Introduction to Optimization Modeling in Python

Juan Javaloyes & Daniel Vázquez

February 2020

Contents

- Install PYOMO and solvers
- Mathematical Programming Background
- Introduction to Python
- PYOMO Components
- Case Studies

https://github.com/CAChemE/

Install PYOMO and Solvers

PYOMO

- `conda install -c conda-forge pyomo`
- `conda install -c conda-forge pyomo.extras`

http://www.pyomo.org/installation/

SOLVERS

- glpk [LP, MILP] >> `conda install -c conda-forge glpk`

http://ftp.gnu.org/gnu/glpk/

- gurobi [LP, MILP] >> download and install. Free university license

http://www.gurobi.com/

- IPOPT [NLP] >> `conda install -c conda-forge ipopt`

https://www.coin-or.org/download/binary/

- SCIP [MINLP] >> download and add the solver installation to the path environment variable

http://scip.zib.de/#download

PYOMO Sources

Homepage:

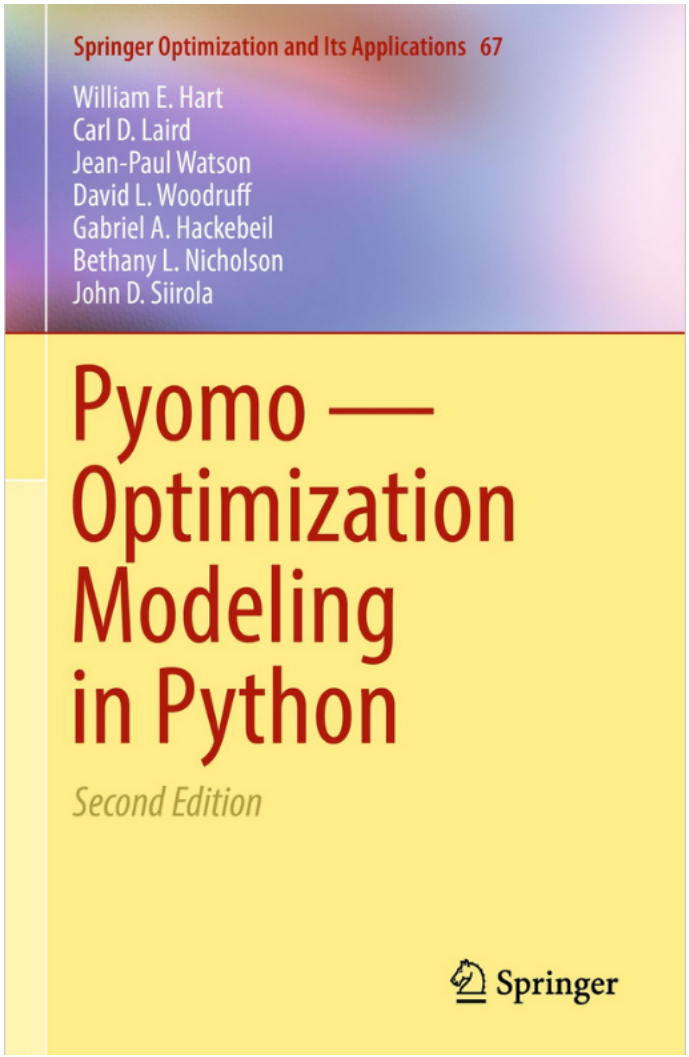
PYOMO
HOME / ABOUT / DOWNLOAD / DOCUMENTATION / BLOG

http://www.pyomo.org/

Sandia National Laboratories PYOMO

https://software.sandia.gov/trac/pyomo

Book Reference



PYOMO Sources

GitHub

https://github.com/Pyomo

Help Forums

https://groups.google.com/forum/#!forum/pyomo-forum

Stack Overflow

https://stackoverflow.com/questions/tagged/pyomo

Mathematical Programming Background

Mathematical Programming

Classes of Optimization Problems

$$\begin{aligned} \min \quad & f_0(x) \\ s.t \quad & f_i(x) \leq b_i \qquad i = 1, \ldots, m. \\ & x \in \mathbb{R}^n \end{aligned}$$

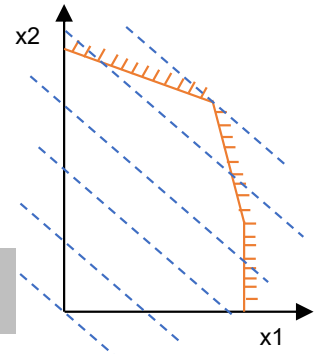
← Objective Function

← Constraints

$$f_0(x) : \mathbb{R}^n \rightarrow \mathbb{R}, f_i(x) : \mathbb{R}^n \rightarrow \mathbb{R}$$

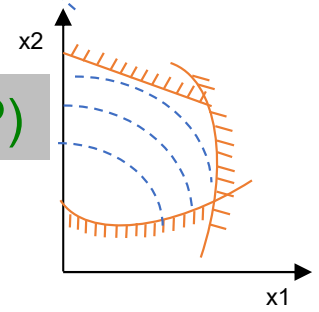
$$f_i(\alpha x_1 + \beta x_2) = x f_i(x_1) + \beta f_i(x_2) \quad x_1, x_2 \in \mathbb{R}^n, \alpha, \beta \in \mathbb{R}$$

← Linear Program (LP)



$$f_i(\alpha x_1 + \beta x_2) \neq x f_i(x_1) + \beta f_i(x_2) \quad x_1, x_2 \in \mathbb{R}^n, \alpha, \beta \in \mathbb{R}$$

← Nonlinear Program (NLP)



$$f_i(\alpha x_1 + \beta x_2) \leq x f_i(x_1) + \beta f_i(x_2) \quad x_1, x_2 \in \mathbb{R}^n, \alpha, \beta \in \mathbb{R} \text{ with } \alpha + \beta = 1, \alpha, \beta \geq 0$$

← Convex Optimization

Mathematical Programming

Classes of Optimization Problems

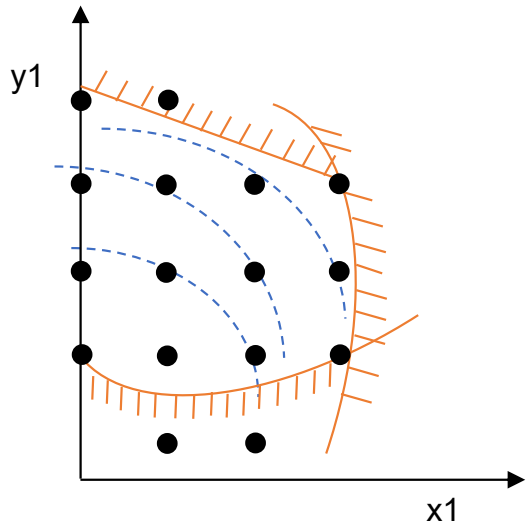
$$\begin{aligned} \min \quad & f_0\left(x, y\right) \\ s.t \quad & f_i\left(x, y\right) \leq b_i \qquad i = 1, \ldots, m. \\ & x \in R^n, \; y \in \left\{0,1\right\}^q \end{aligned}$$

$$f_0\left(x, y\right) : R^n \rightarrow R, \; f_i\left(x, y\right) : R^n \rightarrow R$$

← Objective Function

← Constraints

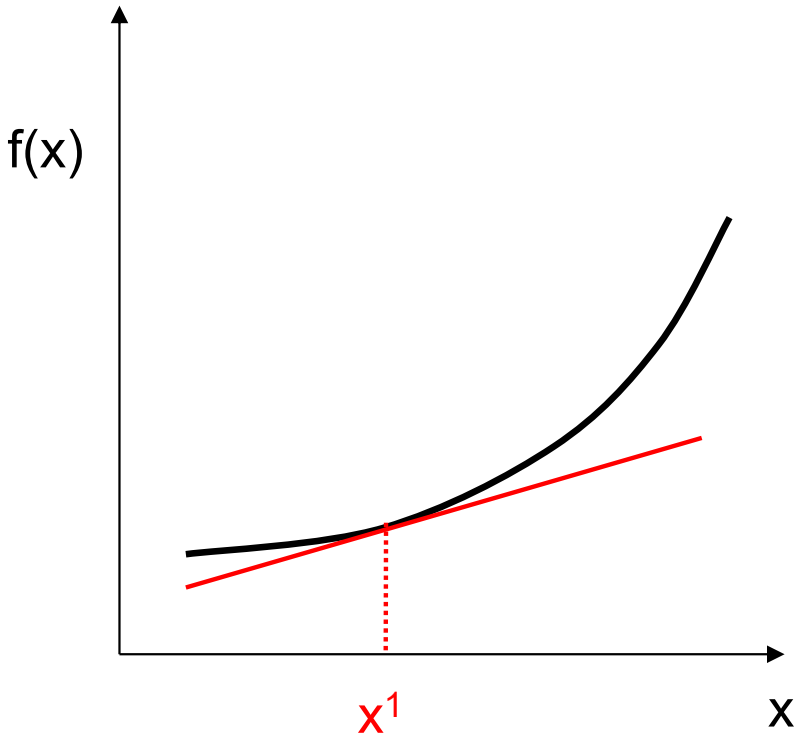
← Mixed-Integer (Non) Linear Programming (MI(N)LP)



Mathematical Programming

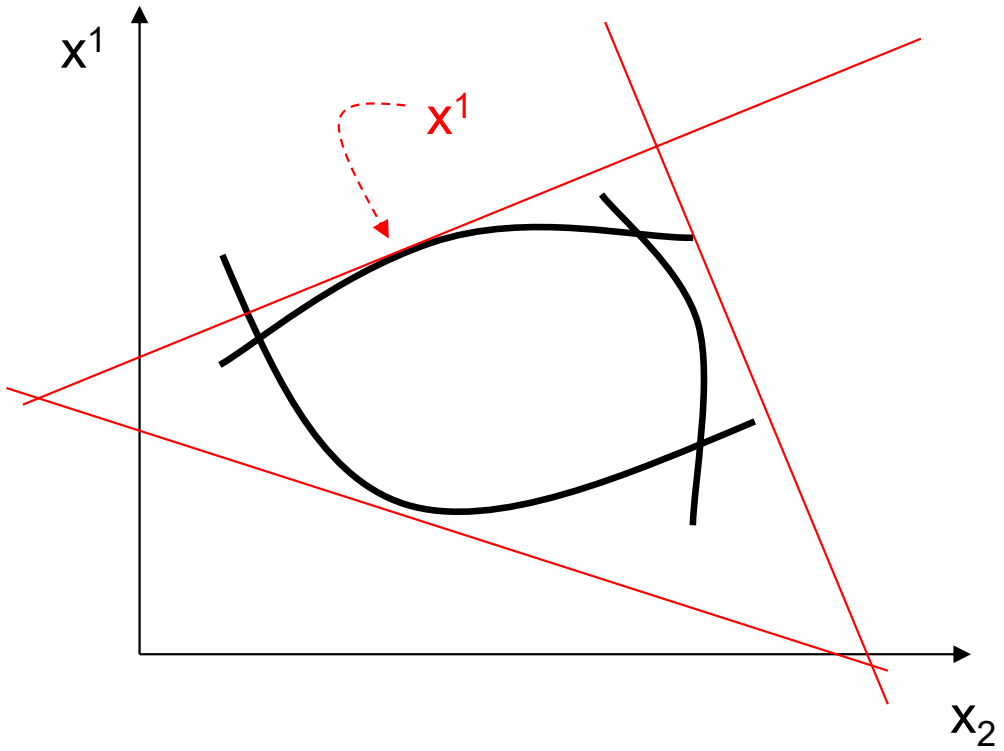
Classes of Optimization Problems – Convex Problems

Convex objective function



Sub-estimation of the Objective function

Convex feasible region

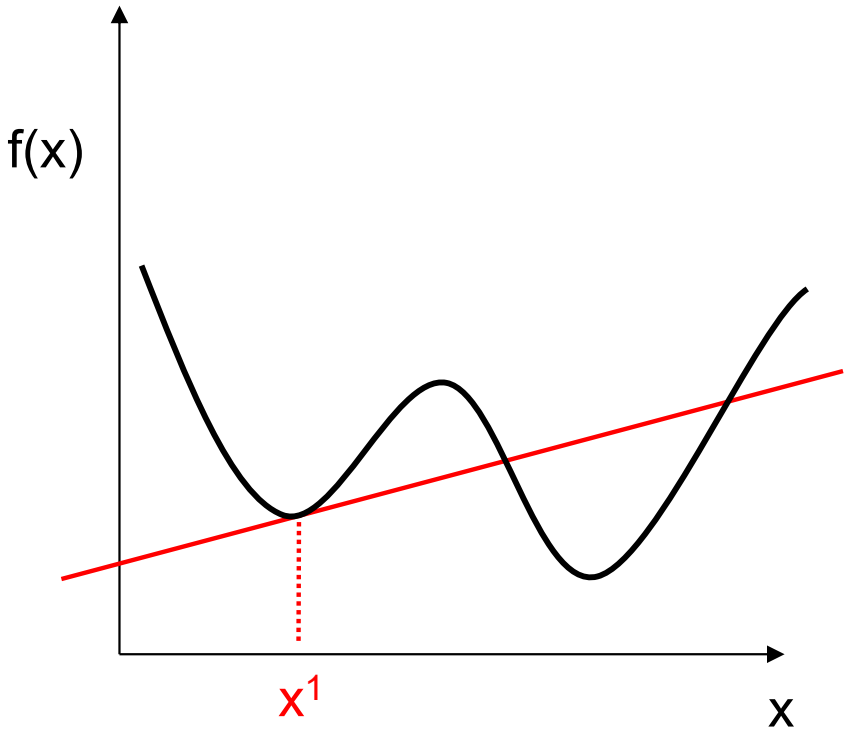


Overestimation of the feasible region

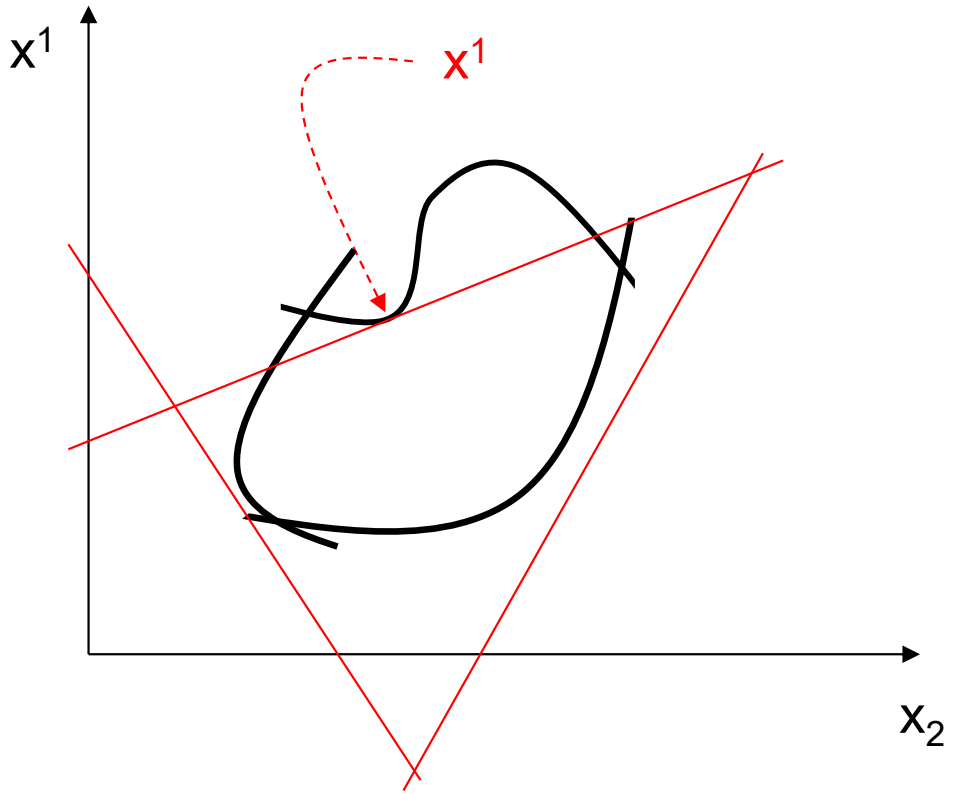
Mathematical Programming

Classes of Optimization Problems – Non-Convex Problems

Non-Convex objective function

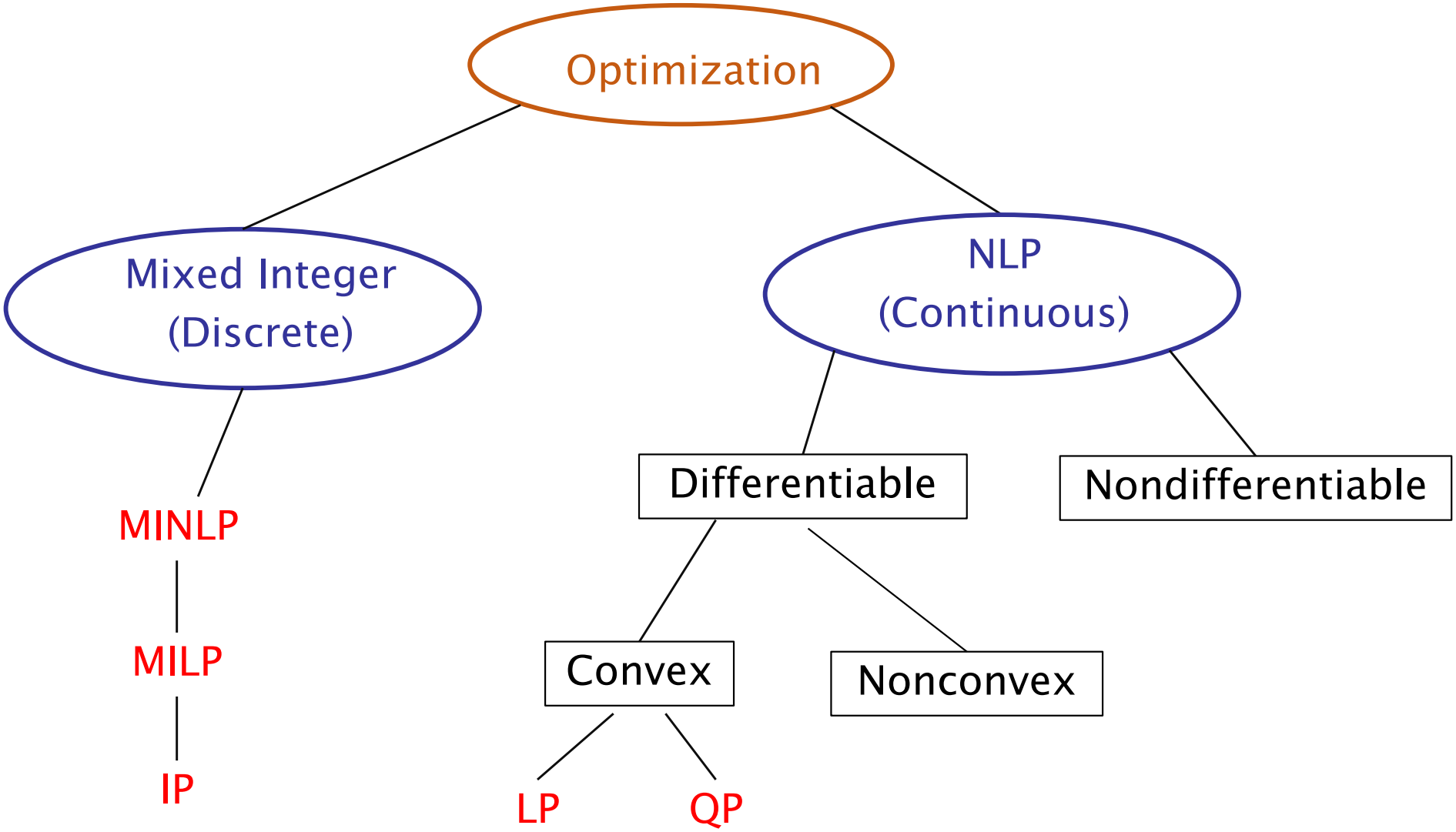


Non-Convex feasible region



Mathematical Programming

Classes of Optimization Problems



Mathematical Programming

Classes of Optimization Problems - Solvers

	Problem Class			
Solver	LP	MILP	NLP	MINLP
ALPHAECP				X
ANTIGONE			X	X
BARON	X	X	X	X
CONOPT4	X		X	
CPLEX	X	X		
DICOPT				X
GUROBI	X	X		
GLPK	X	X		
IPOPT	X		X	
SCIP		X	X	X

Mathematical Programming

Modeling of Discrete-Continuous Optimization Problems

Motivation Example (Grossmann & Trespalacios (2013), doi.org/10.1002/aic.14088)

"A company has to decide whether to produce either product A or product B in order to maximize its profit. The profit of product A is 3, and the profit of product B is 2. The limit on production of A is 4, and the limit in production of B is 5."

Mathematical Programming

Modeling of Discrete-Continuous Optimization Problems

Motivation Example (Grossmann & Trespalacios (2013), doi.org/10.1002/aic.14088)

$$
\begin{aligned}
\max \quad & 3A + 2B \\
s.t \quad & A\, y_2 = 0 \\
& B\, y_1 = 0 \\
& y_1\, y_2 = 0 \\
& y_1 + y_2 = 1 \\
& 0 \le A \le 4 \\
& 0 \le B \le 5 \\
& 0 \le y_1, y_2 \le 1 \\
& A, B, y_1, y_2 \in \mathbb{R}^n
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & 3A + 2B \\
s.t \quad & A \le 10\ (1 - y_2) \\
& \mathrm{B} \le 10\ (1 - y_1) \\
& y_1 + y_2 = 1 \\
& 0 \le A \le 4 \\
& 0 \le B \le 5 \\
& A, B \in \mathbb{R} \\
& y_1, y_2 \in \{0,1\}
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & 3A + 2B \\
s.t \quad & 0 \le A \le 4\ y_1 \\
& 0 \le B \le 5\ y_2 \\
& y_1 + y_2 = 1 \\
& A, B \in \mathbb{R} \\
& y_1, y_2 \in \{0,1\}
\end{aligned}
$$

Mathematical Programming

Generalized Disjunctive Programming (GDP) Formulation

$$\min: \; z = f(x)$$

$$s.t. \quad g(x) \leq 0$$

$$h(x) = 0$$

Objective function and global constraints

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{k,i} \\ r_{i,k}(x) \leq 0 \end{bmatrix} \qquad k \in K$$

$$\underline{\bigvee_{i \in D_k}} \; Y_{k,i} \qquad k \in K$$

Disjunctions

$$\Omega(Y) = True$$

Logic propositions

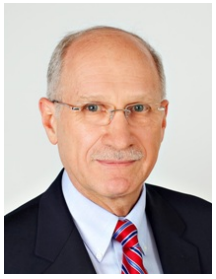
$$x^{lo} \leq x \leq x^{up}$$

$$x \in \mathbb{R}^n;$$

$$Y \in \{True, \; False\} \qquad k \in K, i \in D_k$$

Balas (1979)

Raman and Grossmann (1994)



Mathematical Programming

Generalized Disjunctive Programming (GDP) Formulation

GDP Reformulation (Grossmann & Trespalacios (2013), doi.org/10.1002/aic.14088)

GDP

$$\min: \ z = f(x)$$
$$s.t. \quad g(x) \le 0$$
$$h(x) = 0$$

$$\bigvee_{i \in D_k} \begin{bmatrix} Y_{k,i} \\ r_{i,k}(x) \le 0 \end{bmatrix} \quad k \in K$$

$$\bigvee_{i \in D_k} Y_{k,i} \quad k \in K$$

$$\Omega(Y) = True$$
$$x^{lo} \le x \le x^{up}, \quad x \in \mathbb{R}^n$$
$$Y \in \{True, False\} \quad k \in K, i \in D_k$$

Big-M (BM)

$$\min: \ z = f(x)$$
$$s.t. \quad g(x) \le 0$$
$$h(x) = 0$$

$$r_{ki}(x) \le M^{ki}(1 - y_{ki}) \quad k \in K, i \in D_k$$

$$\sum_{i \in D_k} y_{ki} = 1 \quad k \in K$$

$$Hx \ge h$$
$$x^{lo} \le x \le x^{up}, \quad x \in \mathbb{R}^n$$
$$y_{ki} \in \{0,1\} \quad k \in K, i \in D_k$$

Hull Reformulation (HR) [Linear]

$$\min: \ z = f(x)$$
$$s.t. \quad g(x) \le 0$$
$$h(x) = 0$$

$$x = \sum_{i \in D_k} v^{ki} \quad k \in K$$

$$y_{ki} r_{ki}\left(v^{ki} / y_{ki}\right) \le 0 \quad k \in K, i \in D_k$$

$$x^{lo} y_{ki} \le v^{ki} \le x^{up} y_{ki} \quad k \in K, i \in D_k$$

$$\sum_{i \in D_k} y_{ki} = 1 \quad k \in K$$

$$Hx \ge h$$
$$x \in \mathbb{R}^n$$
$$y_{ki} \in \{0,1\} \quad k \in K, i \in D_k$$

Mathematical Programming

Generalized Disjunctive Programming (GDP) Formulation

Systematic procedure to derive MILP constraints for Logic Propositions (Grossmann & Trespalacios (2013), doi.org/10.1002/aic.14088)

Proposition: set of literals Y_i separated by OR, AND, IMPLICATION

OR $$Y_1 \vee Y_2 \vee \ldots \vee Y_r \rightarrow y_1 + y_2 + \ldots + y_r \geq 1$$

AND $$Y_1 \wedge Y_2 \wedge \ldots \wedge Y_r \rightarrow y_1 \geq 1, y_2 \geq 1, \ldots, y_r \geq 1$$

IMPLICATION $$Y_1 \Rightarrow Y_2 \; equivalent \; to \quad \neg Y_1 \vee Y_2 \rightarrow 1 - y_1 + y_2 \geq 1 \quad \left(y_2 \geq y_1\right)$$

Mathematical Programming

Generalized Disjunctive Programming (GDP) Formulation

Systematic procedure to derive MILP constraints for Logic Propositions (Grossmann & Trespalacios (2013), doi.org/10.1002/aic.14088)

The goal is to convert logical expressions into Conjunctive Normal Form:

$Q_1 \wedge Q_2 \wedge \ldots \wedge Q_s \quad (\text{all AND})$ where clause $Q_i : Y_1 \vee Y_2 \vee \ldots \vee Y_i \quad (\text{all OR})$

Basic Steps

(1) Replace implication by disjunction $Y_1 \Rightarrow Y_2 \quad \Leftrightarrow \quad \neg Y_1 \vee Y_2$

(2) Move negation inward applying De Morgan's law $\neg(Y_1 \vee Y_2) \quad \Leftrightarrow \quad (\neg Y_1) \wedge (\neg Y_2)$

(3) Recursively distribute OR over AND $(Y_1 \wedge Y_2) \vee Y_3 \quad \Leftrightarrow \quad (Y_1 \vee Y_3) \wedge (Y_2 \vee Y_3)$

PYOMO Components

PYOMO Components

Example: Machinery Problem

A company manufacture four types of machinery. The factory is divided in three sections. The first section has available 960 h/week, the second 1110 h/week and the third 400 h/week. Each machinery unit requires the following time at each section

Plants	hours per machinery			Profit [units/machinery]
	Machining	Painting	Assembly	
Machinery 1	6	3	2	12
Machinery 2	4	3	1	8
Machinery 3	4	6	2	12
Machinery 4	8	9	1	17

Determine the number of units of machinery for each type that should be manufacture per week to maximize the profit.

PYOMO Components

Example: Machinery Problem

Nomenclature

- m ➔ machinery type (set)
- s ➔ factory section (set)
- $profit_m$ ➔ profit per machinery type (parameter)
- b_s ➔ time availability in each section per week (parameter)
- $T_{m,s}$ ➔ time required for each machinery type in each section (parameter)
- x_m ➔ number of units of machinery for each type (variable)

$$\max_x : \sum_m profit_m \, x_m$$

Objective function

$$s.t. \quad \sum_m T_{m,s} \, x_m \leq b_i \qquad \forall s$$

Factory section time limit

$$x_m \in \mathbb{Z}$$

Model Structure

```
from pyomo.environ import *
```

Import packages

```
m = ConcreteModel()
```

Create model object

```
M = m.M = Set(initialize =  ['m1', 'm2', 'm3', 'm4'])
S = m.S = Set(initialize =  ['s1', 's2', 's3'])
```

Sets declarations

```
profit    = {'m1':12, 'm2':8, 'm3':12, 'm4':17}
max_time = {'s1': 960, 's2': 1110, 's3': 400}
time_x_section = {
('m1','s1'): 6 , ('m1','s2'): 3 , ('m1','s3'): 2,
('m2','s1'): 4 , ('m2','s2'): 3 , ('m2','s3'): 1,
('m3','s1'): 4 , ('m3','s2'): 6 , ('m3','s3'): 2,
('m4','s1'): 8 , ('m4','s2'): 9 , ('m4','s3'): 1}
```

Specify/import problem data

```
x = m.x = Var( M, within = PositiveIntegers)
```

Variable declarations

```
m.value = Objective(
expr = sum( profit[i] * m.x[i] for i in M),
            sense = maximize )
```

Objective function declaration

```
def constraint_rule(m, j):
    return sum(time_x_section[i,j] * x[i] for i in M)
                    <= max_time[j]
m.constraint = Constraint(S, rule = constraint_rule)
```

Constraint functions declaration

```
opt = SolverFactory('glpk').solve(m)
```

Solver call

Case Studies

Knap-Sack Problem

In this problem, we are adventurous thieves. We want to loot all of the treasures that we can before the guards arrive. Since it will not be possible to come back to loot whatever we leave behind, we must ensure that we maximize the benefit of what we steal. Our horse can handle up until 2500 g of weight (It's a tiny pony) and a volume of 2000 cm3. Considering the loot table, what should we carry out there to sell?

Item	Market Price	Volume (cm3)	Unit Weight (g)	Units available
Chest	50	1000	2000	1
Ring	5	2	20	10
Necklace	3	10	300	1
Mirror	20	500	1000	1
Bracelet	16	15	300	15
Ruby	5	3	75	1
Parfum	1	100	100	1
Diamond	30	5	50	1
Gold goblet	12	250	500	1
Spice	40	100	100	1

Knap-Sack Problem

Mathematical Model

$$\max\left(\sum_i MP_i n_i\right)$$ Maximize Profit

$$s.t.$$

$$\sum_i V_i n_i \leq 2000$$ Volume constraint

$$\sum_i W_i n_i \leq 2500$$ Weight constraint

$$n_i \leq N_i \qquad \forall i$$ Amount constraint

Sudoku problem

Nomenclature

r ➔ rows | **c** ➔ columns | **k** ➔ value

$y_{r,c,k}$ ➔ binary variable. $y_{r,c,k} = 1$ means cell [r. c] is assigned number k

	c1	c2	c3	c4	c5	c6	c7	c8	c9
r1	5	3			7				
r2	6			1	9	5			
r3		9	8					6	
r4	8				6				3
r5	4			8					1
r6	7				2				6
r7		6					2	8	
r8				4	1				5
r9					8			7	9

Every position in the Sudoku is filled

$$\sum_{k} y_{r,c,k} = 1 \quad \forall r, c$$

Cells in the same column must be assigned distinct numbers

$$\sum_{r} y_{r,c,k} = 1 \quad \forall c, k$$

Cells in the same row must be assigned distinct numbers

$$\sum_{c} y_{r,c,k} = 1 \quad \forall r, k$$

Cells in the same 3x3 grid must be assigned distinct numbers

$$\sum_{r=3p-2}^{3p} \sum_{c=3q-2}^{3q} y_{r,c,k} = 1 \quad \forall k,\ p, q = \{1, 2, 3\}$$

Non Sharp Separation

Nomenclature

Sets

$C := \{c : c \text{ is a column}\}$

$I := \{i : i \text{ is a component}\}$

Continuous Variables

$F_{c,i}$: molar feed flow to column c of component i

$D_{c,i}$: molar distillate flow of column c and component i

$B_{c,i}$: molar bottoms flow of column c and component i

$S1_i$: Feed stream bypass to node A, component i

$S2_i$: Feed stream bypass to node B, component i

$S3_i$: Feed stream bypass to node C, component i

$ProdA_i$: molar productA flow of component i

$ProdB_i$: molar productB flow of component i

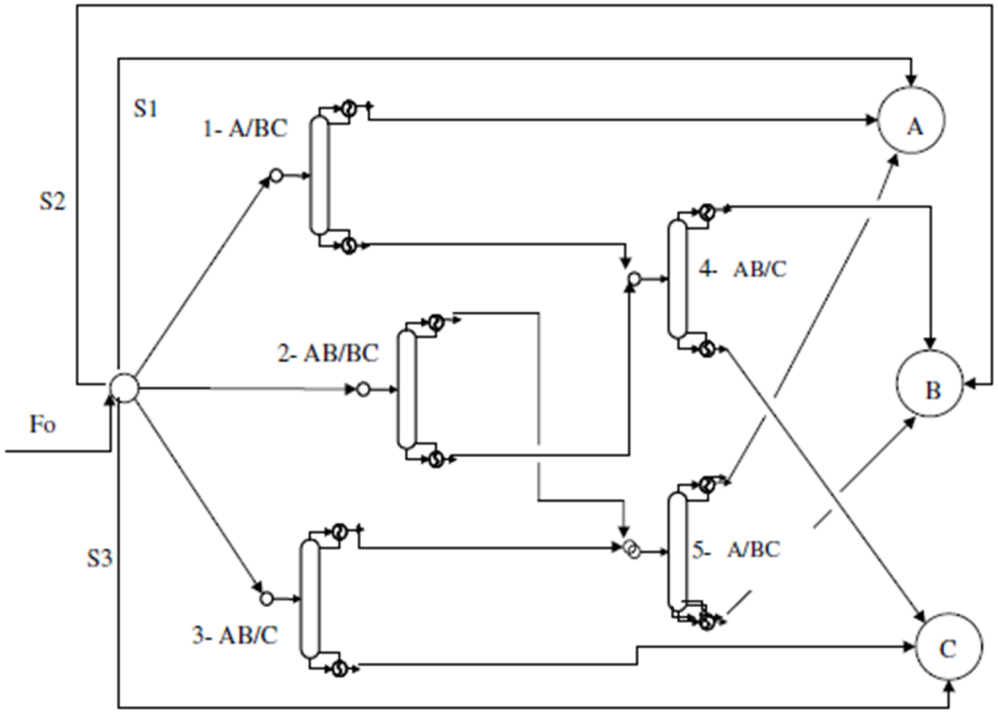
$ProdC_i$: molar productC flow of component i

$Cost_c$: cost of distillation column c

Binary Variables

y_c : 1 if column c is selected. 0 otherwise

Distillation Sequence Superstructure



Parameters

F_{0i} : molar feed flow to distillation sequence of component i

FC_c : column c fixed cost

FV_c : column c variable cost

$x_{c,i}$: column c component i fraction in distillate

Non Sharp Separation

GDP Formulation

$$\min \sum_{c \in C} Cost_c$$

Objective function

$$
\begin{aligned}
s.t \quad & F_{0i} = S1_i + S2_i + S3_i + F_{c1,i} + F_{c2,i} + F_{c3,i} \quad \forall i \in I \\
& F_{c4,i} = B_{c1,i} + B_{c2,i} \quad \forall i \in I \\
& F_{c5,i} = D_{c2,i} + D_{c3,i} \quad \forall i \in I \\
& \\
& ProdA_i = S1_i + D_{c1,i} + D_{c5,i} \quad \forall i \in I \\
& ProdB_i = S2_i + D_{c4,i} + B_{c5,i} \quad \forall i \in I \\
& ProdC_i = S3_i + B_{c3,i} + B_{c4,i} \quad \forall i \in I \\
& \\
& \sum_{i \in I} SX_i \; F_{0i} = \sum_{i \in I} F_{0i} \; SX_i \quad \forall i \in I, SX = \{S1, S2, S3\} \\
& \\
& \sum_{i \in I} F_{c,i} \; F_{0i} = \sum_{i \in I} F_{0i} \; F_{c,i} \quad \forall i \in I, c = \{c1, c2, c3\} \\
& F_{c,i} = D_{c,i} + B_{c,i} \quad \forall c \in C, i \in I \\
& D_{c,i} = F_{c,i} \, x_{c,i} \quad \forall c \in C, i \in I
\end{aligned}
$$

Global constraints

Non Sharp Separation

GDP Formulation

$$\begin{bmatrix} Y_c \\ Cost_c = FC_c + FV_c \sum_{i \in I} D_{c,i} \\ F_{c,i} \leq F_{0i} \quad \forall i \in I \end{bmatrix} \vee \begin{bmatrix} \neg Y_c \\ Cost_c = 0 \\ F_{c,i} = 0 \quad \forall i \in I \end{bmatrix} \quad \forall c \in C$$

Disjunctions

$$Y_1 \vee Y_2 \vee Y_3$$
$$\Omega(Y) = True$$

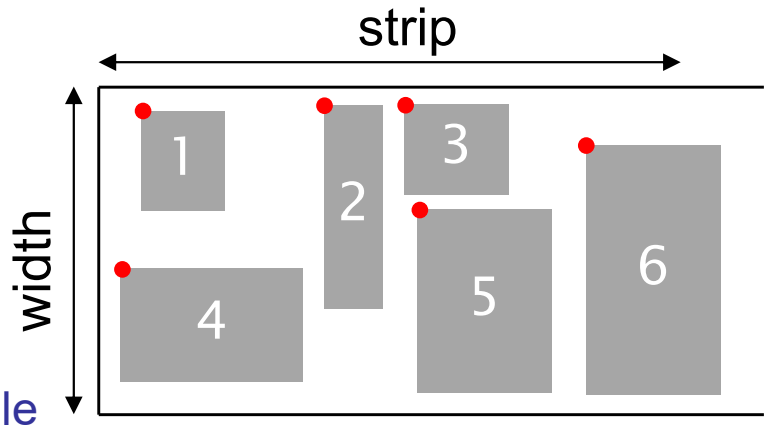
Logic propositions

$$F_{ci}, D_{ci}, D_{c,i}, S1_i, S2_i, S3_i, ProdA_i, ProdB_i, ProdC_i, Cost_c \in \mathbb{R}$$
$$Y_{i,j} \in \{Ture, False\}$$

Strip packing 2D problem

Nomenclature

- **i** ➔ rectangles, i = {1,2,…,n}
- **lt** ➔ length of the strip
- **(x_i,y_i)** ➔ rectangle coordinates
- L_i, H_i ➔ Length and height of rectangle i
- W ➔ Width of the strip
- UB_i ➔ Upper bound for the x-coordinate of every rectangle



Source: Sawaya & Grossmann (2005), https://doi.org/10.1016/j.compchemeng.2005.04.004

GDP Formulation

$$\min\ lt$$

$$s.t\quad lt \geq x_i + L_i \qquad \forall i \in N$$

$$\begin{bmatrix} Y^1_{i,j} \\ y_j - H_j \geq y_i \end{bmatrix} \vee \begin{bmatrix} Y^2_{i,j} \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Y^3_{i,j} \\ y_i - H_i \geq y_j \end{bmatrix} \vee \begin{bmatrix} Y^4_{i,j} \\ x_j + L_j \leq x_i \end{bmatrix} \qquad \forall i,j \in N, i < j$$

$$Y^1_{i,j} \veebar Y^2_{i,j} \veebar Y^3_{i,j} \veebar Y^4_{i,j} \qquad \forall i,j \in N, i < j$$

$$x_i \leq UB - L_i \qquad \forall i \in N$$

$$H_i \leq y_i \leq W \qquad \forall i \in N$$

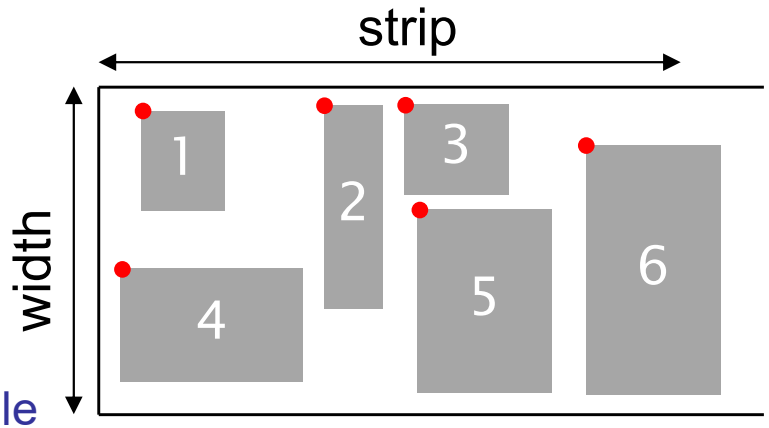
$$lt, x_i, y_i \in \mathbb{R}$$

$$Y_{i,j} \in \{Ture, False\}$$

Strip packing 2D problem

Nomenclature

- **i** ➔ rectangles, i = {1,2,…,n}
- **lt** ➔ length of the strip
- **(x_i,y_i)** ➔ rectangle coordinates
- L_i, H_i ➔ Length and height of rectangle i
- W ➔ Width of the strip
- UB_i ➔ Upper bound for the x-coordinate of every rectangle



MILP Formulation [Big-M]

$$\min\ lt$$

$$s.t\quad lt \geq x_i + L_i \qquad \forall i \in N$$

$$y_j - H_j \geq y_i - M_{ij}^1\left(1 - w_{ij}^1\right) \qquad \forall i,j \in N, i < j$$

$$x_i + L_i \leq x_j + M_{ij}^2\left(1 - w_{ij}^2\right) \qquad \forall i,j \in N, i < j$$

$$y_i - H_i \geq y_j - M_{ij}^3\left(1 - w_{ij}^3\right) \qquad \forall i,j \in N, i < j$$

$$x_j + L_j \leq x_i + M_{ij}^4\left(1 - w_{ij}^4\right) \qquad \forall i,j \in N, i < j$$

$$\sum_{d \in D} w_{ij}^d = 1 \qquad \forall i,j \in N, i < j$$

$$x_i \leq UB - L_i \qquad \forall i \in N$$

$$H_i \leq y_i \leq W \qquad \forall i \in N$$

$$lt, x_i, y_i \in \mathbb{R}, \qquad w_{i,j} \in \{0,1\}$$

More Complex Constraints Examples

The Expanded Transshipment Model (Papoulias & Grossmann, 1983.)

$$R_{i,k} + \sum_{j \in C_{jk}} Q_{ijk} + \sum_{n \in W_{nk}} Q_{ink} = R_{i,k-1} + Q^H_{ik} \qquad \forall i,k \in HS_{i,k}$$

GAMS

```
EQ_01_EB_A(HS(i,k))$(ord(k) eq 1).. R(i,k) + sum(j$C(j,k), Qijk(i,j,k)) +
                                    sum(n$W(n,k), Qink(i,n,k)) =E=  Q_H(i,k);

EQ_02_EB_A(HS(i,k))$(ord(k) ne 1).. R(i,k) + sum(j$C(j,k), Qijk(i,j,k)) +
                                    sum(n$W(n,k), Qink(i,n,k)) =E=  Q_H(i,k) +R(i,k-1);
```

PYOMO

```
def energy_balance_A_rule(model, i, k):
    if (i,k) in model.HS:
        if k == model.K[1]:
            return (
                    model.R[i,k] + sum(model.Qijk[i,j,k] for j in model.J if (j,k) in model.C) +
                    sum(model.Qink[i,n,k] for n in model.N if (n,k) in model.W) ==
                    model.Q_H[i,k]
                    )
        else:
            return (
                    model.R[i,k] + sum(model.Qijk[i,j,k] for j in model.J if (j,k) in model.C) +
                    sum(model.Qink[i,n,k] for n in model.N if (n,k) in model.W) ==
                    model.R[i, model.K[model.K.ord(k)-1]] + model.Q_H[i,k]
                    )
    else:
        return Constraint.Skip
model.energy_balance_A = Constraint(model.I, model.K, rule = energy_balance_A_rule)
```

Introduction to Optimization Modeling in Python

Juan Javaloyes & Daniel Vázquez

February 2020