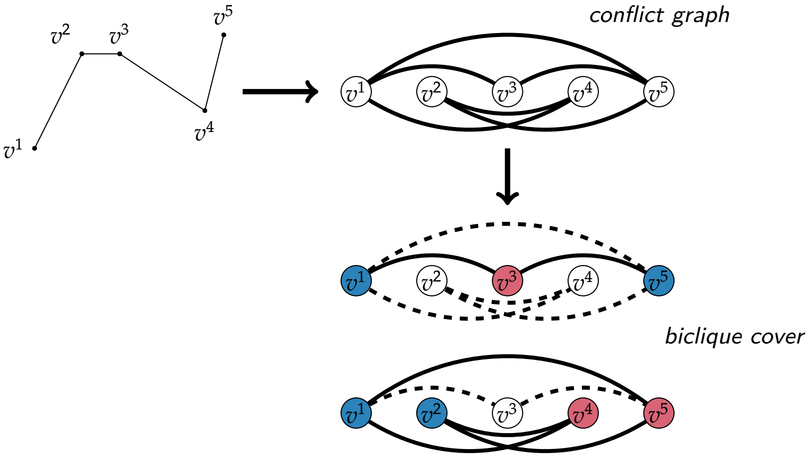


A combinatorial way to build formulations

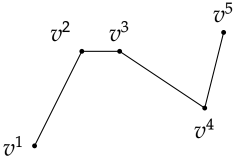


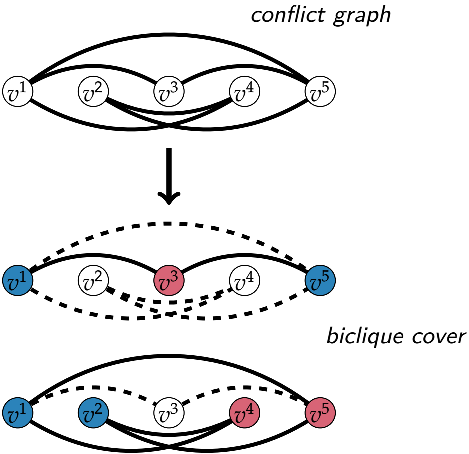


A combinatorial way to build formulations



A combinatorial way to build formulations





$$\lambda_1 + \lambda_5 \leq 1 - z_1$$
$$\lambda_3 \leq z_1$$
$$\lambda_1 + \lambda_2 \leq 1 - z_2$$
$$\lambda_4 + \lambda_5 \leq z_2$$
$$(\lambda, z) \in \Delta^5 \times \{0,1\}^2$$

Independent branching formulations

- *Conflict graph*: $\mathscr{G}^c = ([n], E)$, where

$$E = \left\{ \{u, v\} \in [n]^2 : \{u, v\} \not\subseteq T^i \text{ for each } i \right\}$$

- *Biclique cover* for $\mathscr{G}^c$: $\{(A^j, B^j)\}_{j=1}^t$ where $E = \bigcup_{j=1}^t (A^j \times B^j)$

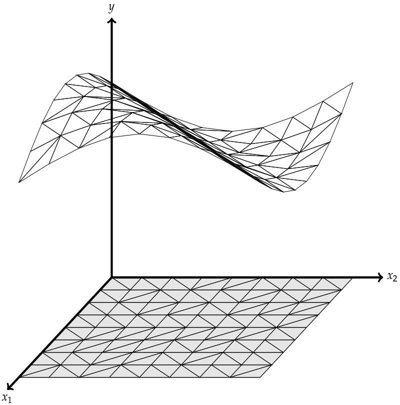
Theorem (H. and Vielma 2016)

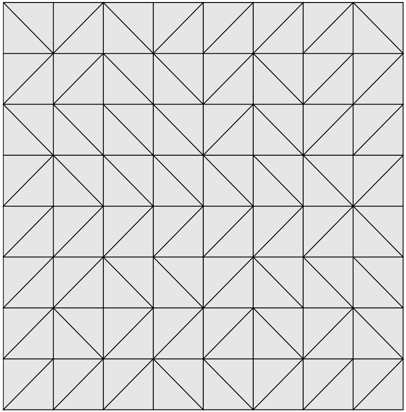
If an independent branching formulation exists* for $\bigcup_{i=1}^d P(T^i)$, then

$$\sum_{v \in A^j} \lambda_v \leq z_j, \quad \sum_{v \in B^j} \lambda_v \leq 1 - z_j, \quad z_j \in \{0, 1\} \quad \forall j \in [t]$$

is an ideal formulation for $\bigcup_{i=1}^d P(T^i)$ if and only if $\{(A^j, B^j)\}_{j=1}^t$ is a biclique cover for $\mathscr{G}^c$.

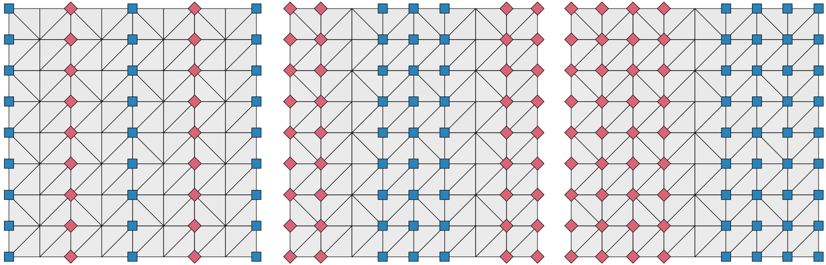
Bivariate piecewise linear functions





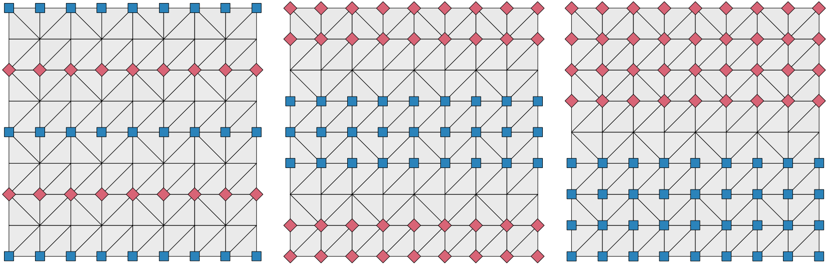
`Stencil` formulation for bivariate functions

- Aggregated SOS2 along x direction
- Separated edges between vertices that are "far apart" in x direction
- Needs $\lceil \log_2(\text{\# breakpoints in } x \text{ direction}) \rceil$ levels (variables)



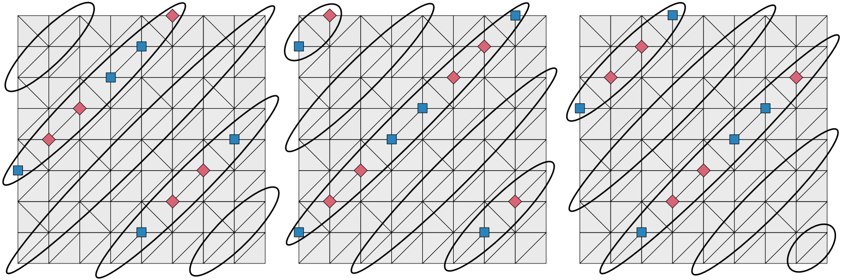
Stencil formulation for bivariate functions

- Aggregated SOS2 along y direction
- Separated edges between vertices that are "far apart" in y direction
- Needs $\lceil \log_2(\text{\# breakpoints in } y \text{ direction}) \rceil$ levels (variables)



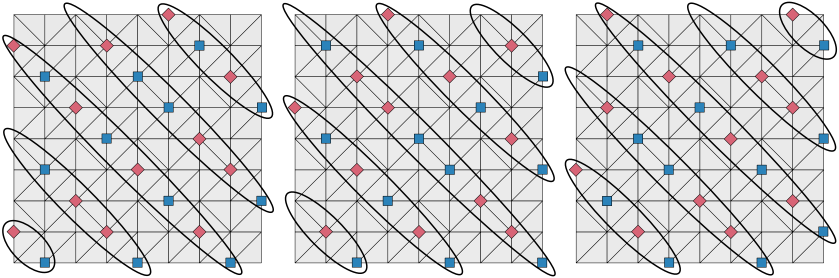
Stencil formulation for bivariate functions

- Separate all edges along diagonal lines
- Can aggregate diagonal lines that are "far apart"
- Needs 3 levels (variables)



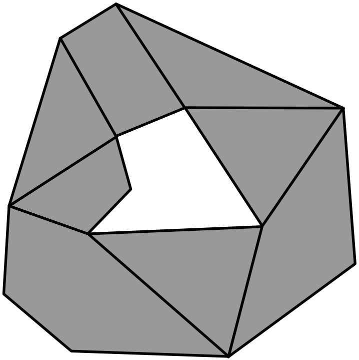
Stencil formulation for bivariate functions

- Separate all edges along anti-diagonal lines
- Can aggregate anti-diagonal lines that are "far apart"
- Needs 3 levels (variables)



A combinatorial way to build formulations

- How do we do this automatically?
- Especially important for more unstructured constraints:

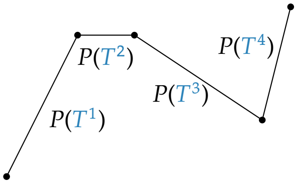


A combinatorial way to build formulations

- How do we do this automatically?
- Simple MIP formulation for minimum biclique cover
- Implemented in `PiecewiseLinearOpt.jl` to make stencil formulation “smaller”
- Unfortunately, it doesn’t scale
- **Wishlist:**
  1. Practically efficient algorithm for minimum biclique cover...
  2. ...and an implementation in Julia

Approach #3: Geometric construction framework

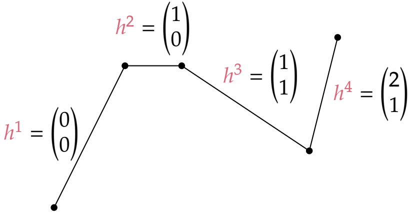
The embedding approach



Two ingredients:

1. The sets $\mathscr{T} = (T^i \subseteq [n])_{i=1}^{d}$ (correspond to faces of simplex; not in (x, z)-space!)

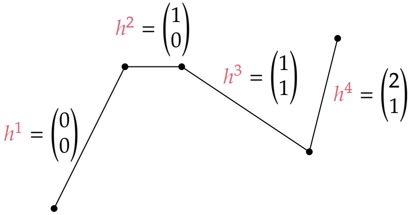
The embedding approach



Two ingredients:

1. The sets $\mathscr{T} = (T^i \subseteq [n])_{i=1}^d$ (correspond to faces of simplex; not in (x, z)-space!)
2. Unique *codes* $H = (h^i)_{i=1}^d \subset \mathbb{R}^r$ (also hole-free, in convex position)

The embedding approach



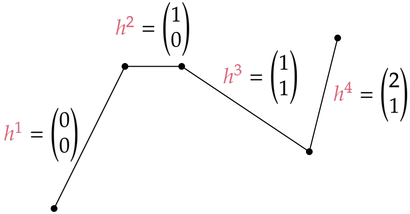
Two ingredients:

1. The sets $\mathscr{T} = (T^i \subseteq [n])_{i=1}^d$ (correspond to faces of simplex; not in (x, z)-space!)
2. Unique *codes* $H = (h^i)_{i=1}^d \subset \mathbb{R}^r$ (also hole-free, in convex position)

Build *embedding*:

$$\mathrm{Em}(\mathscr{T}, H) = \begin{pmatrix} P(T^1) \\ h^1 \end{pmatrix} \cup \begin{pmatrix} P(T^2) \\ h^2 \end{pmatrix} \cup \cdots \cup \begin{pmatrix} P(T^d) \\ h^d \end{pmatrix}$$

The embedding approach



Two ingredients:

1. The sets $\mathscr{T} = (T^i \subseteq [n])_{i=1}^d$ (correspond to faces of simplex; not in (x,z)-space!)
2. Unique *codes* $H = (h^i)_{i=1}^d \subset \mathbb{R}^r$ (also hole-free, in convex position)

Proposition (Vielma 2017)

$\mathrm{Conv}(\mathrm{Em}(\mathscr{T}, H))$ is an ideal formulation. Conversely, any non-extended ideal formulation implies the existence of some corresponding $\mathscr{T}$ and H.

Geometric formulation construction

Theorem (H. and Vielma 2017a)

If $\mathscr{T}$ is path connected and H is in convex position, then $\mathrm{Conv}(\mathrm{Em}(\mathscr{T}, H))$ is

$$\sum_{v=1}^{n} \min_{s: v \in T^s} \{b \cdot h^s\} \lambda_v \leq b \cdot z \leq \sum_{v=1}^{n} \max_{s: v \in T^s} \{b \cdot h^s\} \lambda_v \quad \forall b \in B$$

$$(\lambda, z) \in \Delta^n \times \mathrm{aff}(H),$$

where B contains normal directions to all hyperplanes spanned by $C = \left\{ h^j - h^i : T^i \cap T^j \neq \emptyset \right\}$ in $\mathrm{span}(C)$.

Geometric formulation construction

Theorem (H. and Vielma 2017a)

If $\mathscr{T}$ is path connected and H is in convex position, then $\mathrm{Conv}(\mathrm{Em}(\mathscr{T}, H))$ is

$$\sum_{v=1}^{n} \min_{s:v \in T^s} \{b \cdot h^s\} \lambda_v \leq b \cdot z \leq \sum_{v=1}^{n} \max_{s:v \in T^s} \{b \cdot h^s\} \lambda_v \quad \forall b \in B$$

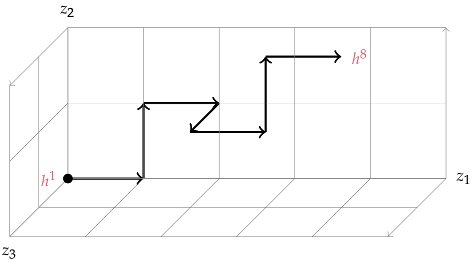
$$(\lambda, z) \in \Delta^n \times \mathrm{aff}(H),$$

where B contains normal directions to all hyperplanes spanned by $C = \left\{h^j - h^i : T^i \cap T^j \neq \emptyset\right\}$ in $\mathrm{span}(C)$.

Crucial points:

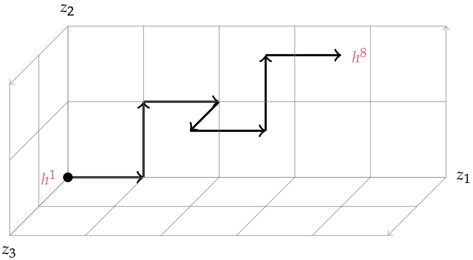
1. # variables = # of components of codes in H
2. # constraints = 2 ×(# hyperplanes)

Geometric formulation construction



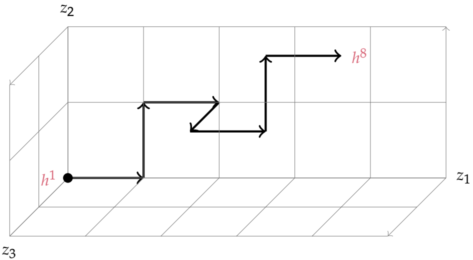
1. Ambient space $\mathbb{R}^{\log_2(d)} \Longrightarrow \log_2(d)$ variables

Geometric formulation construction



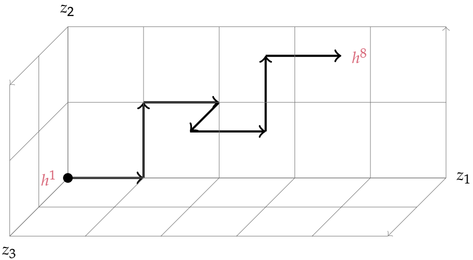
$$C = \left\{ h^j - h^i : T^i \cap T^j \neq \emptyset \right\}$$

Geometric formulation construction



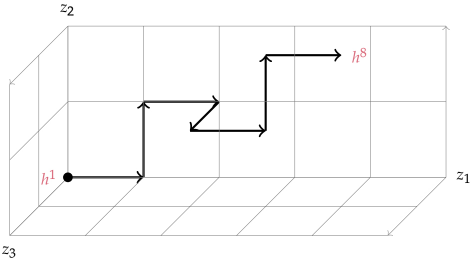
$$C = \left\{ h^{i+1} - h^{i} \right\}_{i=1}^{d-1}$$

Geometric formulation construction



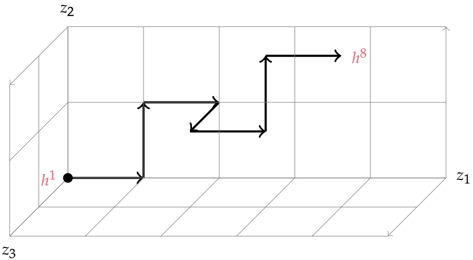
$$C = \left\{ \mathbf{e}^i \right\}_{i=1}^{\log_2(d)}$$

Geometric formulation construction



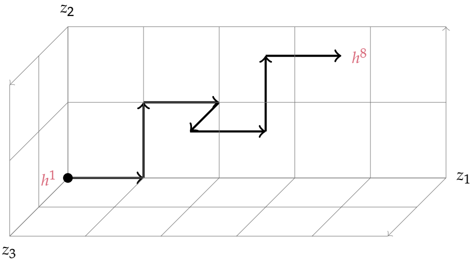
$B =$ normal directions to hyperplanes spanned by C

Geometric formulation construction



$$B = \left\{\mathbf{e}^i\right\}_{i=1}^{\log_2(d)}$$

Geometric formulation construction



$$B = \left\{\mathbf{e}^i\right\}_{i=1}^{\log_2(d)}$$

2. directions in C are axis-aligned $\implies 2\log_2(d)$ constraints

Interlude: Modeling tools

Here's the math ($d = 8$):

$$
\begin{aligned}
\min_{x} \quad & \sum_{i \in S} \sum_{j \in D} f_{i,j}(x_{i,j}) \\
\text{s.t.} \quad & \sum_{j \in D} x_{i,j} = s_i && \forall i \in S \\
& \sum_{i \in S} x_{i,j} = d_j && \forall j \in D \\
& x_{i,j} \geq 0 && \forall i \in S, j \in D
\end{aligned}
$$

Interlude: Modeling tools

Here's the math ($d = 8$):

$$
\begin{array}{lll}
\min\limits_{x\geq 0} & \sum_{i\in S}\sum_{j\in D} z_{i,j} & \\
\text{s.t.} & \sum_{j\in D} x_{i,j} = s_i \quad \forall i \in S & \\
& \sum_{i\in S} x_{i,j} = d_j \quad \forall j \in D & \\
& (x_{i,j}, z_{i,j}) = \sum_{k=1}^{N+1} v_{i,j}^k \lambda_k^{i,j} & \forall i \in S, j \in D \\
& \lambda_3^{i,j} + \lambda_4^{i,j} + 2\lambda_5^{i,j} + 2\lambda_6^{i,j} + 3\lambda_7^{i,j} + 3\lambda_8^{i,j} + 4\lambda_9^{i,j} \leq z_1^{i,j} & \forall i \in S, j \in D \\
& \lambda_2^{i,j} + \lambda_3^{i,j} + 2\lambda_4^{i,j} + 2\lambda_5^{i,j} + 3\lambda_6^{i,j} + 3\lambda_7^{i,j} + 4\lambda_8^{i,j} + 4\lambda_9^{i,j} \geq z_1^{i,j} & \forall i \in S, j \in D \\
& \lambda_4^{i,j} + \lambda_5^{i,j} + \lambda_6^{i,j} + \lambda_7^{i,j} + 2\lambda_8^{i,j} + 2\lambda_9^{i,j} \leq z_2^{i,j} & \forall i \in S, j \in D \\
& \lambda_3^{i,j} + \lambda_4^{i,j} + \lambda_5^{i,j} + \lambda_6^{i,j} + 2\lambda_7^{i,j} + 2\lambda_8^{i,j} + 2\lambda_9^{i,j} \geq z_2^{i,j} & \forall i \in S, j \in D \\
& \lambda_6^{i,j} + \lambda_7^{i,j} + \lambda_8^{i,j} + \lambda_9^{i,j} \leq z_3^{i,j} \leq \lambda_5^{i,j} + \lambda_6^{i,j} + \lambda_7^{i,j} + \lambda_8^{i,j} + \lambda_9^{i,j} & \forall i \in S, j \in D \\
& (\lambda^{i,j}, z^{i,j}) \in \Delta^9 \times \{0,1,2,3,4\} \times \{0,1,2\} \times \{0,1\} & \forall i \in S, j \in D
\end{array}
$$

Now turn this into code.

Interlude: Modeling tools

```
using JuMP, PiecewiseLinearOpt
model = Model()
@variable(model, x[i in S, j in D] >= 0)
for j in D
    @constraint(model, sum(x[i,j] for i in S) == d[j])
end
for i in S
    @constraint(model, sum(x[i,j] for j in D) == s[i])
end
for i in S, j in D
    z[i,j] = piecewiselinear(model, x[i,j], t[i,j],
     ↪  f[i,j], method=:ZigZag)
end
@objective(model, Min, sum(z))
solve(model)
```

Building ideal formulations computationally

- **Wishlist:**
  1. Practically efficient algorithm for spanning hyperplanes...
  2. ...and a Julia implementation

Proposition (Vielma 2017)

$\mathrm{Conv}(\mathrm{Em}(\mathscr{T}, H))$ is an ideal formulation. Conversely, any non-extended ideal formulation implies the existence of some corresponding $\mathscr{T}$ and H.

- **Key point:** Compute convex hull for an ideal formulation!
- Instead of computing spanning hyperplanes directly...use Julia!

Building ideal formulations computationally

- **Tower puzzle** (Juan Pablo Vielma and Austin Herrling): place integers on rectangular grid, subject to "vision number" constraints
- Which formulation for "vision number" constraints? Compute it!

```
using CDDLib, Polyhedra
...
vertices = compute_vision_numbers(idx)
points = SimpleVRepresentation(vertices)
poly = polyhedron(points, CDDLibrary(:exact))
removehredundancy!(poly)
ineq = SimpleHRepresentation(poly) #ineq.A, ineq.b
...
```

Building intuition with computational tools

- What if I want a generic ideal formulation? Compute examples!
- Generate some data and turn this...

```
m = Model()
@variable(m, l[i] <= x[i=1:d] <= u[i])
@variable(m, y >= 0)
@variable(m, z0 >= 0)
@variable(m, z1 >= 0)
@variable(m, x0[1:d])
@variable(m, y0)
@variable(m, x1[1:d])
@variable(m, y1 >= 0)
@constraint(m, [i=1:d], x[i] == x0[i] + x1[i])
@constraint(m, y == y0 + y1)
@constraint(m, 1 == z0 + z1)
@constraint(m, y0 == 0)
@constraint(m, dot(w,x0) + b <= 0)
@constraint(m, [i=1:d], x0[i] >= l[i]*z1)
@constraint(m, [i=1:d], x0[i] <= u[i]*z1)
@constraint(m, y1 == dot(w,x1) + b)
@constraint(m, [i=1:d], x1[i] >= l[i]*z0)
@constraint(m, [i=1:d], x1[i] <= u[i]*z0)
poly = polyhedron(m, CDDLibrary(:exact))
P   = eliminate(poly, [eliminate_vars;])
removehredundancy!(P)
```

Building intuition with computational tools

- What if I want a generic ideal formulation? Compute examples!
- ...into this...

```
-1 x_1 + 0 x_2 + 0 x_3 + -1 y + -39 z <= 4
-1 x_1 + 2 x_2 + 0 x_3 + -1 y + -9 z <= 20
1 x_1 + -2 x_2 + 3 x_3 + 1 y + 50 z <= 51
1 x_1 + -2 x_2 + 0 x_3 + 1 y + -7 z <= 21
0 x_1 + -2 x_2 + 3 x_3 + 1 y + 39 z <= 45
0 x_1 + -2 x_2 + 0 x_3 + 1 y + -18 z <= 15
1 x_1 + 0 x_2 + 3 x_3 + 1 y + 20 z <= 37
0 x_1 + 0 x_2 + 3 x_3 + 1 y + 9 z <= 31
1 x_1 + 0 x_2 + 0 x_3 + 1 y + -37 z <= 7
0 x_1 + 0 x_2 + 0 x_3 + 1 y + -48 z <= 1
0 x_1 + 2 x_2 + 0 x_3 + -1 y + -20 z <= 15
0 x_1 + 0 x_2 + 0 x_3 + -1 y + -50 z <= -1
1 x_1 + 0 x_2 + 0 x_3 + 0 y + 0 z <= 6
0 x_1 + 0 x_2 + 1 x_3 + 0 y + 0 z <= 10
-1 x_1 + 0 x_2 + 0 x_3 + 0 y + 0 z <= 5
0 x_1 + 1 x_2 + 0 x_3 + 0 y + 0 z <= 8
0 x_1 + -1 x_2 + 0 x_3 + 0 y + 0 z <= 7
0 x_1 + 0 x_2 + 0 x_3 + -1 y + 0 z <= 0
-1 x_1 + 2 x_2 + -3 x_3 + -1 y + 0 z <= -2
```

Building intuition with computational tools

- What if I want a generic ideal formulation? Compute examples!
- ...and then eventually this:

Proposition (Huchette 2018)

An ideal formulation for `MAX` is:

$$y \geq w \cdot x + b$$

$$y \leq \sum_{i \in I} w_i x_i - \sum_{i \in I} w_i L_i (1 - z) + \left(b + \sum_{i \notin I} w_i U_i \right) z \quad \forall I \subseteq [\![d]\!]$$

$$y \geq \sum_{i \in I} w_i x_i - \sum_{i \in I} w_i U_i (1 - z) + \left(b + \sum_{i \notin I} w_i L_i \right) z \quad \forall I \subseteq [\![d]\!]$$

$$(x, y, z) \in [L, U] \times \mathbb{R}_{\geq 0} \times \{0, 1\}.$$

Conclusion

- Choice of formulation can greatly affect performance
- Many ways to build different formulations:
  1. Ad-hoc
  2. Combinatorially
  3. Geometrically
  4. Computationally, using Julia
- Wishlist: Efficient algorithm and Julia implementation of:
  - minimum biclique cover
  - spanning hyperplanes of set of directions

Thanks for listening!