Sandia National Laboratories

Exceptional service in the national interest

NEGOTIATING WITH SOLVERS

Using Generalized Disjunctive Programming to find Computationally Performant MIP Formulations

Emma S. Johnson, Soren A. Davis, John D. Siirola

Sandia National Laboratories

MIP Workshop 2025, June 3-6, 2025

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2025-06812C

ACKNOWLEDGMENTS

PROMMIS
Process Optimization and Modeling for Minerals Sustainability

In addition to my coauthors, thanks to Sandia student intern AnaMaria Perez for many of the initial experiments that seeded this work.

This effort was funded by the U.S. Department of Energy's Process Optimization and Modeling for Minerals Sustainability (PrOMMiS) Initiative, supported by the Office of Fossil Energy and Carbon Management's Office of Resource Sustainability. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

NATIONAL ENERGY TECHNOLOGY LABORATORY

BERKELEY LAB

Sandia National Laboratories

Carnegie Mellon

WestVirginiaUniversity.

UNIVERSITY OF NOTRE DAME

Georgia Tech

(GENERALIZED) DISJUNCTIVE PROGRAMMING

- I interpret the "G" to refer to including logical constraints (in addition to algebraic ones) in the formulation.
- In general, then, a GDP has the form:

$$
\begin{aligned}
\min \quad & f(x) && && (1)\\
\text{s.t.} \quad & Ax \leq b && && (2)\\
& g_j(x) \leq 0 && \forall j \in J && (3)\\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ C^{ik}x \leq d^{ik} \\ h_{ik\ell}(x) \leq 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} && \forall k \in K && (4)\\
& \texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) && \forall k \in K && (5)\\
& \Omega(Y, Z) = True && && (6)\\
& Y_{ik} \in \{True, False\} && \forall k \in K, i \in D_k && (7)\\
& Z_n \in \{True, False\} && \forall n \in N && (8)\\
& x \in \mathbb{R} && && (9)
\end{aligned}
$$

THE QUESTION

Is Generalized Disjunctive Programming (GDP) a ***useful*** generalization of MIP?

- My goal is to convince you the answer is 'yes.'
- Why? When I'm wearing my engineering hat:
  1. Automating reformulations from GDP to MIP allows me to computationally evaluate 5-10 different MI(N)LP formulations for the same problem after coding one model.
  2. Despite existing theory, *I cannot predict* which of those formulations will perform best, nor how they will scale.
  3. Each Gurobi release changes which formulations perform well (in absolute and relative terms).

REFORMULATIONS FROM GDP TO MIP

$$
\begin{aligned}
\min \quad & f(x) && && (10)\\
\text{s.t.} \quad & g_j(x) \le 0 && \forall j \in J && (11)\\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik\ell}(x) \le 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} && \forall k \in K && (12)\\
& \texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) && \forall k \in K && (13)\\
& \Omega(Y, Z) = True && && (14)\\
& Y_{ik} \in \{True, False\} && \forall k \in K, i \in D_k && (15)\\
& Z_n \in \{True, False\} && \forall n \in N && (16)\\
& x \in \mathbb{R} && && (17)
\end{aligned}
$$

BIG-M REFORMULATION

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik\ell}(x) \leq 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} \qquad \forall k \in K \qquad (12)$$

$$\texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) \qquad \forall k \in K \qquad (13)$$

$$\Omega(Y, Z) = True \qquad (14)$$

$$Y_{ik} \in \{True, False\} \qquad \forall k \in K, i \in D_k \qquad (15)$$

$$Z_n \in \{True, False\} \qquad \forall n \in N \qquad (16)$$

$$h_{ik\ell}(x) \leq M_{ik\ell}(1 - y_{ik}) \qquad \forall k \in K, i \in D_k, \ell \in \mathcal{L}_{ik} \qquad (18)$$

$$\sum_{i \in D_k} y_{ik} = 1 \qquad \forall k \in K \qquad (19)$$

$$Hy + Gz \leq h \qquad (20)$$

$$y_{ik} \in \{0, 1\} \qquad \forall k \in K, i \in D_k \qquad (21)$$

$$z_n \in \{0, 1\} \qquad \forall n \in N \qquad (22)$$

HULL REFORMULATION

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik\ell}(x) \leq 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} \qquad \forall k \in K \qquad (12)$$

$$\texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) \qquad \forall k \in K \qquad (13)$$

$$\Omega(Y, Z) = True \qquad (14)$$

$$Y_{ik} \in \{True, False\} \qquad \forall k \in K, i \in D_k \qquad (15)$$

$$Z_n \in \{True, False\} \qquad \forall n \in N \qquad (16)$$

$$x = \sum_{i \in D_k} v_{ik} \qquad \forall k \in K \quad (23)$$

$$0 \leq v_{ik} \leq U y_{ik} \qquad \forall k \in K, i \in D_k \quad (24)$$

$$\sum_{i \in D_k} y_{ik} = 1 \qquad \forall k \in K \quad (25)$$

$$y_{ik} h_{ik\ell}(v_{ik}/y_{ik}) \leq 0 \qquad \forall k \in K, i \in D_k, \ell \in \mathcal{L}_{ik} \quad (26)$$

$$Hy + Gz \leq h \qquad (27)$$

$$y_{ik} \in \{0, 1\} \qquad \forall k \in K, i \in D_k \quad (28)$$

$$z_n \in \{0, 1\} \qquad \forall n \in N \quad (29)$$

Furman, Sawaya, and Grossmann, 2020

HULL REFORMULATION

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik\ell}(x) \leq 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} \qquad \forall k \in K \qquad (12)$$

$$\texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) \qquad \forall k \in K \qquad (13)$$

$$\Omega(Y, Z) = True \qquad (14)$$

$$Y_{ik} \in \{True, False\} \qquad \forall k \in K, i \in D_k \qquad (15)$$

$$Z_n \in \{True, False\} \qquad \forall n \in N \qquad (16)$$

$$x = \sum_{i \in D_k} v_{ik} \qquad \forall k \in K \quad (23)$$

$$0 \leq v_{ik} \leq U y_{ik} \qquad \forall k \in K, i \in D_k \quad (24)$$

$$\sum_{i \in D_k} y_{ik} = 1 \qquad \forall k \in K \quad (25)$$

~~$y_{ik} h_{ik\ell}(v_{ik}/y_{ik}) \leq 0 \qquad \forall k \in K, i \in D_k, \ell \in \mathcal{L}_{ik}$ (26)~~

$$Hy + Gz \leq h \qquad (27)$$

$$y_{ik} \in \{0, 1\} \qquad \forall k \in K, i \in D_k \quad (28)$$

$$z_n \in \{0, 1\} \qquad \forall n \in N \quad (29)$$

$$((1-\epsilon)y_{ik} + \epsilon)\, h_{ik\ell}\left(\frac{v_{ik}}{(1-\epsilon)y_{ik} + \epsilon}\right) - \epsilon h_{ik\ell}(0)(1 - y_{ik}) \leq 0 \quad \forall k \in K, i \in D_k, \ell \in \mathcal{L}_{ik} \quad (30)$$

Furman, Sawaya, and Grossmann, 2020

MULTIPLE BIG-M REFORMULATION

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik\ell}(x) \leq 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} \qquad \forall k \in K \qquad (12)$$

$$\texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) \qquad \forall k \in K \qquad (13)$$

$$\Omega(Y, Z) = True \qquad (14)$$

$$Y_{ik} \in \{True, False\} \qquad \forall k \in K, i \in D_k \qquad (15)$$

$$Z_n \in \{True, False\} \qquad \forall n \in N \qquad (16)$$

$$h_{ik\ell}(x) \leq \sum_{i' \in D_k, i' \neq i} M_{ii'k\ell} y_{i'k} \qquad \forall k \in K, i \in D_k, \ell \in \mathcal{L}_{ik} \qquad (31)$$

$$\sum_{i \in D_k} y_{ik} = 1 \qquad \forall k \in K \qquad (32)$$

$$Hy + Gz \leq h \qquad (33)$$

$$y_{ik} \in \{0, 1\} \qquad \forall k \in K, i \in D_k \qquad (34)$$

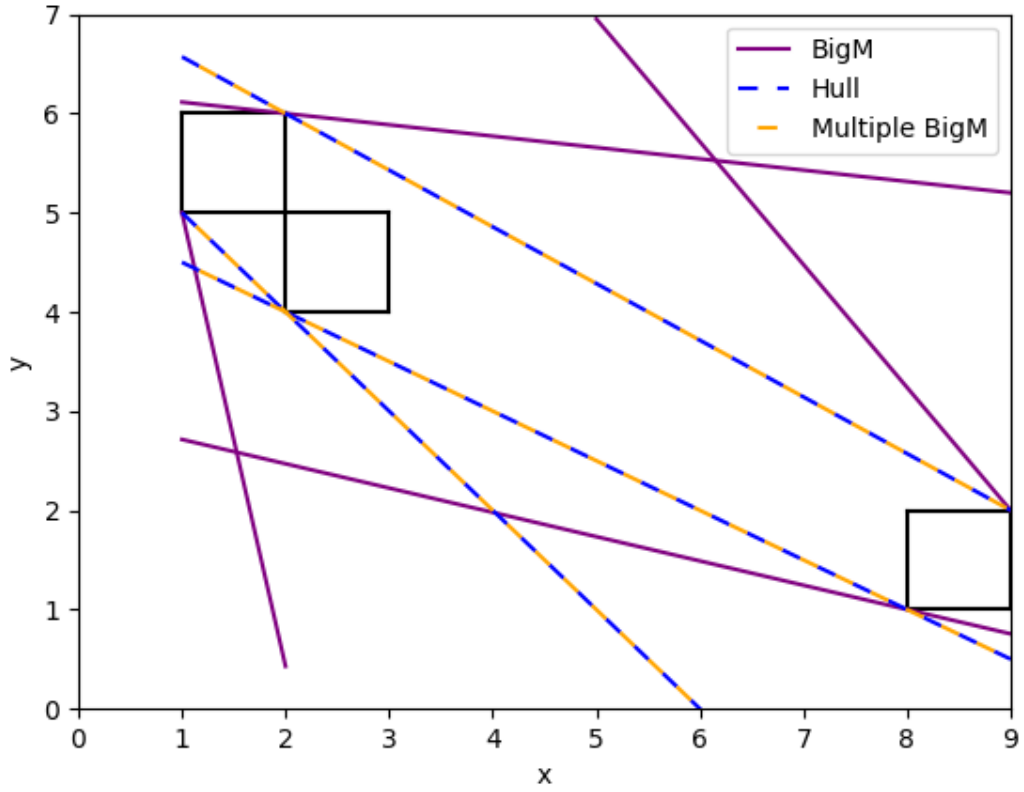
$$z_n \in \{0, 1\} \qquad \forall n \in N \qquad (35)$$

Trespalacios and Grossmann, 2015

REFORMULATIONS FROM GDP TO MIP

Transformation	Advantages	Disadvantages
Big-M	Simple, requires few variables/constraints, familiar structure for solvers	Potentially weak continuous relaxation
Hull	Tighter continuous relaxation	Large model: Requires many variables/constraints
Multiple Big-M	Tighter continuous relaxation with smaller model	Requires calculating quadratically many M values

$$
\begin{aligned}
\min \quad & x + y \\
\text{s.t.} \quad & \begin{bmatrix} Y_1 \\ 1 \le x \le 2 \\ 5 \le y \le 6 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2 \le x \le 3 \\ 4 \le y \le 5 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ 8 \le x \le 9 \\ 1 \le y \le 2 \end{bmatrix} \\
& \texttt{exactly\_one}(Y_1, Y_2, Y_3) \\
& 1 \le x \le 9 \\
& 1 \le y \le 6 \\
& Y_i \in \{\texttt{True}, \texttt{False}\}, \quad \forall i \in \{1, 2, 3\}
\end{aligned}
$$



BINARY MULTIPLICATION REFORMULATION

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik\ell}(x) \leq 0 \quad \forall \ell \in \mathcal{L}_{ik} \end{bmatrix} \qquad \forall k \in K \qquad (12)$$

$$\texttt{exactly\_one}(\{Y_{ik} : i \in D_k\}) \qquad \forall k \in K \qquad (13)$$

$$\Omega(Y, Z) = True \qquad (14)$$

$$Y_{ik} \in \{True, False\} \qquad \forall k \in K, i \in D_k \qquad (15)$$

$$Z_n \in \{True, False\} \qquad \forall n \in N \qquad (16)$$

$$y_{ik} h_{ik\ell}(x) \leq 0 \qquad \forall k \in K, i \in D_k, \ell \in \mathcal{L}_{ik} \qquad (36)$$

$$\sum_{i \in D_k} y_{ik} = 1 \qquad \forall k \in K \qquad (37)$$

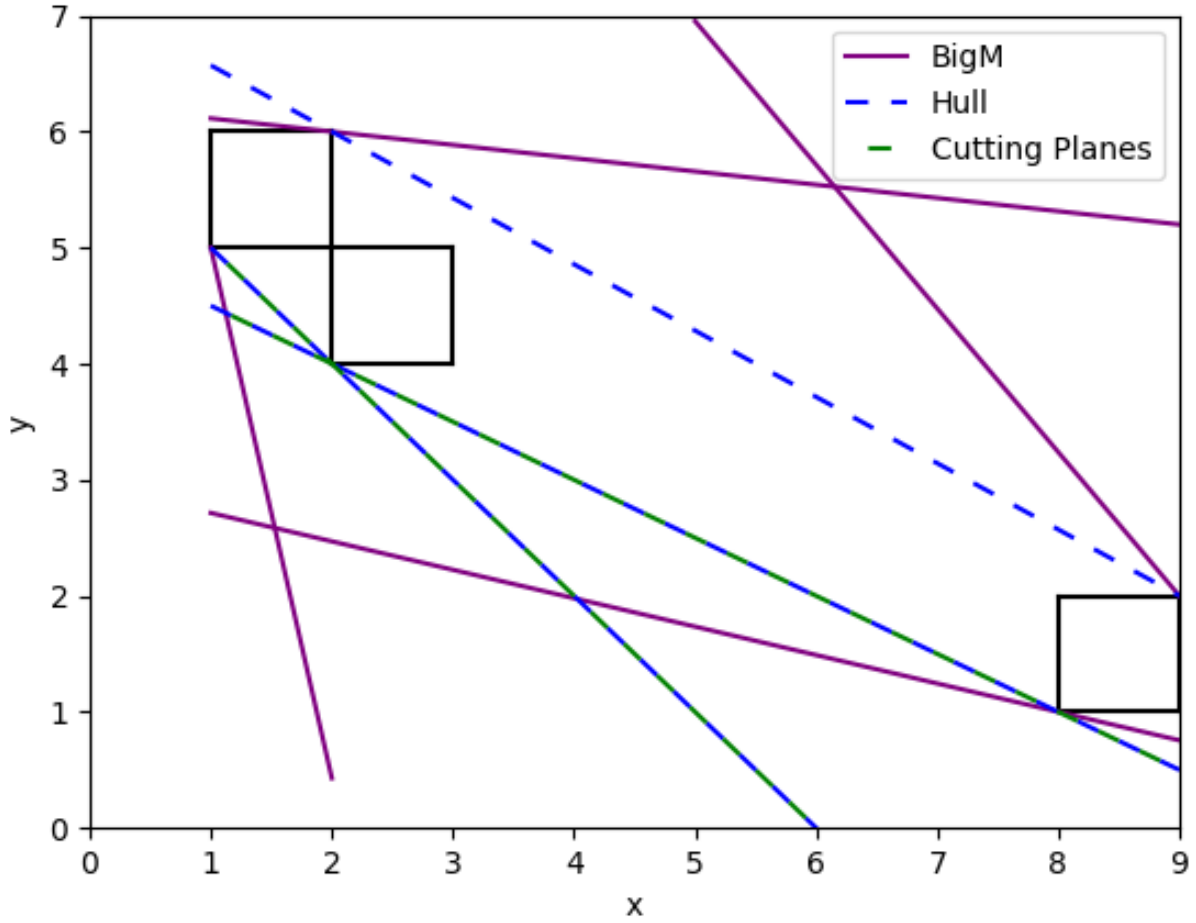
$$Hy + Gz \leq h \qquad (38)$$

$$y_{ik} \in \{0, 1\} \qquad \forall k \in K, i \in D_k \qquad (39)$$

$$z_n \in \{0, 1\} \qquad \forall n \in N \qquad (40)$$

CUTTING PLANES REFORMULATION

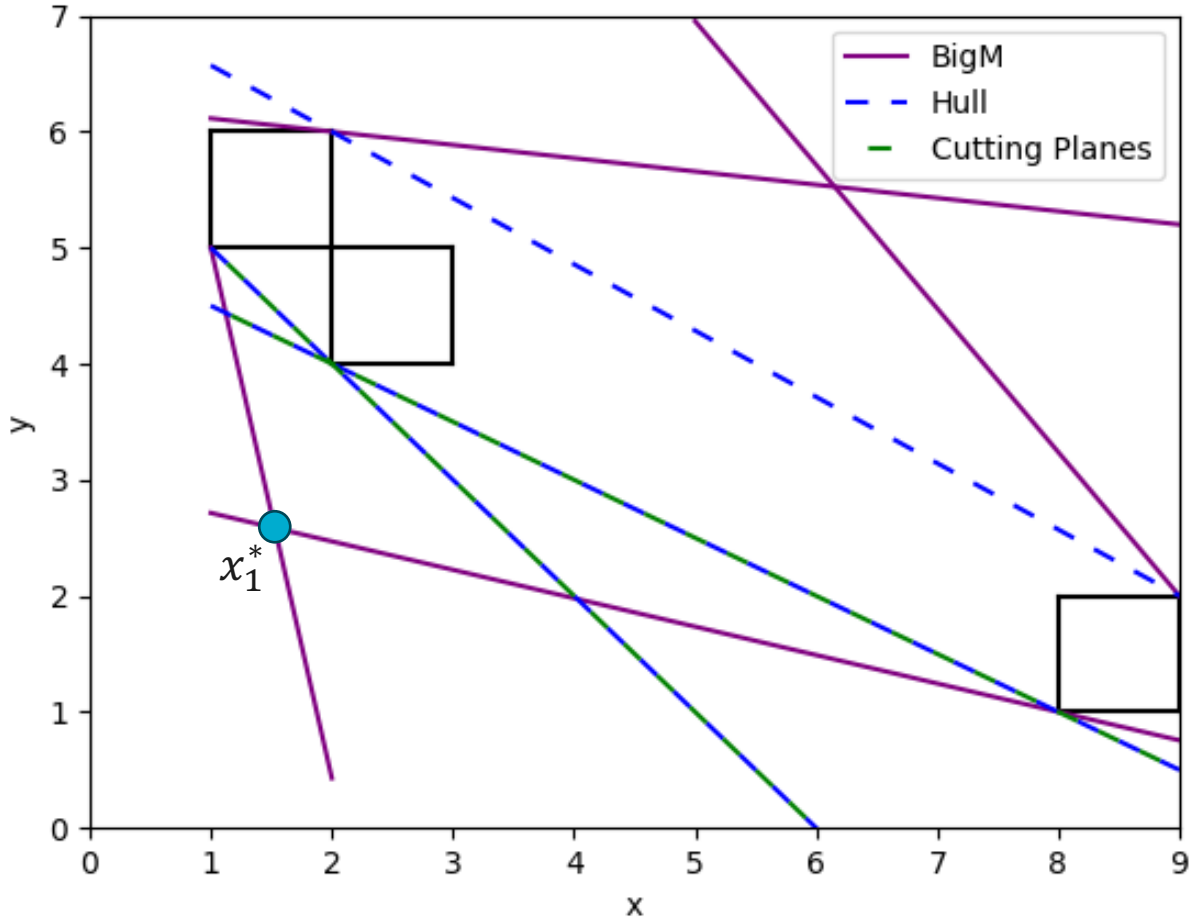
- Begin with LP relaxation of Big-M reformulation, (rBigm) and the LP relaxation of the Hull reformulation (rHull).



Trespalacios and Grossmann, 2016

CUTTING PLANES REFORMULATION

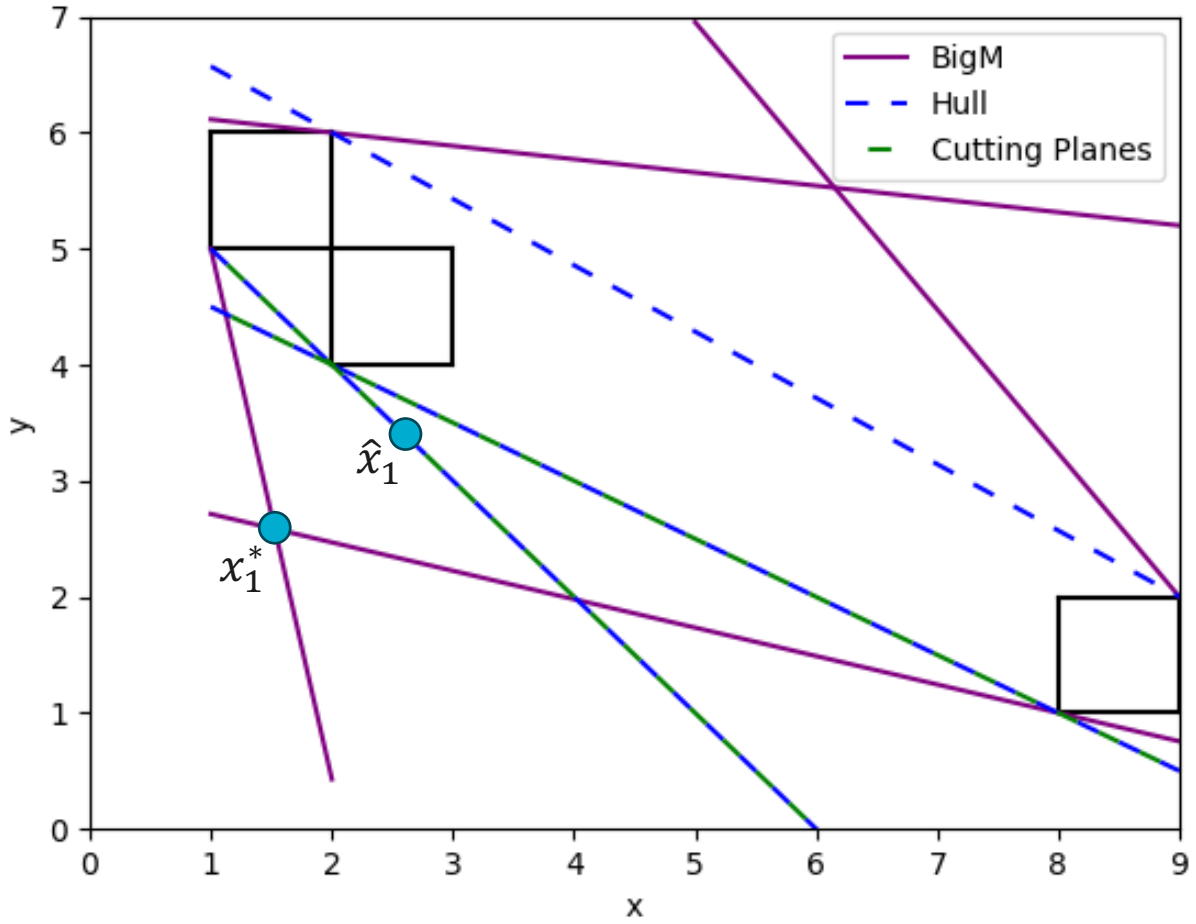
- Begin with LP relaxation of Big-M reformulation, (rBigm) and the LP relaxation of the Hull reformulation (rHull).
- In iteration *i:*
  - Solve (rBigm) for an optimal value x_i^*.



Trespalacios and Grossmann, 2016

CUTTING PLANES REFORMULATION

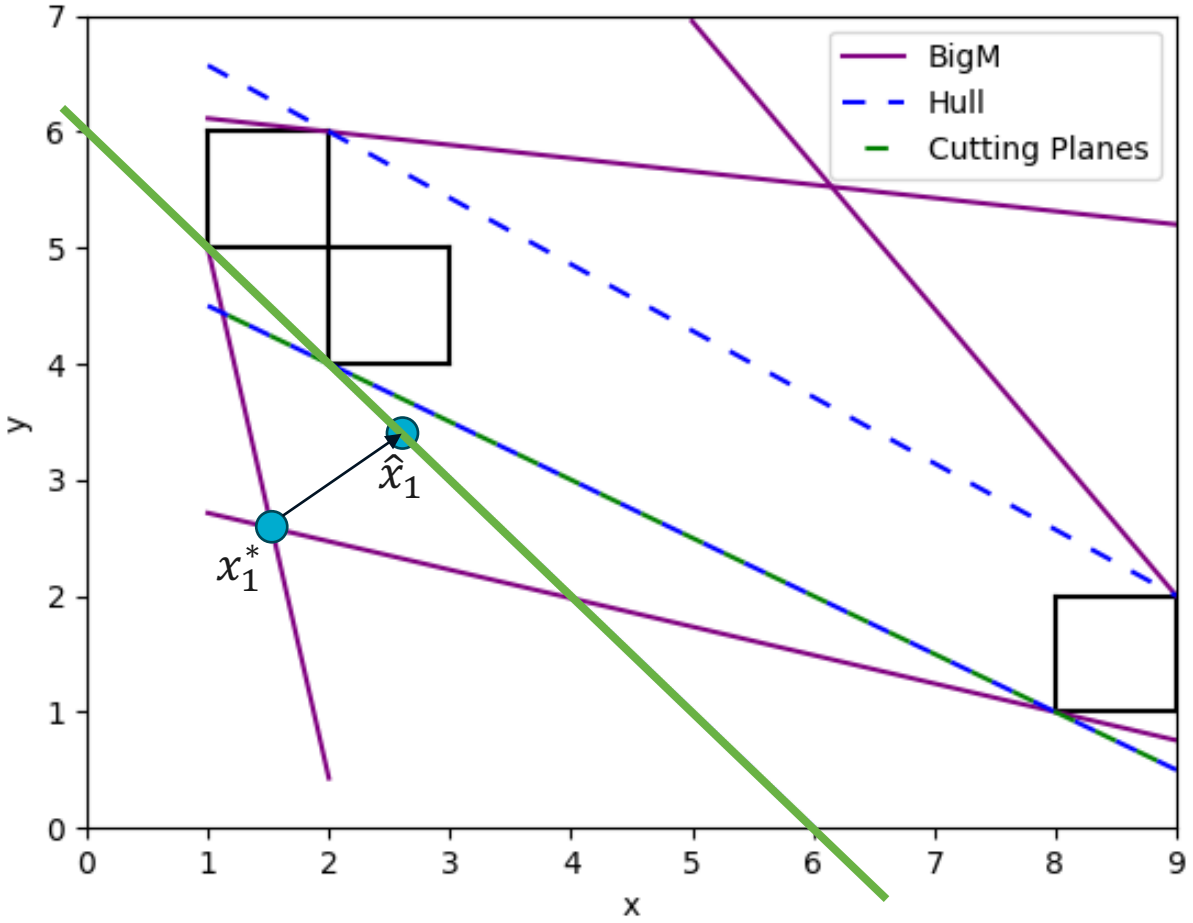
- Begin with LP relaxation of Big-M reformulation, (rBigm) and the LP relaxation of the Hull reformulation (rHull).
- In iteration *i:*
  - Solve (rBigm) for an optimal value x_i^*.
  - Solve a separation problem finding the closest point to x_i^* (according to a norm of your choosing), subject to (rHull). Call this $\hat{x}_i$.



Trespalacios and Grossmann, 2016

CUTTING PLANES REFORMULATION

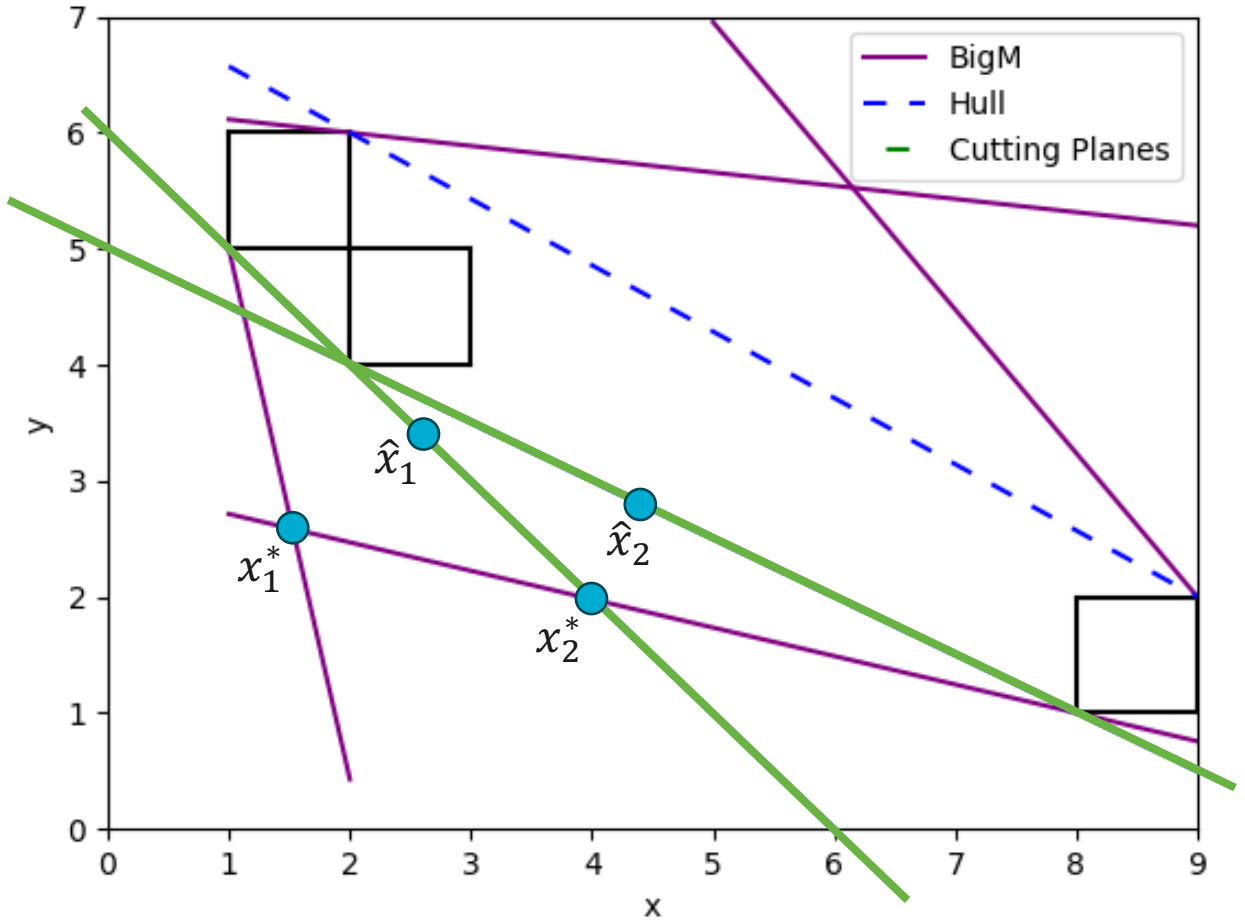
- Begin with LP relaxation of Big-M reformulation, (rBigm) and the LP relaxation of the Hull reformulation (rHull).
- In iteration *i:*
  - Solve (rBigm) for an optimal value x_i^*.
  - Solve a separation problem finding the closest point to x_i^* (according to a norm of your choosing), subject to (rHull). Call this $\hat{x}_i$.
  - Add a cut at $\hat{x}$ perpendicular to $\hat{x}_i - x_i^*$.



Trespalacios and Grossmann, 2016

CUTTING PLANES REFORMULATION

- Begin with LP relaxation of Big-M reformulation, (rBigm) and the LP relaxation of the Hull reformulation (rHull).
- In iteration *i:*
  - Solve (rBigm) for an optimal value x_i^*.
  - Solve a separation problem finding the closest point to x_i^* (according to a norm of your choosing), subject to (rHull). Call this $\hat{x}_i$.
  - Add a cut at $\hat{x}$ perpendicular to $\hat{x}_i - x_i^*$.
  - Stop if the (rBigm) objective value is the same as the previous iteration.



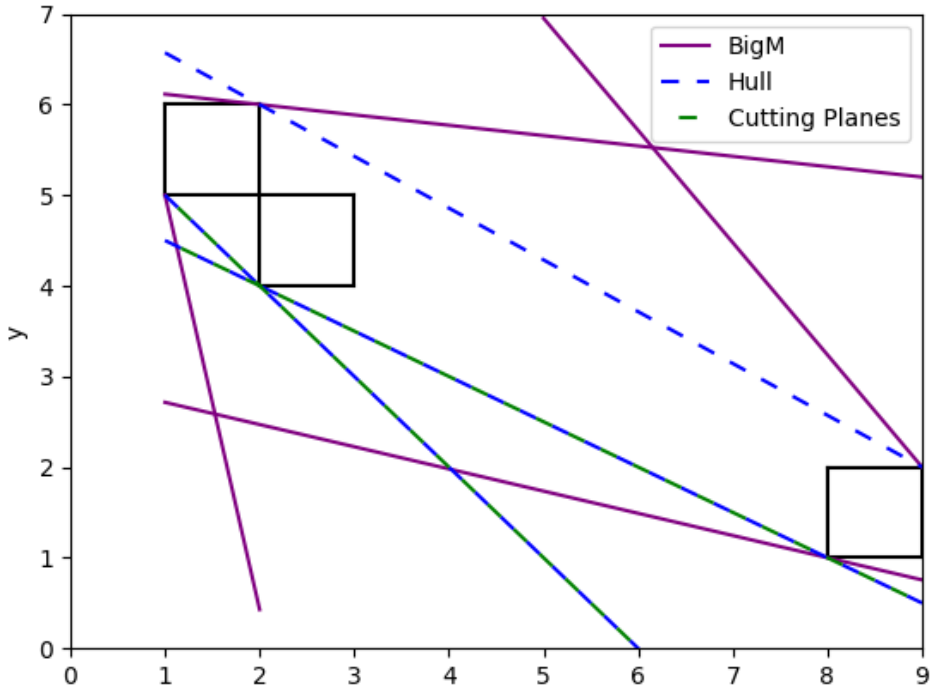
Trespalacios and Grossmann, 2016

REFORMULATIONS FROM GDP TO MI(N)LP

Hybrids of Big-M and Hull

Transformation	Advantages	Disadvantages
Binary Multiplication	Additional structure for solver to exploit	Introduces nonlinearity
Cutting planes	Targeted tightening of Big-M relaxation in direction of improved objective values	Can cause numerical instability
Between Steps	Tighter continuous relaxation with smaller model in cases where there are many more variables than constraints in each disjunct	Relaxation quality is sensitive to choice of variable partition and variable bounds

$$
\begin{aligned}
\min \quad & x + y \\
\text{s.t.} \quad & \begin{bmatrix} Y_1 \\ 1 \le x \le 2 \\ 5 \le y \le 6 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ 2 \le x \le 3 \\ 4 \le y \le 5 \end{bmatrix} \vee \begin{bmatrix} Y_3 \\ 8 \le x \le 9 \\ 1 \le y \le 2 \end{bmatrix} \\
& \texttt{exactly\_one}(Y_1, Y_2, Y_3) \\
& 1 \le x \le 9 \\
& 1 \le y \le 6 \\
& Y_i \in \{\texttt{True}, \texttt{False}\}, \quad \forall i \in \{1, 2, 3\}
\end{aligned}
$$



Kronqvist, Misener, and Tsay, 2021

REFORMULATING LOGICAL CONSTRAINTS

- There are choices here too!
- Pyomo implements two methods:
  - Transformation to conjunctive normal form (core.logical_to_linear)
  - Sparser “factorable programming” inspired transformation that adds many auxiliary variables (contrib.logical_to_disjunctive)

- Anecdotally, which of these we use can matter a lot. But we have very few test cases that actually use logical constraints, so the jury’s really still out here.

TEST CASES: GDPLIB

Problem	# of Vars	# of Disjuncts	# of Disjunctions	# of Linear Constraints	# of Quadratic Constraints	# of General NL Constraints
batch processing	269	18	9	600	0	1
CLay0203	30	55	15	80	60	0
CLay0204	30	55	15	80	60	0
CLay0205	30	55	15	80	60	0
CLay0303	30	55	15	80	60	0
CLay0304	30	55	15	80	60	0
CLay0305	30	55	15	80	60	0
disease	1198	52	26	831	0	0
gdp reactor (cstr)	56	20	10	83	12	5
methanol	277	8	4	374	31	24
mod_hens	138	48	24	158	8	16
modprodnet: Decay	486	2	1	485	0	1
modprodnet: Dip	486	2	1	485	0	1
modprodnet: Growth	486	2	1	485	0	1
positioning	6	50	25	5	25	0
spectralog	68	60	30	150	8	0
Syngas	57618	192	96	14941	0	18
water_network	323	46	23	495	39	9
water_network: quadratic_nonzero_origin	239	10	5	296	28	5

https://github.com/SECQUOIA/gdplib

SO… WHICH IS BEST?

	BigM	Multiple BigM	Hull	Cutting Planes	Binary Multiplication
batch processing		240.4		173.2	
CLay0203	23.0	21.9	48.6	17.5	45.9
CLay0204	26.2	27.5	46.3	18.3	47.5
CLay0205	22.6	28.7	88.7	18.1	20.0
CLay0303	26.3	16.7	34.5	27.9	45.1
CLay0304	25.5	23.7	47.8	17.8	43.2
CLay0305	22.9	26.7	72.5	18.1	45.4
disease	4.8	5.0	5.3	4.7	28.7
gdp reactor (cstr): logical_to_disjunctive	0.9		45.4	1.0	
gdp reactor (cstr): logical_to_linear		254.8	43.6	32.0	
methanol	0.5				
mod_hens	39.5	3.3		29.3	63.3
modprodnet: Decay	4.1	1.0	0.3	5.5	18.6
modprodnet: Dip	5.3	1.1	0.3	7.5	7.9
modprodnet: Growth	8.9	1.3	0.3	2.6	11.4
positioning	1.4	1.2	15.4	1.4	1.7
spectralog	4.1	3.0	4.4	4.2	
Syngas: logical_to_disjunctive			1.8		2.1
Syngas: logical_to_linear		4.3	0.5		0.7
water_network					123.1
water_network: quadratic_nonzero_origin		248.1			

Solve times (s) from Baron:

- Within 1 second of best solve time
- Timeout at 5 minutes
- Error (unsupported transformation or solver error)

SO… WHICH IS BEST?

	BigM	Multiple BigM	Hull	Cutting Planes	Binary Multiplication
batch processing		240.4		173.2	
CLay0203	23.0	21.9	48.6	17.5	45.9
CLay0204	26.2	27.5	46.3	18.3	47.5
modprodnet: Growth	8.9	1.3	0.3	2.6	11.4
positioning	1.4	1.2	15.4	1.4	1.7
spectralog	4.1	3.0	4.4	4.2	
Syngas: logical_to_disjunctive			1.8		2.1
Syngas: logical_to_linear		4.3	0.5		0.7
water_network					123.1
water_network: quadratic_nonzero_origin		248.1			

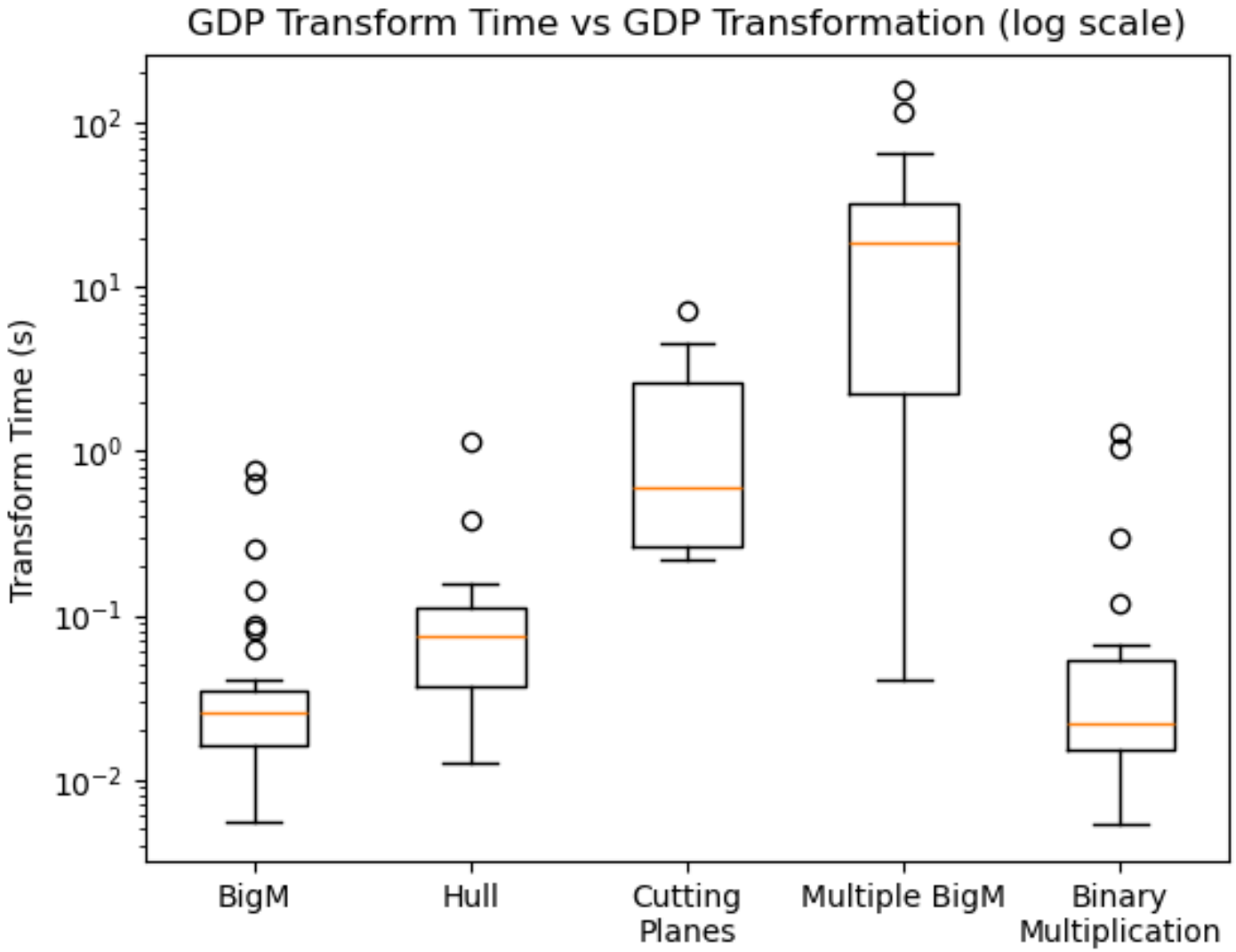
We also ran these instances with Gurobi 12:

- In general these problems are “too easy” for Gurobi, solving in less than 1-2 seconds.
- Of the handful still hard enough to differentiate between transformations, but not so hard that every solve times out, Gurobi usually prefers the same transformation as Baron (with some exceptions).

Solve times (s) from Baron:

- Within 1 second of best solve time
- Timeout at 5 minutes
- Error (unsupported transformation or solver error)

NO FREE LUNCH!



ARE THERE HINTS OF A PATTERN?

Problem	# of Vars	# of Disjuncts	# of Disjunctions	# of Linear Constraints	# of Quadratic Constraints	# of General NL Constraints	Best Transformation
positioning	6	50	25	5	25	0	mBigM
CLay0203	30	55	15	80	60	0	Cutting planes
CLay0204	30	55	15	80	60	0	Cutting planes
CLay0205	30	55	15	80	60	0	Cutting planes
CLay0303	30	55	15	80	60	0	mBigM
CLay0304	30	55	15	80	60	0	Cutting planes
CLay0305	30	55	15	80	60	0	Cutting planes
gdp reactor (cstr)	56	20	10	83	12	5	BigM
spectralog	68	60	30	150	8	0	Hull
mod_hens	138	48	24	158	8	16	mBigM
water_network: quadratic_nonzero_origin	239	10	5	296	28	5	mBigM
batch processing	269	18	9	600	0	1	Cutting planes
methanol	277	8	4	374	31	24	mbigM
water_network	323	46	23	495	39	9	Binary Mult
modprodnet: Decay	486	2	1	485	0	1	mBigM
modprodnet: Dip	486	2	1	485	0	1	mBigM
modprodnet: Growth	486	2	1	485	0	1	Binary Mult
disease	1198	52	26	831	0	0	Cutting planes
Syngas	57618	192	96	14941	0	18	Hull

THE ANSWER CAN CHANGE WITH EVERY GUROBI VERSION

Gurobi version	Transformation(s)	Constraints before presolve	Constraints after presolve	Variables before presolve	Variables after presolve	Solve Time	Nodes Explored
10.0.3	core.logical_to_linear, gdp.bigm	254650	18357	21906	20781	125.2	2655
10.0.3	**core.logical_to_linear, gdp.bound_pretransformation, gdp.bigm**	**34858**	**18242**	**21906**	**20465**	**15.03**	**1362**
10.0.3	contrib.logical_to_disjunctive, gdp.bigm	332749	18349	47939	20781	44.59	2291
10.0.3	contrib.logical_to_disjunctive, gdp.bound_pretransformation, gdp.bigm	112957	18242	47939	20465	35.6	1756
11.0.3	core.logical_to_linear, gdp.bigm	254650	22219	21906	20781	14.1	2543
11.0.3	core.logical_to_linear, gdp.bound_pretransformation, gdp.bigm	34858	18242	21906	20465	29.5	1246
11.0.3	contrib.logical_to_disjunctive, gdp.bigm	332749	21782	47939	20781	34.4	3423
11.0.3	**contrib.logical_to_disjunctive, gdp.bound_pretransformation, gdp.bigm**	**112957**	**18242**	**47939**	**20465**	**12**	**656**
12.0.1	core.logical_to_linear, gdp.bigm	254650	22601	21906	20782	31.8	6229
12.0.1	core.logical_to_linear, gdp.bound_pretransformation, gdp.bigm	34858	18242	21906	20466	27.5	1515
12.0.1	contrib.logical_to_disjunctive, gdp.bigm	332749	21189	47939	20782	28.7	3936
12.0.1	**contrib.logical_to_disjunctive, gdp.bound_pretransformation, gdp.bigm**	**112957**	**18242**	**47939**	**20466**	**17.1**	**443**

“IT DEPENDS…?”

- No formulation is the *general* answer.
- Even if one formulation was the *general* answer, I’m skeptical it would remain the same for future solver versions.
- These results are too preliminary to support any particular hypothesis about what problem attributes lend themselves to what formulations.
- ”Hybrid” formulations that have stronger relaxations than BigM but are not in the extended space of Hull do seem to have potential.
  - I’m curious about Between Steps for this reason.

Practical takeaway: General methods for reformulating GDPs that we can automate enable us to solve real problems! Trying everything implemented in pyomo.gdp only takes a computer’s time—not mine!

RETURNING TO THE QUESTION

Is Generalized Disjunctive Programming (GDP) a ***useful*** generalization of MIP?

- In the sense that it provide an automatic way to get from a (somewhat more) formulation-agnostic representation to a bunch of MI(N)LP formulations that may or may not serve me in solving the problem at hand: yes, definitely.

Open question: Could it be useful theoretically?

- There's very little research on *using* the logical structure in the solution method. Everything I presented here essentially still throws that away.
- Could the generalization help to characterize what formulations are computationally efficient, and why?

REFERENCES

- K.C. Furman, N.W. Sawaya, I.E. Grossmann, “A computationally useful algebraic representation of nonlinear disjunctive convex sets using the perspective function.” *Comput Optim Appl,* Volume 76, 589–614 (2020).
- Francisco Trespalacios, Ignacio E. Grossmann, “Improved Big-M reformulation for generalized disjunctive programs,” *Computers & Chemical Engineering,* Volume 76, 98-103, 2015.
- Francisco Trespalacios, Ignacio E. Grossmann, “Cutting Plane Algorithm for Convex Generalized Disjunctive Programs.” *INFORMS Journal on Computing* 28(2): 209-222, 2016.
- Jan Kronqvist, Ruth Misener, Calvin Tsay, “Between Steps: Intermediate Relaxations Between Big-M and Convex Hull Formulations,” In: Stuckey, P.J. (eds) *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* CPAIOR 2021. Lecture Notes in Computer Science, vol 12735, 2021