# Good Optimization Modeling Practices with GAMS

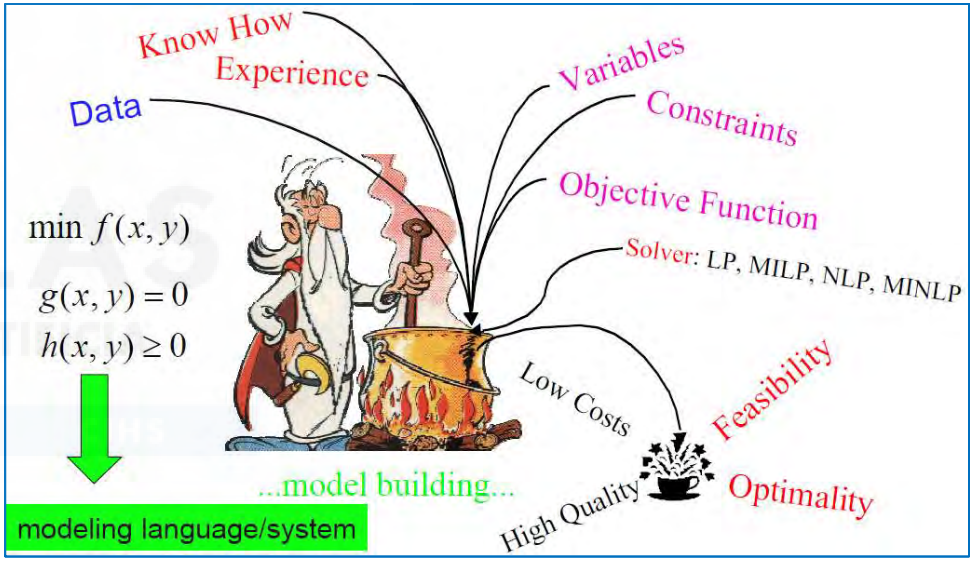
*All You Wanted to Know About Practical Optimization but Were Afraid to Ask*


Andrés Ramos
Andres.Ramos@comillas.edu
https://www.iit.comillas.edu/aramos/

Pedro de Otaola
Pedro.Otaola@comillas.edu

# Do not confuse the ingredients of the recipe

- Mathematical formulation
  - LP, MIP, NLP, QCP, MCP
- Algebraic modeling language
  - GAMS, Pyomo
- Solver
  - Gurobi, IBM CPLEX, HiGHS, PATH
- Optimization algorithm
  - Primal simplex, dual simplex, interior point
- Input/output interfaces
  - Text file, CSV, Microsoft Excel, Matlab, Microsoft Access
- Operating system
  - Windows, Linux, macOS
- Advanced algorithms
  - Benders decomposition, Lagrangian relaxation, genetic algorithms
- Stochastic extensions
  - EMP



TEPES

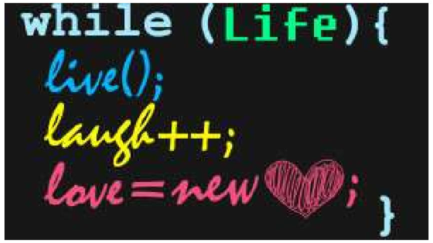
*TEPES Long-Term Transmission Expansion Planning Model for an Electric System*

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

https://www.iit.comillas.edu/aramos/TEPES.htm

Run

Load results




1 — Programming Style

1. **Programming Style**
2. GAMS Code
3. Embedded Python
4. Connect
5. Performance issues
6. Advanced Algorithms

2 — GAMS Code

1. Programming Style
2. **GAMS Code**
3. Embedded Python
4. Connect
5. Performance issues
6. Advanced Algorithms

3 — Embedded Python

1. Programming Style
2. GAMS Code
3. **Embedded Python**
4. Connect
5. Performance issues
6. Advanced Algorithms

4 — Connect (data input and output)

1. Programming Style
2. GAMS Code
3. Embedded Python
4. **Connect**
5. Performance issues
6. Advanced Algorithms

4 — Performance Issues

1. Programming Style
2. GAMS Code
3. Embedded Python
4. Connect
5. **Performance issues**
6. Advanced Algorithms

5 — Advanced Algorithms

1. Programming Style
2. GAMS Code
3. Embedded Python
4. Connect
5. Performance issues
6. **Advanced Algorithms**

$$\sum_{i=t-TU_g^x+1}^{t} \sum_{y \in \mathcal{M}_g^{\mathrm{F},x}} v_{gi}^{yx} \le u_{gt}^x \quad \forall g, x, t \in [TU_g^x, T] \qquad (4)$$

$$\sum_{i=t-TD_g^x+1}^{t} \sum_{x \in \mathcal{M}_g^{\mathrm{F},y}} v_{gi}^{xy} \le 1 - u_{gt}^x \quad \forall g, x, t \in [TD_g^x, T]\,. \qquad (5)$$

```
* bounds on variables

vProduct.up (sc,n,g) $pScenProb(sc) = pMaxProd   (g   ) ;
vConsump.up (sc,n,g) $pScenProb(sc) = pMaxCons   (g   ) ;
vProduct1.up(sc,n,t) $pScenProb(sc) = pMaxProd   (t   ) - pMinProd(t) ;
vIG.up      (sc,n  ) $pScenProb(sc) = pIntermGen (n,sc) ;
vENS.up     (sc,n  ) $pScenProb(sc) = pDemand    (n   ) ;
vReserve.up (sc,n,g) $pScenProb(sc) = pMaxReserve(g   ) ;
vReserve.lo (sc,n,g) $pScenProb(sc) = pMinReserve(g   ) ;

vCommitt.up (   n,g) = 1 ;
vStartup.up (   n,g) = 1 ;
vShutdown.up(   n,g) = 1 ;

* solve stochastic daily unit commitment model

solve SDUC using MIP minimizing vTotalVCost ;
```

1

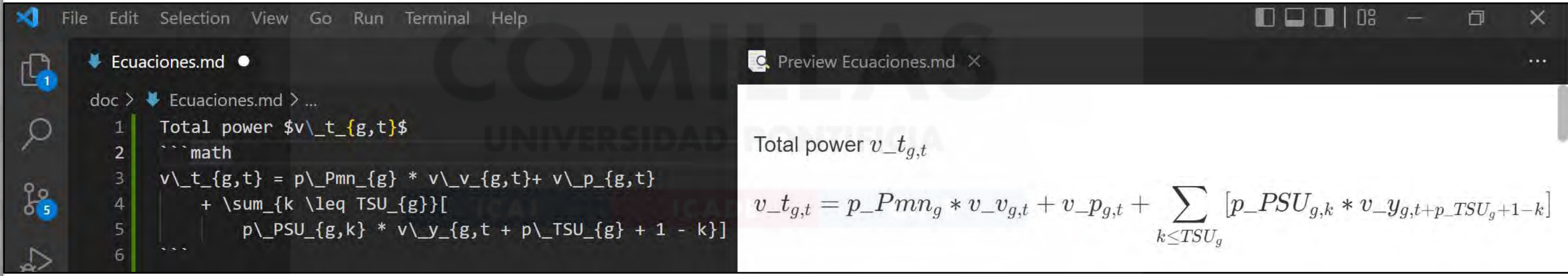


# Programming Style

# Write the equations before trying to code them!!

Digital formats are useful for storing the documentation with the code

- Word: easy for beginners
- Markdown (or LaTeX):
  - Faster to write once you learn
  - Easy to keep track of changes using a repository
  - “Reusable” to produce code

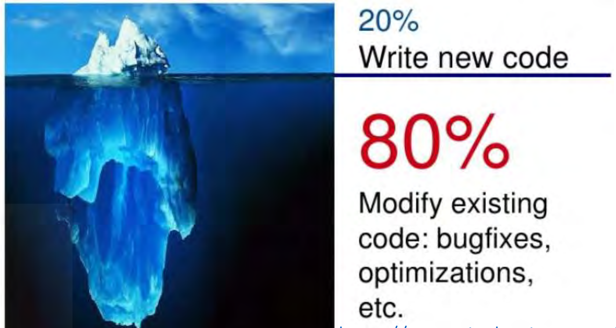



- `MarkDown: v\_t_{g,t} =   p\_Pmn_{g} * v\_v_{g,t} + v\_p_{g,t}`
- `GAMS:     v_t  {g,t} =E= p_Pmn  {g} * v_v  {g,t} + v_p  {g,t}`
- `MarkDown : + \sum_{k \leq TSU_{g}}[p\_PSU_{g,k} * v\_y_{g,t + p\_TSU_{g} + 1 - k}]`
- `GAMS:      + sum[{k${k <= TSU{g}}, p_PSU  {g,k} * v_y  {g,t + p_TSU  {g} + 1 - k}]`

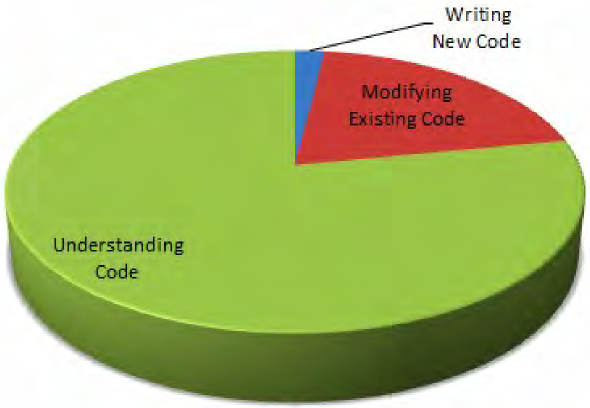
# General recommendations

Clarity
Modularity
Completeness
Interoperability
Maintainability
Standardization

- Act according to the Pareto principle
  - It takes 20 % to create the prototype
  - 80 % of code development is devoted to maintenance and refinement
- MAINTAINABILITY and reusability are crucial
- Code is developed to be read by humans, not by machines. Write code to understand the model, not to obscure it.
- Say what you mean and directly.
- Don't stop with your first draft. Refine it.




http://www.jetbrains.com/


http://blog.codinghorror.com/when-understanding-means-rewriting/

# Code style


- Any project manager ought to define the style before starting up a multiple-participant project (or maybe just for their help)
- Systematic and consistent use of uppercase and lowercase letters
  - Use lowercase letters instead of uppercase. We are more used to reading lowercase letters.
  - GAMS doesn't distinguish them; you are responsible for always using the same.
- Clean code and take care of the aesthetics when coding
  - Aesthetics is as important as the content. The code must be read immediately.
- Format the code to help the reader understand it.
  - Indent to show the logical structure of a program.
  - Keep coherence in the coding rules (indent in repetitive sentences)
  - Align code to show patterns.
  - Make reading easier (parallelism among consecutive similar sentences, indent)
- Use meaningful and long names for identifiers. The consistent use of identifiers in different parts of the code.

# Efficiency vs. Clarity

- Make it clear and right before you make it faster
- Keep it simple to make it faster
- Don't sacrifice clarity for small gains in efficiency

# Documentation. Comments

- It is a crucial task in code development
  - GAMS was born to include documentation in the code explicitly.
- Code must be self-documented
- Illustrative comments and well-localized
- Make sure comments and code agree
- Don’t just echo the code with comments - make every comment count
- Don’t comment on lousy code or tricks - rewrite
- Don’t patch the wrong code - rewrite it
- Don’t over-comment

# Procrastination

- Don’t procrastinate when coding



2

# GAMS Code

# Search, compare, and if you find something better, use it

v. 49.6.0

GAMS

AIMMS

v. 1.25.0

JUMP

AMPL
STREAMLINED MODELING
FOR REAL OPTIMIZATION

v. 6.9.2

PYOMO

MAXIMAL SOFTWARE

R

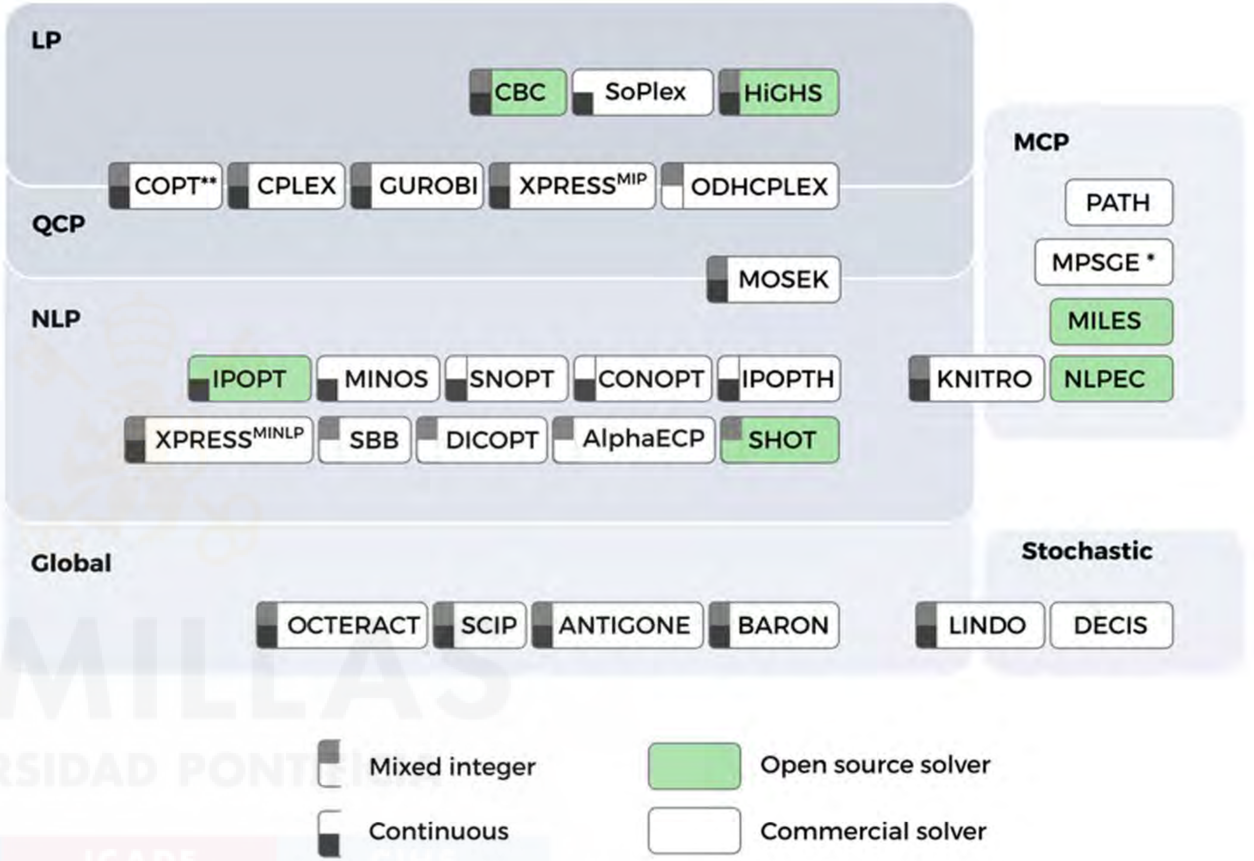
MathWorks

LINDO SYSTEMS INC.

FRONTLINE solvers

Solver Studio for Excel

# Solvers




https://www.gams.com/blog/2022/09/an-overview-of-math-programming-solvers/

* MPSGE is not a solver, but a GAMS subsystem dedicated to solving economic equillibrium models

** COPT does not handle MIQCPs

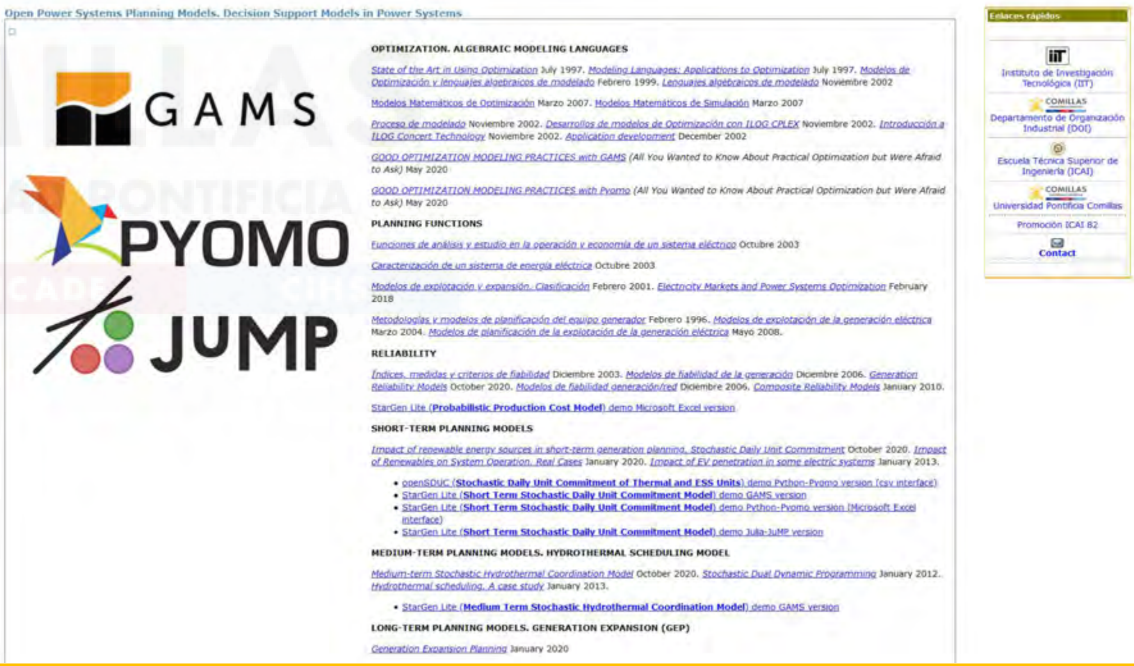
Fig. 1: Overview of the solvers included with GAMS. LP: Linear Program, QCP: Quadratically Contrained Program, NLP: Non-linear Program, MCP: Mixed Complementarity Program. Solvers that lie on the boundaries of two problem types are well suited to solve both problem types. Please note that this figure does not accurately reflect every solvers capabilites. Instead, it is meant to give a quick anser to the question: Which solver should I try first, given a problem class?

# Learning by reading first, and then by doing

- GAMS Model Libraries

  (https://www.GAMS.com/modlibs/)

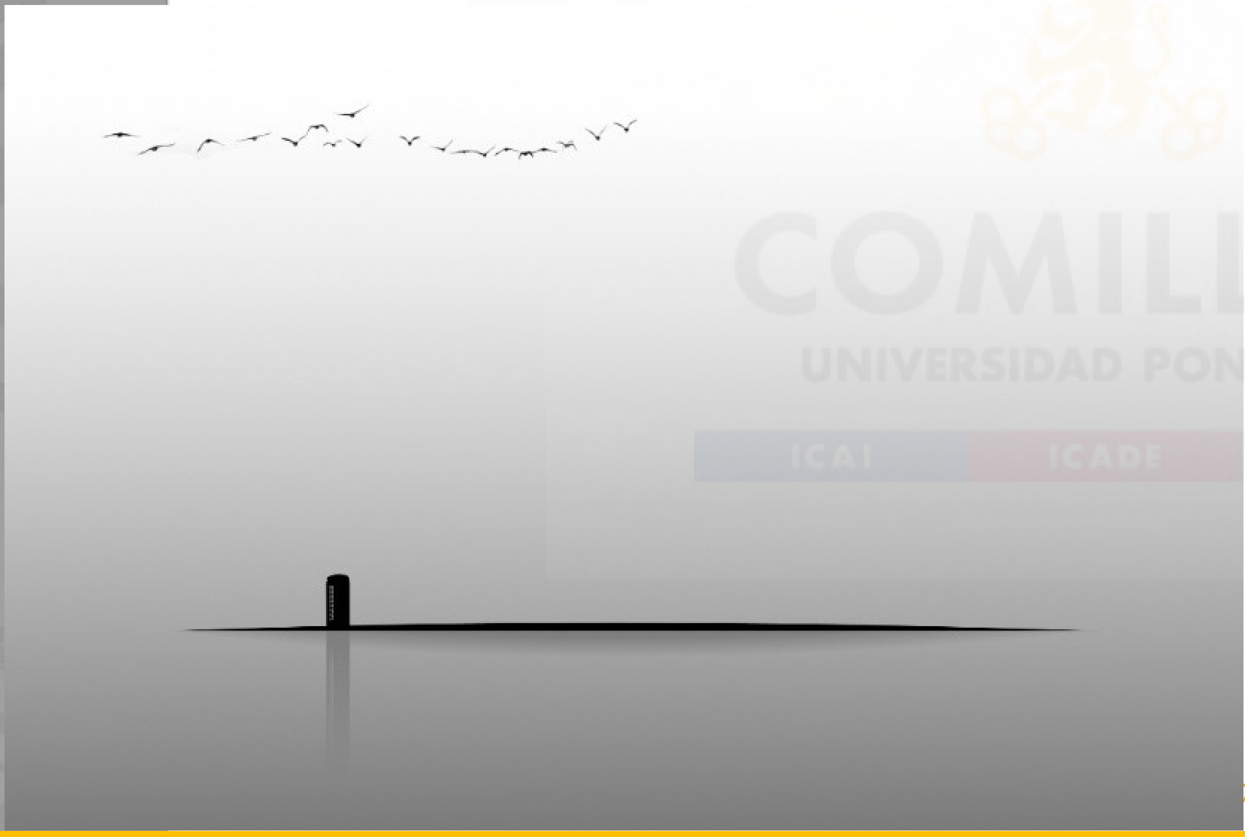
- Decision Support Models in the Electric Power Industry
  (https://pascua.iit.comillas.edu/aramos/openmodels.htm)



# Primer on optimization

- Optimization techniques
  - https://pascua.iit.comillas.edu/aramos/OT.htm
- Deterministic optimization cases
  - https://pascua.iit.comillas.edu/aramos/simio/transpa/s_OptimizationCases.pdf
- Stochastic optimization cases
  - https://pascua.iit.comillas.edu/aramos/simio/transpa/s_StochasticOptimizationCases.pdf
- A GAMS Tutorial by Richard E. Rosenthal
  - https://www.GAMS.com/latest/docs/UG_Tutorial.html

# My first minimalist optimization model



```
positive variables x1, x2
         variable  z

equations of, e1, e2, e3 ;

of .. 3*x1 + 5*x2 =e=  z ;
e1 ..   x1        =l=  4 ;
e2 ..        2*x2 =l= 12 ;
e3 .. 3*x1 + 2*x2 =l= 18 ;

model minimalist / all /
solve minimalist maximizing z using LP
```

$$
\begin{aligned}
\max_{x_1,x_2} z &= 3x_1 + 5x_2 \\
x_1 & \qquad\qquad \le 4 \\
& \quad 2x_2 \quad \le 12 \\
3x_1 & + 2x_2 \quad \le 18 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

# Blocks in a GAMS model

- Mandatory

  `variables`
  `equations`
  `model`
  `solve`

- Optional

  `sets: (alias)`
  - `alias (i,j)` *i* and *j* can be used indistinctly
  - Checking of domain indexes

  data: `scalars, parameters, table`

# Transportation model

There are $i$ can factories and $j$ consumption markets. Each factory has a maximum capacity of $a_i$ cases, and each market demands a quantity of $b_j$ cases (it is assumed that the total production capacity is greater than the total market demand for the problem to be feasible). The transportation cost between each factory $i$ and each market $j$ for each case is $c_{ij}$. The demand must be satisfied at a minimum cost.

The decision variables of the problem will be cases transported between each factory $i$ and each market $j$, $x_{ij}$.

# My first transportation model (classical organization)

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/TransportModel.gms

```
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pB(j) destination demand
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

table pC(i,j) per unit transportation cost
          MADRID BARCELONA VALENCIA
VIGO        0.06      0.12      0.09
ALGECIRAS   0.05      0.15      0.11

variables
   vX(i,j) units transported
   vCost   transportation cost

positive variable vX

equations
   eCost        transportation cost
   eCapacity(i) maximum capacity of each origin
   eDemand  (j) demand supply at destination ;

eCost        .. sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) .. sum[   j ,          vX(i,j)] =l= pA(i) ;
eDemand  (j) .. sum[ i   ,          vX(i,j)] =g= pB(j) ;

model mTransport / all /
solve mTransport using LP minimizing vCost
```

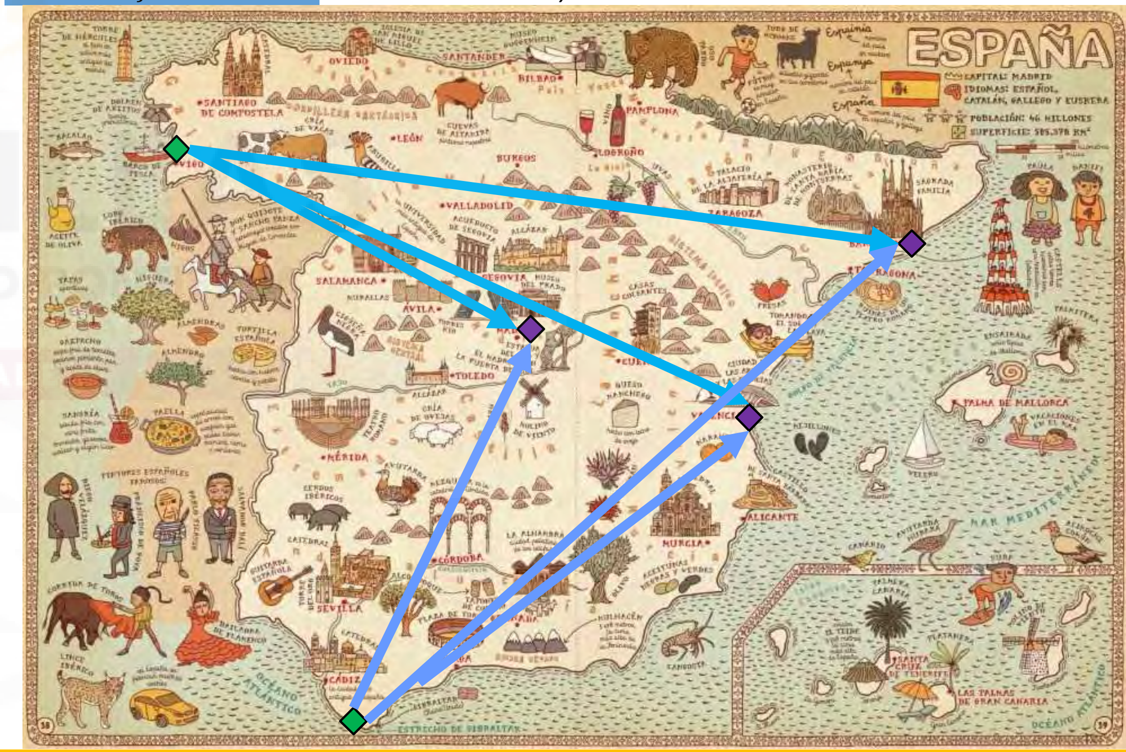
$$\min_{x_{ij}} \sum_{ij} c_{ij} x_{ij}$$

$$\sum_{j} x_{ij} \le a_i \quad \forall i$$

$$\sum_{i} x_{ij} \ge b_j \quad \forall j$$

$$x_{ij} \ge 0$$

A. Mizielinska y D. Mizielinski *Atlas del mundo: Un insólito viaje por las mil curiosidades y maravillas del mundo* Ed. Maeva 2015



# General structure of GAMS sentences

- Commenting
  - Lines with * in the first column
  - `$OnText $OffText` to comment on many lines
- No distinction between uppercase and lowercase letters
- Parentheses (), square brackets [], or braces {} can be used indistinctly to distinguish levels.
- Language-reserved words appear in bold
- Sentences end with a ";"
  - Can be suppressed when the following word is a reserved one (in blue (light theme) or orange (dark theme))

# Parentheses (), square brackets [] or braces {}

- Markdown/LaTeX/Pyomo do differentiate; keep in mind that you may want to reuse the code when choosing your style!
- Establish a style and be consistent
- Take advantage of the available option to differentiate operations and make the code easier to follow
- Suggestion:
  - Mathematical expressions: (A + B)
  - Sets: A{s}
  - Functions and conditions: sum[..], smax[..], $[..]
  - Example:
    - `Just parentheses: A = sum(s, B(s) * (C(s) + D(s)$(condition(s))));`
    - `Suggested option: A = sum[s, B{s} * (C{s} + D{s}$[condition{s}])];`

With complex code makes it easier to follow

# Basic input/output in text format

- Data input from a text file

```
$include FileName.txt
display IdentifierName (shows its content or value)
```

- Data output to a text file

```
file InternalName / ExternalName.txt /
put       InternalName
   put    IdentifierName
putclose InternalName
```

`ExternalName.txt` is updated each time the instruction **`putclose`** is executed.

- Specific options to control the output format
  - Put Writing Facility

# Reporting of complex processes

- For long processes with multiple optimizations, it's useful to print intermediate data to a file to keep track of them.
- Text is written to the file each time the command putclose is used
- Use `Infoexecution.ap=1` to keep writing in the same file; otherwise, the file will be overwritten each time

```
scalar    s_jnow;
set       week /w1*w5/;
variables v_fob;
equations eq1;
eq1.. v_fob =G= 0;

model mod
/all/
;

file InfoExecution / 'InfoExecution.out' /;
InfoExecution.lw=4;
InfoExecution.nd=2;
InfoExecution.nw=10;
put InfoExecution;
put "Execution report exampel"//;
put "Week solveStat modelStat optcr [%]  Time [s]";
putclose InfoExecution;
InfoExecution.ap = 1;

loop(week,
    s_jnow=jnow;
    SOLVE mod  minimizing v_fob using MIP;
    s_jnow = [(jnow-s_jnow)*86400];
    put InfoExecution;
        put week.tl;
        put mod.solveStat ;
        put mod.modelStat ;
        put ((100 * abs(mod.objest - mod.objval)
              /(1e-10+abs(mod.objval))
              )$[(1e-10+abs(mod.objval))])  ;
        put s_jnow/;
    putclose InfoExecution;
);
```

```
Execution report example

Week solveStat modelStat optcr [%]  Time [s]
w1        1.00      1.00        NA      0.09
w2        1.00      1.00        NA      0.09
w3        1.00      1.00        NA      0.11
w4        1.00      1.00        NA      0.12
w5        1.00      1.00        NA      0.09
```

# Functions and operators

(https://www.GAMS.com/latest/docs/UG_Parameters.html#UG_Parameters_Functions)

- `+, -, *, /, **` or `power(x,n)`
- `abs, arctan, sin, cos, ceil, floor, exp, log, log10, max, min, mod, round, sign, sqr, sqrt, trunc, normal, uniform`
- `gyear, gmonth, gday, ghour, gminute, gsecond, gdow, gleap, jdate, jnow, jstart, jtime`
- `lt <, gt >, eq =, ne <>, le <=, ge >=`
- `not, and, or, xor`
- `diag(set_element,set_element)={1,0}`
- `sameas(set_element,set_element)={T,F}`
- `ord, card` ordinal and cardinal of a set, `SetName.pos` ordinal of a set
  - set.ord and ord(set) are valid, but only card(set) is valid
- `sum, prod, smax, smin`
- `inf, eps, pi` are valid as data

# Model temporal license

```
abort $[jstart > jdate(2021,11,21)] 'License for this model has expired and it cannot be used any more. Contact the developers'
```

# $ Operator in assignments, summations, constraints

- Sets a condition

```
$(value > 0)             $(number1 <> number2)
```

- On the left of an assignment (p$[condition]=v), it does the assignment ONLY if the condition is satisfied

```
if (condition,
    DO THE ASSIGNMENT
);
```

- On the right of an assignment (p=v$[condition]), it does the assignment ALWAYS, and if the condition is not satisfied, it assigns a value of 0

```
if (condition,
    DO THE ASSIGNMENT
else
    ASSIGNS VALUE 0
);
```

- Conditions to parts: a = b + c $[d]. If d=true, then a = b + c. If d=false, then a = b.
- Useful to avoid division by zero a = b + (c/d) $[d<>0].

# Existence vs. value=0

- Be careful with eps values when protecting against divisions by 0. The two checking options there are:
  - 1: (a/b) $[b] problematic if b=eps
  - 2: (a/b) $[b<>0] works every time, protecting the division even if b=eps

| Entry | Name | Type | D |
|---|---|---|---|
| 2 | comprobacion | Set | |
| 1 | nulo | Set | |
| 3 | par | Parameter | |

| | eps | 0 |
|---|---|---|
| existe | 10 | 1 |
| distinto | 1 | 1 |

```
sets
nulo         /eps,0/
comprobacion /existe,distinto/
;
parameters
par(comprobacion,nulo)
;

par(comprobacion,nulo)=1;

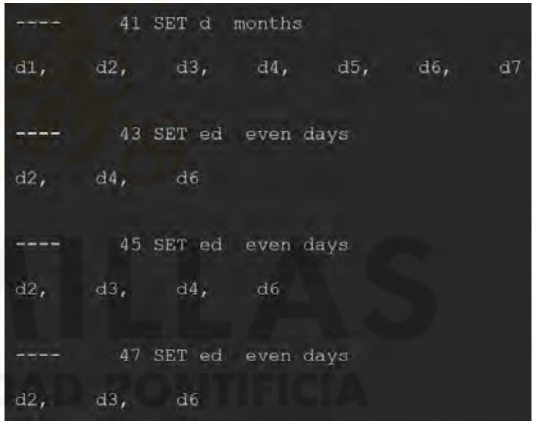
par('existe'  ,'eps')$[eps      ] = 10 ;

par('existe'  ,'0'  )$[0        ] = 10 ;
par('distinto','eps')$[eps <> 0] = 10 ;
par('distinto','0'  )$[0   <> 0] = 10 ;
```

# Dynamic sets

- Efficiency is strongly related to the use of dynamic sets
- Subsets of static sets whose content may change by assignments

```
sets d      months /d1*d7/
     ed(d) even days
display d;
ed(d) $[mod(ord(d),2) = 0] = yes;
display ed;
ed('d3') = yes;
display ed;
ed(d) $[ord(d) = 4] = no;
display ed;
```



```
----      41 SET d  months

d1,    d2,    d3,    d4,    d5,    d6,    d7

----      43 SET ed  even days

d2,    d4,    d6

----      45 SET ed  even days

d2,    d3,    d4,    d6

----      47 SET ed  even days

d2,    d3,    d6
```

- Fundamental elements in developing GAMS models
- Must be used systematically to avoid the formulation of superfluous equations, variables, or assignments

According to legend **Roland**'s Breach was cut by Count Roland with his sword **Durendal** to destroy that sword, after being defeated during the Battle of **Roncesvalles** in 778.

comillas.edu

COMILLAS UNIVERSIDAD PONTIFICIA ICAI ICADE CIHS

# Index shifting. Lag and lead


- `t=J,F,MAR,AP,MAY,JUN,JUL,AU,S,O,N,D`
  `vReserve(t-1) + pInflow(t) - vOutflow(t) =e= vReserve(t)`

- Vector values out of the domain are 0
  `0 + pInflow('J') - vOutflow('J') =e= vReserve('J')`

- Circular sequence of an index (++, --)
  `t= J,F,MAR,AP,MAY,JUN,JUL,AU,S,O,N,D`
  `vReserve(t--1) + pInflow( t ) - vOutflow( t ) =e= vReserve(t)`
  `vReserve('D' ) + pInflow('J') - vOutflow('J') =e= vReserve('J')`

- Inverted order sequence of `PP` index even though `t` is traversed in increasing order
  `PP(t+[card(t)-2*ord(t)+1])`

# Operations with sets

- Intersection

```
d(a) = b(a) * c(a)
```

- Union

```
d(a) = b(a) + c(a)
```

- Complementary

```
d(a) = NOT c(a)
```

- Difference

```
d(a) = b(a) - c(a)
```

# These constructs also exist in GAMS

```
loop (set,
) ;
```

```
while (condition,
) ;
```

```
repeat
until condition;
```

```
if (condition,
else
) ;
```

```
for (i=beginning to/downto end by increment,
) ;
```

```
Break
Continue
```

Jump out of the cycle

# Efficiency in GAMS code usage (loop)

```
set i / 1*2000 /
alias (i,ii)
parameter X(i,i)

loop ((i,ii),
   X(i,ii) = 4 ;
) ;
```

75.3 s

```
set i / 1*2000 /
alias (i,ii)
parameter X(i,i) ;

X(i,ii) = 4 ;
```

0.3 s

If you think you need a loop, Think again!

Among all the times a loop can be used, situations where they are needed are scarce.

# Efficiency in GAMS code usage (index order)

```
option Profile=10, ProfileTol=0.01

set i / 1*200 /
    j / 1*200 /
    k / 1*200 /
parameter X(k,j,i), Y(i,j,k) ;

Y(i,j,k) = 2 ;

X(k,j,i) = Y(i,j,k)
```

```
option Profile=10, ProfileTol=0.01

set i / 1*200 /
    j / 1*200 /
    k / 1*200 /
parameter X(i,j,k), Y(i,j,k) ;

Y(i,j,k) = 2 ;

X(i,j,k) = Y(i,j,k)
```

4.5 s

1.3 s

# Efficiency in GAMS code usage (condition checking)

```
scalar s_jnow;
sets i /1*1000/
     j /1*1000/
     k /1*1000/;
Parameter p_P    {i,j,k}
          p_Cond{i     };
p_Cond{'1'}=3;
s_jnow=jnow;


p_P{i,j,k}$[p_cond{i}=3]=1;
s_jnow = [(jnow-s_jnow)*86400];
```

Condition checked $i*j*k=10^9$ times
40.017s

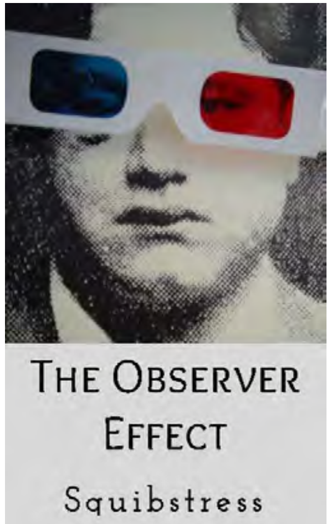
```
scalar s_jnow;
sets i /1*1000/
     j /1*1000/
     k /1*1000/;
Parameter p_P    {i,j,k}
          p_Cond{i     };
p_Cond{'1'}=3;
s_jnow=jnow;
set fix{i};
fix{i}$[p_Cond{i}=3]=yes;
p_P{fix{i},j,k}=1;
s_jnow = [(jnow-s_jnow)*86400];
```

Condition checked $i=10^6$ times
0.082s

Dynamic sets are your friends!!

# Observer effect

- Changes that the act of observation will make on a phenomenon being observed




```
option Profile=10, ProfileTol=0.01

set i / 1*2000 /
alias (i,ii)
parameter X(i,i)

loop ((i,ii),
   X(i,ii) = 4 ;
) ;
```

74.2 s

```
set i / 1*2000 /
alias (i,ii)
parameter X(i,i)

loop ((i,ii),
   X(i,ii) = 4 ;
) ;
```

72.9 s

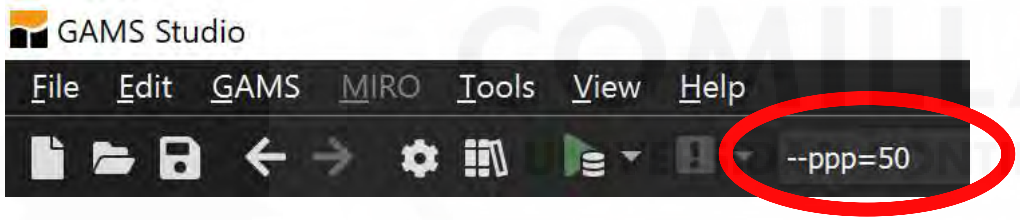
# Introducing flexibility

```
$SetGlobal ppp 100

parameter pDimension / %ppp% /
set u       / unit1*unit%ppp% /

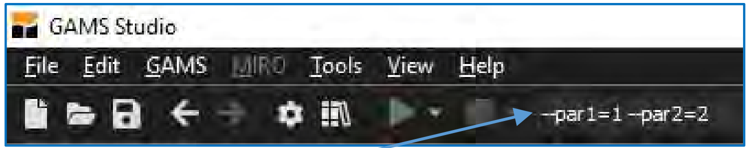
display pDimension, u
```

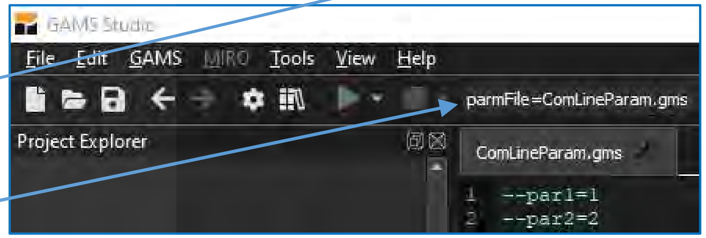
Alterative, modify the value from the command line.




ppp defined from the command line:
-> pDimension = 50
command line is empty
-> pDimension = 100

# Execution options and parameters



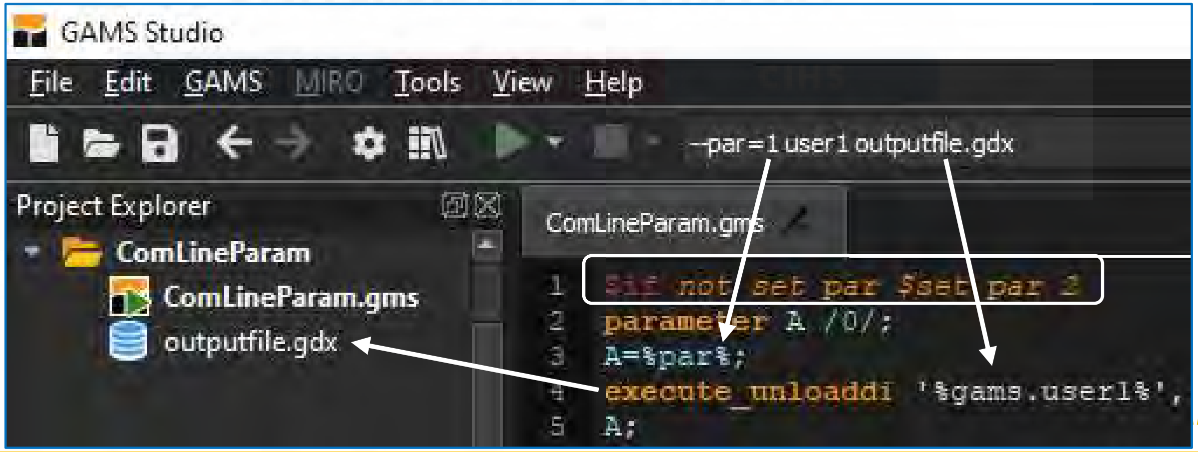

Passing parameters to a GAMS execution:

- Executing the model from the console (or from another software)
  - Syntax: "GAMS modelName.gms [parameters]"
  - GAMS directory must be included in the environment variables of the OS
- Executing the model from GAMS studio
  - Write parameters in the command line
- Define the parameters in a file, and send the file as a parameter to the execution from the command line or GAMS Studio: "parmFile=filename"

- Option parameters: parameters to control the execution such as the type of log (logOption=4), the depth of the profiling (profile=1), or defining a save file for the execution (-save the file.g00).
  - A complete list of options is here: https://www.GAMS.com/latest/docs/UG_GamsCall.html
- User-defined parameter: can store numeric values or strings. Definition "userN=value". Usage in the code %GAMS.userN% (substitute N by a number from 1 to 5).
- Double dash parameters: like user-defined parameters, but there is no limit; the names can be specified and can only store numeric values. Definition "--name=value". Usage in the code %name%.

# Execution options and parameters

- Both user-defined and double-dash parameters are substituted in the code by their values at compilation time.
- For example, during the compilation time, the following substitutions are performed:
  - %GAMS.user1% is substituted by outputfile.gdx (defined in the command line with user1)
  - %par% is substituted by 1 (defined in the command line with –par)
- It is possible to include a check in the code to assign default values when parameters are not defined in the command line. In the example, the first line establishes that when par is not set, it should be set equal to 2



# Detection of isolated subnetworks

```
$Phantom null

sets
   nd                nodes                     / node01 * node19 /
   ndref (nd)        current   reference node  / node01 /
   refnd (nd)        subset of reference nodes / node01 /
   nc    (nd)        current   connected nodes / null   /
   nod   (nd)        subset of connected nodes / null   /
   ln    (nd,nd)     lines
   subnet(nd,nd,nd)  subnetworks

parameters
   pAux1 auxiliary / 0 /
   pAux2 auxiliary / 1 /

alias (nd,n1,n2,ni,nf)

file out / out.gms / put out ;

* create a naïve network, a chain
ln(ni,nf) $[ni.pos = nf.pos-1] = yes ;

* break these links
ln('node10','node11') = no ;
ln('node15','node16') = no ;

* detection of isolated subnetworks

* for every subnetwork => max number of iterations
loop (n1 $[sum(nod(nd), 1) < card(nd)],

*  define the reference node for 2nd+ iterations
   ndref(nd) $[n1.pos > 1 and not refnd(nd) and nd.pos = smin(n2 $[not nod(n2)], n2.pos)] = yes ;

*  empty the set of connected nodes
   nc(nd    ) =  no ;
*  connect the reference node
   nc(ndref) = yes ;

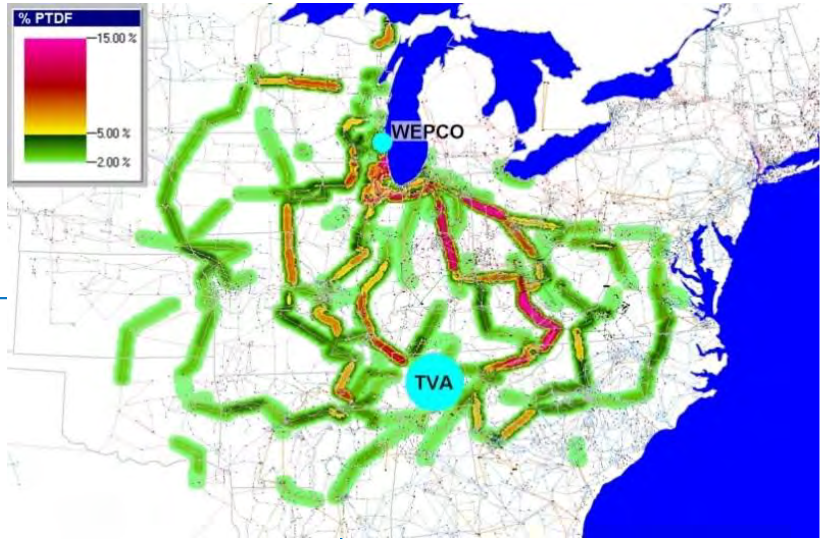
   pAux2 = 1 ;

*  for every node => max number of iterations
   loop (n2 $pAux2,
*     count the number of connected nodes
      pAux1  = sum[nc,                            1]       ;
*     add nodes to the set of already connected nodes
      nc(nf) $ sum[nc $[ln(nc,nf) or ln(nf,nc)], 1] = yes ;
*     count the new added nodes
      pAux2  $[sum[nc,                            1] - pAux1 = 0] = 0 ;

      if (pAux2 = 0,
*        subnetwork of the connected lines to a reference node
         subnet(ln(nc,nf),ndref) = yes ;
         subnet(ln(nf,nc),ndref) = yes ;
*        subnetwork of the connected nodes
         nod   (     nc          ) = yes ;
*        subnetwork of the reference nodes
         refnd (          ndref) = yes ;
      ) ;
   ) ;
   display subnet, nod, refnd ;

*  disconnect the reference node
   ndref(nd) = no ;
) ;
```

# Inverting a matrix, e.g., PTDF




```
set i / i1*i3 /

table a(i,i) matrix to invert
   i1 i2 i3
i1  1
i2     3
i3        5

parameter ainv(i,i) inverted matrix

execute_unload              'GDXForInverse.gdx' i a
executeTool.checkErrorLevel 'linalg.invert      i a ainv -gdxIn=GDXForInverse.gdx -gdxOut=GDXFromInverse.gdx'
execute_load                'GDXFromInverse.gdx'    ainv
execute              'del    GDXForInverse.gdx GDXFromInverse.gdx'
```

```
* computation of the susceptance matrix of the corridors
pYBUS(c2) = - sum[la(c2,cc), 1/pLineX(la)] ;
pYBUS(nf,ni) $c2(ni,nf) = pYBUS(ni,nf)  ;
pYBUS(nf,nf) =  - sum[ni, pYBUS(ni,nf)] ;
pYBUS(ni,ndref(nd)) = 0 ;
pYBUS(ndref(nd),nf) = 0 ;
pYBUS(ni,nd) $[not nc(ni)] = 0 ;
pYBUS(nd,nf) $[not nc(nf)] = 0 ;

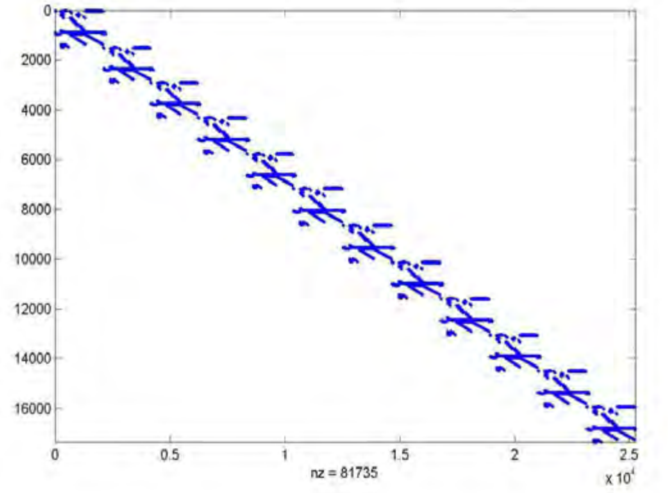
* obtaining the inverse of pYBUS and saving it into pYBUSInv
execute_unload        'GDXForInverse.gdx' noref pYBUS
executeTool.checkErrorLevel 'linalg.invert noref pYBUS pYBUSInv -gdxIn=GDXForInverse.gdx -gdxOut=GDXFromInverse.gdx'
execute_load          'GDXFromInverse.gdx'                  pYBUSInv
execute          'del    GDXForInverse.gdx              GDXFromInverse.gdx' ;

* computation of the PTDF matrix
pPTDF(la(ni,nf,cc),ngd) = [pYBUSInv(ni,ngd) - pYBUSInv(nf,ngd)]/pLineX(la) + eps ;
```


# Observe the constraint matrix

- It is important to know the estimated size of the optimization problem and its dependence considering the core elements
- It can be used for detecting formulation errors
- Use LimRow/LimCol
- Suitable to know the constraint matrix structure (GAMSChk)

```
option LP=GAMSChk
```




### D. Scaling - Maximum & Minimum Coefficients by Block -- Strip 1

| | | vTotalVCos | vProduct | vReserve | vSpillage | vArc | vENS | vPNS | vProductCo | vRACSlack | vIrrigatio | vCommitt | RHS MaxMin | Equ MaxMin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| eTotalVCos | Max | 1 | 26.07 | 4.6E-06 | 4.6E-06 | | 630 | 30 | 1.081 | 230 | | | 3.083E-03 | 630 |
| | Min | 1 | 1.154E-03 | 1.231E-09 | 1.231E-09 | | 0.4 | 0.3 | 8.75E-06 | 2.3 | | | 3.083E-03 | 1.231E-09 |
| eOpReserve | Max | | | | | | | 1 | | | | 0.765 | 1.975 | 1 |
| | Min | | | | | | | 1 | | | | 7.44E-04 | 0.375 | 7.44E-04 |
| eBalance | Max | | 1 | | | | 1 | | | | | | 10.76 | 1 |
| | Min | | 1 | | | | 1 | | | | | | 5.11 | 1 |
| eVirPlants | Max | | 29.36 | | | | | | | | | | 1 | 29.36 |
| | Min | | 1.308 | | | | | | | | | | 1 | 1.308 |
| eMaxOutput | Max | | 1344 | | | | | | | | | 1 | | 1344 |
| | Min | | 1.308 | | | | | | | | | 1 | | 1 |
| eProductCo | Max | | 1 | | | | | | 1 | | | | | 1 |
| | Min | | 1 | | | | | | 1 | | | | | 1 |
| eOtRateCon | Max | | | | | | | | 1.207E-02 | | | | 5.49E-03 | 1.207E-02 |
| | Min | | | | | | | | .0075 | | | | 1.458E-04 | .0075 |
| eFuelOtRat | Max | | 1.393E-02 | | | | | | | | | | 3.98E-03 | 1.393E-02 |
| | Min | | 7.59E-03 | | | | | | | | | | 1.104E-03 | 7.59E-03 |
| eFuelAvail | Max | | 0.815 | | | | | | | | | | .0005 | 0.815 |
| | Min | | 3.096E-02 | | | | | | | | | | .0005 | 3.096E-02 |
| eWtReserve | Max | | 0.512 | 1 | 1 | 1 | | | | | 1 | | 5.18 | 1 |
| | Min | | 1.478E-03 | 1 | 1 | 1 | | | | | 1 | | 6.66E-06 | 1.478E-03 |
| eMaxFlow | Max | | 0.512 | | | | | | | | | | 0.59 | 0.512 |
| | Min | | 1.478E-03 | | | | | | | | | | 3.28E-03 | 1.478E-03 |
| eRAC | Max | | | 4.6 | | | | | | 1 | | | 12.76 | 4.6 |
| | Min | | | .1231 | | | | | | 1 | | | 4 | .1231 |
| eSCmin | Max | | .0652 | | | | | | | | | | 0.575 | .0652 |
| | Min | | 2.678E-04 | | | | | | | | | | 0.486 | 2.678E-04 |
| eSCmax | Max | | 0.578 | | | | | | | | | | 1.08 | 0.578 |
| | Min | | 1.936E-05 | | | | | | | | | | 3.97E-04 | 1.936E-05 |
| eGmin | Max | | 1 | | | | | | | | | | 0.312 | 1 |
| | Min | | 1 | | | | | | | | | | 0.15 | 1 |
| eGmax | Max | | 1 | | | | | | | | | | 0.313 | 1 |
| | Min | | 1 | | | | | | | | | | 0.313 | 1 |
| Total Var | Max | 1 | 1344 | 4.6 | 1 | 1 | 630 | 30 | 1.081 | 230 | 1 | 1 | 12.76 | |
| | Min | 1 | 1.936E-05 | 1.231E-09 | 1.231E-09 | 1 | 0.4 | 0.3 | 8.75E-06 | 1 | 1 | 7.44E-04 | 6.66E-06 | |

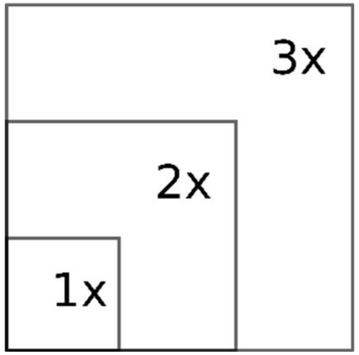
Source: MPES


# Analytical number of equations/variables for MHE

| | | Ecuaciones |
|---|---|---|
| Función objetivo | 1 | 1 |
| Balance demanda | $PP'S$ | 2000 |
| Producción térmica subperiodo | $P(P'-1)ST$ | 240000 |
| Producción hidráulica subperiodo | $P(P'-1)SH$ | 100000 |
| Bombeo hidráulica subperiodo | $P(P'-1)SH'$ | 20000 |
| Horas por escenario | $PST$ | 120000 |
| Horas mínimas y máximas por escenario | $2ST$ | 12000 |
| Horas mínimas y máximas todos escenarios | $2T$ | 240 |
| Balance embalses | $PSE$ | 50000 |
| Gasto del embalse por central | $PSEH$ | 250000 |
| Gasto total del embalse | $PSE$ | 50000 |
| Bombeo desde el embalse | $PSE'$ | 10000 |
| Desvío gasto embalse | $PSE$ | 50000 |
| Reserva final embalse | $SE$ | 2500 |
| Reserva mínima y máxima embalse | $2PSE$ | 100000 |
| TOTAL | | 1006741 |

| | | | Variables |
|---|---|---|---|
| Producción térmica | $pt^s_{pp't}$ | $PP'ST$ | 240000 |
| Producción hidráulica | $ph^s_{pp'h}$ | $PP'SH$ | 100000 |
| Producción del bombeo | $pb^s_{pp'h}$ | $PP'SH'$ | 20000 |
| Vertido del embalse | $v^s_{pe}$ | $PSE$ | 50000 |
| Reserva artificial del embalse | $ra^s_{pe}$ | $PSE$ | 50000 |
| Reserva del embalse | $r^s_{pe}$ | $PSE$ | 50000 |
| Defecto y exceso de entre reservas consecutivas | $dr^s_{pe}, er^s_{pe}$ | $2PSE$ | 100000 |
| Defecto y exceso de reserva | $dr^s_e, er^s_e$ | $2SE$ | 5000 |
| Defecto de reserva mínima y exceso de reserva máxima | $drm^s_{pe}, erm^s_{pe}$ | $2PSE$ | 100000 |
| Horas de funcionamiento por periodo | $hr^s_{pt}$ | $PST$ | 120000 |
| Defecto y exceso de horas de funcionamiento por período | $dhr^s_{pt}, ehr^s_{pt}$ | $2PST$ | 240000 |
| Defecto y exceso de horas de funcionamiento | $dhn^s_t, ehx^s_t$ | $2ST$ | 12000 |
| Gasto del embalse por central | $g^s_{peh}$ | $PSEH$ | 250000 |
| Gasto del embalse | $g^s_{pe}$ | $PSE$ | 50000 |
| Consumo del bombeo | $b^s_{pe}$ | $PSE'$ | 10000 |
| TOTAL | | | 1397000 |


# Scaling




- Solvers are powerful but not magic
- Input data and output results must be in commonly used units
- But internally, variables, equations, and parameters must be around 1 (i.e., from 0.01 to 100). The ratio of the largest to smallest matrix coefficient should be $< 10^5$
- Scaling can be done:
  - Manually (e.g., from MW to GW, from € to M€). Modelers can typically do better because they know the problem
  - Automatically by the language (ModelName.ScaleOpt=1)
  - By the solver (`ScaInd 1` in CPLEX, `ScaleFlag 2` in Gurobi)
- Especially useful in large-scale LP problems or NLP problems and/or when willing to get the dual variables
- The condition number measures the sensitivity of the solution of a system of linear equations to errors in the data.
  - It is the ratio between the largest and smallest eigenvalues
- Condition numbers $<10^6$ are good enough. Numerical problems arise for condition numbers $>10^8$ (*ill-conditioned*)
  - `Quality 1` in CPLEX
  - `Kappa 1` in Gurobi

**Feasibility, optimality, and integrality tolerances** should be less than the smallest meaningful coefficient in the model. Source: Gurobi

Models with **numerical issues can lead to undesirable results**: slow performance, wrong answers, or inconsistent behavior. Source: Gurobi

E. Klotz (2014) *"Identification, Assessment, and Correction of Ill-Conditioning and Numerical Instability in Linear and Integer Programs"* Tutorials in Operations Research 10.1287/educ.2014.0130


# How big is a big optimization problem

- Memory requirements for loading the model (solver)
  - 1 GB for every 1 million rows
- Memory requirements for solving the model (solver)
  - Depends on the difficulty of solving the model
    - The integrality gap is a good performance measure for MIP problems

# Avoid creation of superfluous constraints and variables

- Or how to achieve a compact formulation (small size of the constraint matrix or small density)
- Some “redundant” constraints can introduce a tighter model; see later
- However, introduce logical conditions (with a $ in GAMS) in the creation of equations or the use of variables to avoid superfluous ones
- Reduction rules: mathematical reasoning or common sense based on the problem context
  - Flows by nonexistent connections in a network
- Solvers can detect some of these superfluous equations/variables, but it is more efficient to avoid their creation (pre-processing)
- `Profile`, `ProfileTol`

# Compilation time vs. execution time

- GAMS compiles the entire code and then executes it.
- Some functions are only available for compilation or execution. In contrast, others have two versions (executing external code can be performed with "$call" (compilation time) or "execute" (execution time). Data from a GDX can be read with "$GDXin" and "$load" (compilation time) or execute_load (execution time), etc.
- It is essential to understand these two phases and make the proper choices.
  - For example, when using GDXs as input/output files, usually the read operation is performed during compilation (otherwise, GAMS would give a compilation error because sets and parameters are empty), and write operations are performed during execution (because until the model is executed, there are no values for the variables)

```
*Include files
$include 'file.gms'
```

```
*GDX reading: It is important to read
* domains (sets) before parameters
$GDXin 'file.gdx'
$loaddc s
$loaddc p_P
$GDXin
```

```
*Use code depending on parameters
*If condition is not satisfied code is not executed
* and not checked (this allows to have different
* definitions that may raise compilation error if
* the execution time version of the if was used)
$ifthen %executionType% == 1
p_P = 7;
$elseif %executionType% == 2
p_P = 0;
$endif
```

```
*execute external programs
*(included other GAMS instances)
$call 'GAMS nameModel.gms'
```

```
*Perform operations with data
* available at compilation time
set d 'days'  /1*7/;
$eval hours 24*card(d)
*nº hours = 24 * nº days
set h 'hours' /1*%hours%/;
```

# Compilation time vs. execution time
# Using $call for data processing

- We receive a GDX file (original.gdx) that contains a set *w* with some elements.
- For our execution, we want *w* to contain the original elements plus others (ad1*ad5).
- We want the code to be generic
  - Hardcoding the name of the original *w* elements is not acceptable.

| | Comp/exe time | mainFile.gms | updateW.gms | createGDX.gms (temporal file) |
|---|---|---|---|---|
| 1 | Comp | Executes updateW.gms | | |
| 2 | Comp | | Reads w from original.gdx | |
| 3 | Exe | | Writes createGDX.gms with all desired w elements | |
| 4 | Exe | | Executes createGDX.gms | |
| 5 | Exe | | | Creates w.gdx |
| 6 | Comp | Reads w.gdx with all desired w elements | | |

- This example gives an idea of the options that the combination of $call, execute and put can provide. However, a much simpler option is available for this case: use $onmulty to add new elements to *w*.

```
*mainFile.gms
set w;
$call 'GAMS updateW.gms'
$GDXin 'w.gdx'
$loaddc  w
$GDXin
```

```
*updateW.gms
set w;
$GDXin 'original.gdx'
$loaddc  w
$GDXin
file createGDX /'createGDX.gms'/;
put createGDX;
put "set w /" ;
loop(w,
    put w.tl "," ;
);
put "ad1*ad5/;"//;
put "execute_unloaddi 'w.gdx'"/;
put "w;"/;
putclose createGDX;
execute 'GAMS createGDX.gms'
```

```
*mainFile.gms
set w;
$GDXin 'original.gdx'
$load w
$GDXin

$onmulti

set w /ad1*ad5/;
```

# Compilation time vs. execution time
# Using $call for data processing

- Killing flies with a *cannon* ($call+put+execute) is not a good idea...
- However, it may come in handy to know how to fire a *cannon*

- Using a *cannon* would be needed, for example, if we wanted the resulting set *w* to have the following:
  - All original elements except some of them
  - All original and new elements alternated

| Order | Original w | New elements | Desired w |
|---|---|---|---|
| 1 | old1 | ad1 | old1 |
| 2 | old2 | ad2 | ad1 |
| 3 | old3 | ad3 | old2 |
| 4 | | | ad2 |
| 5 | | | old3 |
| 6 | | | ad3 |

| Order | Original w | New elements | Desired w |
|---|---|---|---|
| 1 | old1 | ad1 | old1 |
| 2 | old2 | ad2 | ad1 |
| 3 | old3 | ad3 | ad2 |
| 4 | | | ad3 |
| 5 | | | |
| 6 | | | |

# Compilation time vs. execution time

Remember the order of the phases:

1st Compilation

2nd Execution

Execution does not start until compilation is done!!

Order in which the code is written:

1. Include w1 as element of w
2. Create a GDX file with the elements of w
3. Include w2 as element of w

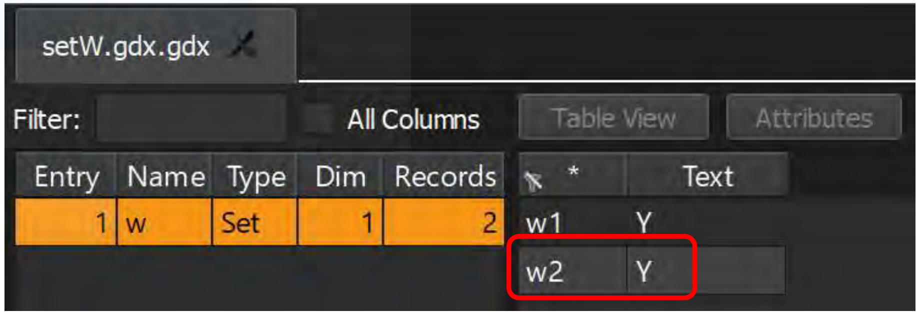
Order in which the operations are performed when the file is run by GAMS

1. Include w1 as element of w
2. Include w2 as element of w
3. Create a GDX file with the elements of w

```
$onmulti
*include w1 as element of w during COMPILATION
Set
w /w1/
;

*create a GDX with all elements of w during EXECUTION
execute_unload 'setW.gdx',
w
;

*include w2 as element of w during COMPILATION
Set
w /w2/
;
```



# Using workfiles

**`-save nameFile`** (or **`-s nameFile`**)
It is a command line parameter that creates a workfile (**`nameFile`**) containing a snapshot of the GAMS execution estate.

**`-restart nameFile`** (or **`-r nameFile`**)
Loads a workfile created with **`-save`**

Options A and B have the exact same effect:

Option A

```
GAMS OnlyFile.gms
```

Option B

```
GAMS FirstFile.gms  -s estado.g00
GAMS SecondFile.gms -r estado.g00
```

```
*OnlyFile.gms
set s/s1/ ;
execute_unload 's.gdx’, s ;
```

```
*FirstFile.gms
set s/s1/;
```

```
*SecondFile.gms
execute_unload 's.gdx’, s ;
```

# Deployment ready (using workfiles)

- Split formulation from data.
- Protect formulation confidentiality
- Secure Work Files
  - Control the access to symbol names
  - Link the model to a specific license

- Set declaration
- Parameter declaration
- Variables declaration
- Equations declaration
- Equations definition
- Model definition

Prepared to be deployed

- Include and manipulate input data: sets and parameters
- Bounds and initialization of variables
- Solve the optimization problem
- Output of the results

# Transportation model (ready for deployment)

Formulation.gms

```
sets
   I origins
   J destinations

parameters
   pA(i  ) origin capacity    [kg]
   pB(j  ) destination demand [kg]
   pC(i,j) per unit transportation cost [€]

variables
   vX(i,j) units transported   [kg]
   vCost   transportation cost [€]

positive variable vX

equations
   eCost          transportation cost            [€]
   eCapacity(i) maximum capacity of each origin [kg]
   eDemand   (j) demand supply at destination    [kg] ;

eCost ..         sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) .. sum[   j ,           vX(i,j)] =l= pA(i) ;
eDemand   (j) .. sum[ i,              vX(i,j)] =g= pB(j) ;

model mTransport / all /
```

Remaining.gms

```
$include Data.gms

solve mTransport using LP minimizing vCost
```

Data.gms

```
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /

parameters
   pA(i) origin capacity [kg]
       / VIGO      350
         ALGECIRAS 700 /

   pB(j) destination demand [kg]
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

table pC(i,j) per unit transportation cost [€]
           MADRID BARCELONA VALENCIA
VIGO        0.06      0.12      0.09
ALGECIRAS  0.05      0.15      0.11
```

Generate the runtime model
GAMS Formulation.gms **Save**=model

→ Send model.g00 →

Execute runtime model + data
GAMS Remaining.gms **Restart**=model

# Debugging (Using workfiles)

- Suppose we have a process where, instead of a single execution with the whole-time horizon, several executions are run using a loop.
- If we detect a problem in execution N, we could run the model till that execution and then abort it. This way, we can restart from a separate file and focus only on the specific execution without having to solve all the previous ones again.

```
*mainFile.gms

*Declarations: sets, parameters, variables, equations
*Definitions: equations, model
*Load data

*Split a week-long execution in 7 solves for 1 day.
set
ej 'executions' /1*7/
ejh(ej,h) 'relation between execution and hours';
ejh(ej,h)=yes$[(ej.ord-1)*24 < h.ord
                         and h.ord <= ej.ord*24];
loop(ej,
    if(ej.ord = N,
        abort ej;
    );
    ha(h)=yes$[ejh(ej,h)];
    solve modModelo minimizing v_fo using MIP;
*   Fix variables for ha
);
```

```
1) Add the abort to mainFile.gms
2) Run mainFile.gms with -s file.g00
3) Use debugFile.gms to debug, running it with -r file.g00
4) Repeat as many times as needed trying different options to find the error
```

```
*debugFile.gms

*modifications to try to find the error

loop(ej $[ej.ord >= N],
    ha(h)=yes$[ejh(ej,h)];
    solve modModelo minimizing v_fo using MIP;
*   Fix variables for ha
    if(ej.ord = N,
        abort ej;
    );
);
```

# Model log

- Open console from GAMSIDE for logging messages from the model
  - Code specific for Windows, UNIX/Linux/macOS

```
$set console
$if '%system.filesys%' == 'MSNT' $set console con
$if '%system.filesys%' == 'UNIX' $set console /dev/tty
$if '%console%.' == '.' abort 'console not recognized'

file console / '%console%' /

sets
   day day      / day01*day10 /
   sc  scenario / sc01* sc02 /

put console
loop ((day,sc),
   putclose 'Day ' day.tl:0 ' Scenario ' sc.tl:0 ' Elapsed Time ' [(jnow-jstart)*86400]:6:3 ' s' sleep(1)
) ;
```

```
$ifthen.MSNT '%system.filesys%' == 'MSNT'
   execute 'del pp.txt' ;
$endif.MSNT
$ifthen.UNIX '%system.filesys%' == 'UNIX'
   execute 'rm  pp.txt' ;
$endif.UNIX
```

Conditional compilation

# GAMS Code Conventions

- Must be defined in blocks. For example, a set and all its subsets should constitute one block in the sets section.
- Names are intended to be meaningful. Follow conventions
  - Items with the same name represent the same concept in different models
  - Units should be used in all definitions
  - Parameters are named pParameterName (e.g., pTotalDemand)
  - Variables are named vVariableName (e.g., vThermalOutput)
  - Equations are named eEquationName (e.g., eLoadBalance)
  - Use short set names (one or two letters) for easier reading
  - Alias duplicate the final letter (e.g., p, pp)
- Equations are laid out as clearly as possible, using brackets for readability
- In the case of variables, the blocks should be defined by meaning and not by variable type (Free (default), Positive, Negative, Binary, Integer, SOS1, SOS2, SemiCont, SemiInt). The objective function must be a free variable

Use of camelCase
(uppercases to differentiate)
Everything long and descriptive
except sets, that are compact

Scalars: s_name
Sets: n
Parameters: p_NameName
Variables: v_nameName
Equations: EQ_NameName
Models: modNameName

# Example model: general structure

- Model information
- Declarations: scalar, sets, alias, variables, parameter, and equations
- Equation definition
- Model definition
- Model solve configuration
- Data input (preprocessed)
- Data processing strictly associated with the model
- Variable limits/fix values
- Model solving
- Data output

# Example model: declarations

```
sets
sc                       '           scenarios'
n                        '           load levels'
gg                       '           units'
gt                       '           technologies'
co                       '           companies'
st                       '           ESS types'
g                 (    gg) '           active generating units '
t                 (    g ) '           thermal active units '
r                 (    g ) '           RES active units'
es                (    g ) '           ESS active units'
nr                (    g ) '           non-RES active units'
;

alias
(n,n2)
;

parameters
pTimeStep                  '[h]        TimeStep '
pStorageType      (    gg) '[-]        Storage type: 1-daily, 2-weekly, 3-monthly '
pENSCost                   '[MEUR/GWh] cost of energy not served '
pCO2Cost                   '[EUR/CO2t] cost of CO2 emission '
pDuration         (   n   ) '[h]        duration of load levels '
pScenProb         (sc     ) '[p.u.]     probabilities of scenarios '
pDemand           (sc,n   ) '[GW]       demand '
pOperReserveUp    (sc,n   ) '[GW]       operating reserve up '
pOperReserveDw    (sc,n   ) '[GW]       operating reserve down '
pEnergyInflows    (sc,n,gg) '[GW]       dynamic energy inflows '
pLinearVarCost    (    gg) '[MEUR/GWh] linear term variable cost '
pConstantVarCost  (    gg) '[MEUR/h]   constant term variable cost '
pStartUpCost      (    gg) '[MEUR]     startup cost '
pShutDownCost     (    gg) '[MEUR]     shutdown cost '
```

Vertical alignment improves readability

Definitions may become very large and require the use of the slider
Having the units first makes it easier to consult

Units

Definitions

# Example model: declarations

```
pRampUp            (      gg) '[GW/h]     ramp up rate '
pRampDw            (      gg) '[GW/h]     ramp down rate '
pCO2EmissionRate   (      gg) '[CO2t/MWh] emission rate '
pUpTime            (      gg) '[h]        minimum up time '
pDwTime            (      gg) '[h]        minimum down time '
pInitialInventory  (      gg) '[GWh]      initial ESS storage '
pEfficiency        (      gg) '[p.u.]     ESS efficiency '
pMaxPower2ndBlock  (sc,n,gg) '            pMaxPower - pMinPower '
pMinPower          (sc,n,gg)
pCycleTimeStep     (      gg)
pInitialUC         (      gg)
pInitialOutput     (      gg)
;
variables
vObjective                    '[MEUR]     Objective: total system cost'
;
positive variables
vTotalVCost                   '[MEUR]     Total system variable cost '
vTotalECost                   '[MEUR]     Total system emission cost '
vTotalOutput       (sc,n,gg) '[GW]        Total output of the unit '
vOutput2ndBlock    (sc,n,gg) '[GW]        Second block of the unit '
vReserveUp         (sc,n,gg) '[GW]        Operating reserve up '
vReserveDown       (sc,n,gg) '[GW]        Operating reserve down '
vESSInventory      (sc,n,gg) '[GWh]       ESS inventory '
vESSSpillage       (sc,n,gg) '[GWh]       ESS spillage '
vESSCharge         (sc,n,gg) '[GW]        ESS charge power '
vENS               (sc,n   ) '[GW]        Energy not served in node '
;
binary variables
vCommitment        (   n,gg) '            Commitment of the unit'
vStartUp           (   n,gg) '            StartUp of the unit'
vShutDown          (   n,gg) '            ShutDown of the unit'
;
```

# Example model: equation definition

```
equations
eTotalTCost                         '[MEUR]    Objective funciotn total system cost '
eTotalVCost                         '[MEUR]    total system variable cost '
eTotalECost                         '[MEUR]    total system emission cost '
eOperReserveUp      (sc,n    ) '[GW]      up operating reserve '
eOperReserveDw      (sc,n    ) '[GW]      down operating reserve '
eBalance            (sc,n    ) '[GW]      load generation balance '
eESSInventory       (sc,n,gg) '[GWh]     ESS inventory balance '
eMaxOutput2ndBlock (sc,n,gg) '[p.u.]    max output of the second block of a committed unit'
eMinOutput2ndBlock (sc,n,gg) '[p.u.]    min output of the second block of a committed unit'
eTotalOutput        (sc,n,gg) '[GW]      total output of a unit '
eUCStrShut          (   n,gg) '          relation among commitment startup and shutdown '
eRampUp             (sc,n,gg) '[p.u.]    maximum ramp up '
eRampDw             (sc,n,gg) '[p.u.]    maximum ramp down '
eMinUpTime          (   n,gg) '[h]       minimum up time '
eMinDownTime        (   n,gg) '[h]       minimum down time '
;

**************************************************** Objective function ****************************************************
*Objective function total system cost [MEUR]
eTotalTCost..
    vObjective =E= vTotalVCost + vTotalECost
;

*total system variable cost [MEUR]
eTotalVCost..
    vTotalVCost =E=   sum[(sc,n       ), pScenProb(sc) * pDuration(n) * pENSCost              * vENS         (sc,n  )]
                    + sum[(sc,n,nr(g)), pScenProb(sc) * pDuration(n) * pLinearVarCost   (g) * vTotalOutput(sc,n,g)]
                    + sum[(   n,nr(g)),                 pDuration(n) * pConstantVarCost(g) * vCommitment (   n,g)
                                                                     + pStartUpCost     (g) * vStartUp    (   n,g)
                                                                     + pShutDownCost    (g) * vShutDown   (   n,g)]
;
```

Align concepts within equations and for different but similar ones

# Example model: equation definition

```
*total system emission cost [MEUR]
eTotalECost..
    vTotalECost =E= sum[(sc,n,nr(g)), pScenProb(sc) * pCO2Cost * pCO2EmissionRate(g) * vTotalOutput(sc,n,g)]
;

**************************************************** Generating reserves/balance/inventory ****************************************************
*up operating reserve [GW]
eOperReserveUp(sc,n)$[pOperReserveUp(sc,n)]..
    sum[(nr(g)), vReserveUp  (sc,n,g)] =G= pOperReserveUp(sc,n)
;

*down operating reserve [GW]
eOperReserveDw(sc,n)$[pOperReserveDw(sc,n)]..
    sum[(nr(g)), vReserveDown(sc,n,g)] =G= pOperReserveDw(sc,n)
;

*load generation balance [GW]
eBalance(sc,n)..
    sum[(g), vTotalOutput(sc,n,g)] - sum[(es(g)), vESSCharge(sc,n,g)] + vENS(sc,n) =E= pDemand(sc,n)
;

*ESS inventory balance [GWh]
eESSInventory(sc,n,es(g))$[n.ord = pCycleTimeStep(g) or (n.ord > pCycleTimeStep(g) and mod(n.ord,pCycleTimeStep(g)) = 0)]..
      pInitialInventory(g)                    $[n.ord = pCycleTimeStep(g)]
    + vESSInventory(sc,n-pCycleTimeStep(g),g) $[n.ord > pCycleTimeStep(g) and mod(n.ord,pCycleTimeStep(g))= 0]
    + sum[(n2)$[n.ord-pCycleTimeStep(g)<=n2.ord and n2.ord<=n.ord], pDuration(n2)*(pEnergyInflows(sc,n2,g) - vTotalOutput(sc,n2,g)
    + pEfficiency(g)*vESSCharge(sc,n2,g))] =E= vESSInventory(sc,n,g) + vESSSpillage(sc,n,g)
;

**************************************************** Generating generation constraints ****************************************************
*max output of the second block of a committed unit [p.u.]
eMaxOutput2ndBlock(sc,n,nr(g))$[pOperReserveUp[sc,n]<>0 and pMaxPower2ndBlock(sc,n,g)<>0]..
    (vOutput2ndBlock(sc,n,g) + vReserveUp  (sc,n,g)) / pMaxPower2ndBlock(sc,n,g) =L= vCommitment(n,g)
;
```

Separate in different lines the “**header**” (equation name and sets) and the **definition** itself. This way you need less horizontal space and avoid the use of the horizontal slider, improving readability. Headers may become really large, as in the following real example:

```
EQ_CenQPsup(wa(w),insa(cen(ins)),kpsb(k,pa(p),sa(s),ba(b)),sala(sal))$[p.ord>s_kgrupo and wp(w,p) and sal.ord<=p_nsal(ins) and p.ord<=s_phoras]..
```

# Example model: equation definition

```
*min output of the second block of a committed unit [p.u.]
eMinOutput2ndBlock(sc,n,nr(g))$[pOperReserveDw[sc,n]<>0 and pMaxPower2ndBlock(sc,n,g)<>0]..
    (vOutput2ndBlock(sc,n,g) + vReserveDown(sc,n,g)) / pMaxPower2ndBlock(sc,n,g) =G= 0
;

*total output of a unit [GW]
eTotalOutput(sc,n,nr(g))..
    vTotalOutput(sc,n,g) / (pMinPower(sc,n,g) + 1$[pMinPower(sc,n,g)=0])
    =E= vCommitment(n,g) + vOutput2ndBlock(sc,n,g) / (pMinPower(sc,n,g) + 1$[pMinPower(sc,n,g)=0])
;

*relation among commitment startup and shutdown
eUCStrShut(n,nr(g))..
    vCommitment(n,g) - pInitialUC(g)$[n.ord = 1] - vCommitment(n-1,g)$[n.ord>1] =E= vStartUp(n,g) - vShutDown(n,g)
;

**************************************************** Generating ramps up/down ****************************************************
*maximum ramp up [p.u.]
eRampUp(sc,n,t(g))$(pRampUp(g)<>0 and pRampUp(g) < pMaxPower2ndBlock(sc,n,g))..
    (vOutput2ndBlock(sc,n,g) - max[pInitialOutput(g)-pMinPower(sc,n,g),0]$[n.ord=1] - vOutput2ndBlock(sc,n-1,g)$[n.ord>1]
    + vReserveUp  (sc,n,g)) / pDuration(n) / pRampUp(g) =L= vCommitment(n,g) - vStartUp(n,g)
;

*maximum ramp down [p.u.]
eRampDw(sc,n,t(g))$[pRampDw(g)<>0 and pRampDw(g) < pMaxPower2ndBlock(sc,n,g)]..
    (vOutput2ndBlock(sc,n,g) - max[pInitialOutput(g)-pMinPower(sc,n,g),0]$[n.ord=1] - vOutput2ndBlock(sc,n-1,g)$[n.ord>1]
    - vReserveDown(sc,n,g)) / pDuration(n) / pRampDw(g) =G= - vCommitment(n-1,g)$[n.ord>1] + vShutDown(n,g) - pInitialUC(g)$[n.ord=1]
;

**************************************************** Generating minimum up/down time ****************************************************
*minimum up time [h]
eMinUpTime(n,t(g))$[pUpTime(g) > 1 and n.ord >= pUpTime(g)]..
    sum[(n2)$[n.ord-pUpTime[g] <=n2.ord and n2.ord<=n.ord], vStartUp  (n2,g)] =L=     vCommitment[n,g]
;
```

For large codes copy the definition of the equations from the declaration to the definition

# Example model: model definition

```
*minimum down time [h]
eMinDownTime(n,t(g))$[pDwTime(g) > 1 and n.ord >= pDwTime(g)]..
    sum[(n2)$[n.ord-pDwTime(g) <= n2.ord and n2.ord<=n.ord], vShutDown(n2,g)] =L= 1 - vCommitment[n,g]
;                                       1                                    2                  3

model mUnits units equations
/
eESSInventory
eMaxOutput2ndBlock
eMinOutput2ndBlock
eTotalOutput
eUCStrShut
eRampUp
eRampDw
eMinUpTime
eMinDownTime
/
;

model mSDUC
/
eTotalTCost
eTotalVCost
eTotalECost
eOperReserveUp
eOperReserveDw
eBalance
mUnits
/
;
```

Use dynamic sets in headers instead rather than definitions. If you change them you need to change it only once, not 3 like in the example

It can be useful to define models containing a specific set of equations and the include them in larger models. Here all equations from “rampas” are included in “modelo”.

# Example model: model attributes and input data

```
* Branch and bound relative tolerance
mSDUC.optcr = 0.01;
* Limit execution time
mSDUC.reslim = 2*60;
* Ommit fixed variables
mSDUC.holdfixed = 1;
* Tolerance to take two numbers as equal.
* It is often useful to set a very small value but different from 0 when numerical errors due to rounding or decimal precisions occur.
* The typical error that pops up is "equation infeasible due to rhs" and when we go to see the error, 0 = very low value like 10e-13
* Then we set the infeasibility tolerance slightly above that value to tell it that in those cases it assumes that they are the same.
mSDUC.tolInfeas = 0.00001;

* Use cplex.opt option file
mSDUC.OptFile = 1;
```

Solver option file

```
$gdxIn datos.gdx
$load sc
$load n
$load gg
$load gt
$load co
$load st
$load g
$load t
$load r
$load es
$load nr
$load pTimeStep
$load pStorageType
$load pENSCost
$load pCO2Cost
$load pDuration
```

Use GDX as input and output data to isolate the model from the data processing

cplex.opt

| Key |
|---|
| lpmethod |
| StartAlg |
| subalg |
| mipstart |
| MemoryEmphasis |
| MemoryEmphasis comprime para usar menos memoria, eso puede causar mas uso de recursos |
| WorkMem |
| WorkMem es la cantidad de memoria usada, seria razonable subirlo hasta 4000 por ejemplo |
| RINSHeur |
| RINSHeur es cada cuanto pasa el heuristico RINS, a 0 es automatico, y a 100 es el vlor que tiene |
| nodefileind |
| nodefileind deja a cplx guardar info en el disco |
| feasopt =1 permite que si el modelo es infactible, busque la solución más próxima a la zona fa |
| feasopt=1 |
| iis=1 hace el análisis de irreductable infeasibility search (busca las ecuaciones, variables que ha |
| iis=1 |

--- All Options ---

Search Option...

| Option | Synonym | DefValue | Range | Type | Description |
|---|---|---|---|---|---|
| .benderspartition | | 0 | [0, 2147483647] | Integer | Benders partition |
| .divflt | | 0 | [-1e+299, 1e+2... | Double | solution pool range filte... |
| .feaspref | | 1 | [0, 1e+20] | Double | feasibility preference |
| .lazy | | 0 | {0, 1} | Boolean | Lazy constraints activati... |
| advind | | 1 | {0,1,2} | EnumInt | advanced basis use |
| aggcutlim | | 3 | [0, 2100000000] | Integer | aggregation limit for cu... |
| aggfill | | 10 | [0, 2147483647] | Integer | aggregator fill parameter |
| aggind | | -1 | [-1, 2100000000] | Integer | aggregator on/off |
| auxrootthreads | | 0 | [-1, 2100000000] | Integer | number of threads for a... |
| baralg | | 0 | {0,1,2,3} | EnumInt | algorithm selection |
| barcolnz | | 0 | [0, 2100000000] | Integer | dense column handling |
| barcrossalg | | 0 | {0,1,2} | EnumInt | barrier crossover method |
| bardisplay | | 1 | {0,1,2} | EnumInt | progress display level |
| barepcomp | | 1e-08 | [1e-12, 1e+75] | Double | convergence tolerance |
| bargrowth | | 1e+12 | [1, 1e+75] | Double | unbounded face detecti... |
| baritlim | | 2147483647 | [0, 2147483647] | Integer | iteration limit |
| barmaxcor | | -1 | [-1, 2147483647] | Integer | maximum correction li... |
| barobjrng | | 1e+20 | [0, 1e+75] | Double | maximum objective fu... |
| barorder | | 0 | {0,1,2,3} | EnumInt | row ordering algorithm ... |
| barqcpepcomp | | 1e-07 | [1e-12, 1e+75] | Double | convergence tolerance ... |
| barstartalg | | 1 | {1,2,3,4} | EnumInt | barrier starting point al... |
| bbinterval | | 7 | [0, 2147483647] | Integer | best bound interval |
| bendersfeascuttol | | 1e-06 | [1e-09, 0.1] | Double | Tolerance for whether a ... |
| bendersoptcuttol | | 1e-06 | [1e-09, 0.1] | Double | Tolerance for optimality... |
| benderspartitioninstage | | 0 | {0, 1} | Boolean | Benders partition throu... |
| bendersstrategy | | 0 | {-1,0,1,2,3} | EnumInt | Benders decomposition... |
| bndrng | | | | StrList | do lower / upper bound... |
| bndstrenind | | -1 | {-1,0,1} | EnumInt | bound strengthening |
| bqpcuts | | 0 | {-1,0,1,2,3} | EnumInt | boolean quadric polyto... |
| brdir | | 0 | {-1,0,1} | EnumInt | set branching direction |
| bttol | | 1 | [0, 1] | Double | backtracking limit |
| calcqcpduals | | 1 | {0,1,2} | EnumInt | calculate the dual value... |

# Example model: input data

```
$load pScenProb
$load pDemand
$load pOperReserveUp
$load pOperReserveDw
$load pEnergyInflows
$load pLinearVarCost
$load pConstantVarCost
$load pStartUpCost
$load pShutDownCost
$load pRampUp
$load pRampDw
$load pCO2EmissionRate
$load pUpTime
$load pDwTime
$load pInitialInventory
$load pEfficiency
$load pMaxPower2ndBlock
$load pMinPower
$load pCycleTimeStep
$load pInitialUC
$load pInitialOutput
$gdxIn
;

*set active sets for debuging
*g(gg)$[condition] = yes;
```

Limit the active set elements when you are debugging, and activate them all for standard executions

# Example model: variable fixing, model solving and data output

```
* fixing the ESS inventory at the last load level at the end of the time scope
vESSInventory.fx(sc,n,es(g))$[n.ord = card(n)] = pInitialInventory(g);

*definition of the time-steps leap to observe the stored energy at ESS
pCycleTimeStep(es(g))$[pStorageType(g)=1] = 1;
pCycleTimeStep(es(g))$[pStorageType(g)=2] = trunc( 24/pTimeStep);
pCycleTimeStep(es(g))$[pStorageType(g)=3] = trunc(168/pTimeStep);

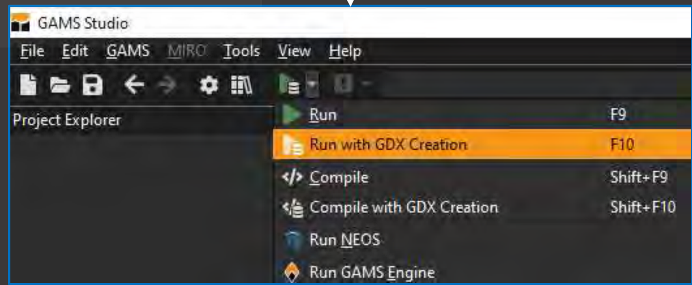
*fixing the ESS inventory at the end of the following pCycleTimeStep (weekly, yearly), i.e., for daily ESS is fixed at the end of the week, for weekly/monthly ESS is fixed at the end of the year
vESSInventory.fx(sc,n,es(g))$[pStorageType(g)=1 and mod(n.ord, 168/pTimeStep) = 0] = pInitialInventory(g);
vESSInventory.fx(sc,n,es(g))$[pStorageType(g)=2 and mod(n.ord,8736/pTimeStep) = 0] = pInitialInventory(g);
vESSInventory.fx(sc,n,es(g))$[pStorageType(g)=3 and mod(n.ord,8736/pTimeStep) = 0] = pInitialInventory(g);

solve mSDUC using MIP minimizing vObjective ;

*generate GDX with certain output data from the model
*no need to specify parameter/variable sets, it includes them automatically
execute_unloaddi 'out.gdx',
vCommitment
vTotalVCost
vTotalECost
vTotalOutput
;
```

Use GDX as input and output data to isolate the model from the data processing

If you run the model with GDX creating a GDX file with the same name as the GAMS file is created containing all the information



# Example model: solver related information

```
*Useful solve information:
* in complex codes this information can be used to condition the development of the program, for example with multiple executions or loops).
*Total time
display mSDUC.etSolve;
*Solver time
display mSDUC.resUsd;
*Solver termination
* 1-Normal Completion 6-Capability Problems 11-Internal Solver Failure
* 2-Iteration Interrupt 7-Licensing Problems 13-System Failure
* 3-Resource Interrupt 8-User Interrupt
* 4-Terminated By Solver 9-Setup Failure
* 5-Evaluation Interrupt 10-Solver Failure
display mSDUC.solveStat;
*Model status code
* 1-Optimal 6-Intermediate Infeasible 11-Licensing Problem 16-Solved
* 2-Locally Optimal 7-Intermediate Nonoptimal 12-Error Unknown 17-Solved Singular
* 3-Unbounded 8-Integer Solution 13-Error No Solution 18-Unbounded - No Solution
* 4-Infeasible 9-Intermediate Non-Integer 14-No Solution Returned 19-Infeasible - No Solution
* 5-Locally Infeasible 10-Integer Infeasible 15-Solved Unique
display mSDUC.modelStat;
*Number of discrete variables of the problem
display mSDUC.numDVar;
*Number of equations
display mSDUC.numEqu;
*Number of variables
display mSDUC.numVar;
scalars
s_optcr "Optcr achieved"
;
*GAMS Optcr
s_optcr = (100 * abs(mSDUC.objest - mSDUC.objval) / max(abs(mSDUC.objest),abs(mSDUC.objval)))$[max(abs(mSDUC.objest),abs(mSDUC.objval))];
display s_optcr;
*Cplex Optcr
s_optcr = (100 * abs(mSDUC.objest - mSDUC.objval) /(1e-10+abs(mSDUC.objval)))$[(1e-10+abs(mSDUC.objval))];
display s_optcr;
```

# Example model 2: description and declarations (sets and variables

```
$Title Example model: GAMS version of the pyomo example https://gitlab001.iit.comillas.edu/pdeotaola/Ejemplo_Optimizacion_Python-Pyomo

*    Developed by
*    Paulo Brito Pereira and Pedro de Otaola Arca
*    Instituto de Investigacion Tecnologica
*    Escuela Tecnica Superior de Ingenieria - ICAI
*    UNIVERSIDAD PONTIFICIA COMILLAS
*    Alberto Aguilera 23
*    28015 Madrid, Spain
*    May 2022

sets
t                 "        Time periods"
ta            (   t) "        Active time periods, can be useful to split a long run into several sequential ones, or for development, to run only one piece even if there is data for a very long horizon"
g                 "        Generation units"
ga            (g   ) "        Active generation units"
;
*alias should be defined here
variables
v_fob             "[M€]     Objective function value"
;
positive variables
v_p           (g, t) "[GW]     Power output above the minimum of the generator"
v_t           (g, t) "[GW]     Total power output of the generator"
v_ct          (g, t) "[M€]     Generation cost"
;
binary variables
v_v           (g, t) "{0,1}    Commitment status"
v_y           (g, t) "{0,1}    Start up decision"
v_z           (g, t) "{0,1}    Shut down decision"
;
```

Definitions may become very large and require the use of the slider
Having the units first makes it easier to consult

Units

Definitions

Define dynamic sets as the active elements of the static sets. Even if you don't use them, they can be very useful to isolate problems when debugging

Vertical alignment improves readability

# Example model 2: declarations (parameters and equations)

```
parameters
p_Gcaco      (g   ) "[M€/h]   Commitment cost"
p_Gcvar      (g   ) "[M€/GWh] Variable cost"
p_Gcarr      (g   ) "[M€]     Start up cost"
p_Gcpar      (g   ) "[M€]     Shut down cost"
p_Gpmn       (g   ) "[GW]     Minimum power output"
p_Gpmx       (g   ) "[GW]     Maximum power output"
p_Gei        (g   ) "[-]      Initial commitment status: {0} off {1} on"
p_GPini      (g   ) "[GW]     Initial power output"
p_Grs        (g   ) "[GW]     Maximum power output increase"
p_Grb        (g   ) "[GW]     Maximum power output decrease"
p_Precio     (   t) "[M/€GWh] Electricity price"
;
equations
EQ_FObj             "Objective function: cost-income minimization"
EQ_CostT     (g, t) "Generation cost"
EQ_PotAcoT   (g, t) "Generation units total power output"
EQ_AcoParPmn (g, t) "Commitment: stop at minimum power"
EQ_AcoArrPmn (g, t) "Commitment: start at minimum power"
EQ_AcoPar    (g, t) "Coherence between commitment status and start up and shut down decisions"
EQ_RampSub   (g, t) "Limit increase in power output"
EQ_RampBaj   (g, t) "Limit decrease in power output"
EQ_Dummy     (g, t) "Dummy: used for explanation"
;
```

# Example model 2: equation definition

```
*Objective function: cost-income minimization
EQ_FObj..
    v_fob =E= sum[{ga{g},ta{t}},v_ct{g,t} - v_t{g,t} * p_Precio{t}];
*Generation cost
EQ_CostT    {ga{g},ta{t}}..
    v_ct{g,t} =E=   p_Gcarr{g} * v_y{g,t}
                  + p_Gcpar{g} * v_z{g,t}
                  + p_Gcaco{g} * v_v{g,t}
                  + p_Gcvar{g} * v_p{g,t};
*Generation units total power output
EQ_PotAcoT  {ga{g},ta{t}}..
    v_t{g,t} =E= p_Gpmn{g} * v_v{g,t} +  v_p{g,t};
*Commitment: stop at minimum power
EQ_AcoParPmn{ga{g},ta{t}}$[ta(t-1)]..
    v_p{g,t-1} =L= (p_Gpmx{g} - p_Gpmn{g}) * (v_v{g,t-1} - v_z{g,t});
*Commitment: start at minimum power
EQ_AcoArrPmn{ga{g},ta{t}}..
    v_p{g,t}   =L= (p_Gpmx{g} - p_Gpmn{g}) * (v_v{g,t}   - v_y{g,t});
*Coherence between commitment status and start up and shut down decisions
EQ_AcoPar   {ga{g},ta{t}}..
    v_y{g,t} - v_v{g,t} - v_z{g,t} + v_v{g,t-1}$[t.ord>1]
                                  + p_Gei{g}  $[t.ord=1] =E= 0;
*Limit increase in power output
EQ_RampSub  {ga{g},ta{t}}..
    + v_p{g,t} - v_p{g,t-1}$[t.ord>1]
               - p_GPini{g}$[t.ord=1] =L= p_Grs{g};
*Limit decrease in power output
EQ_RampBaj  {ga{g},ta{t}}..
    - v_p{g,t} + v_p{g,t-1}$[t.ord>1]
       1       + p_GPini{g}$[t.ord=1] =L= p_Grb{g};
*Dummy          2    3                    4
EQ_Dummy{ga(g), ta(t)}..
    v_ct{g,t} =G= 0;
```

For large codes copy the definition of the equations from the declaration to the definition

Separate in different lines the "header" (equation name and sets) and the definition itself. This way you need less horizontal space and avoid the use of the horizontal slider, improving readability. Headers may become really large, as in the following real example:

```
$ifthen %exla% == 1
EQ_CenQPsup{wa{w},insa{cen{ins}},kpsb{k,pa{p},sa{s},ba{b}},sala{sal}}$[p.ord>s_kgrupo and wp{w,p} and sal.ord<=p_nsal{ins} and p.ord<=s_phoras]..
$elseif %excom% == 1
EQ_CenQPsup{wa{w},insa{cen{ins}},ka{k}, pa{p},sa{s},ba{b} ,sala{sal}}$[p.ord>s_kgrupo and k.ord=p.ord ]..
$endif
```

Align concepts within equations and for different but similar ones

Use dynamic sets in headers instead rather than definitions. If you change them you need to change it only once, not 4 like in the example

# Example model 2: model definition and attributes

```
*Ramp related equations
model rampas
/
EQ_RampSub
EQ_RampBaj
EQ_Dummy
/;
* Complete optimization model
model modelo
/
EQ_FObj
EQ_CostT
EQ_PotAcoT
EQ_AcoParPmn
EQ_AcoArrPmn
EQ_AcoPar
rampas
-EQ_Dummy
/;

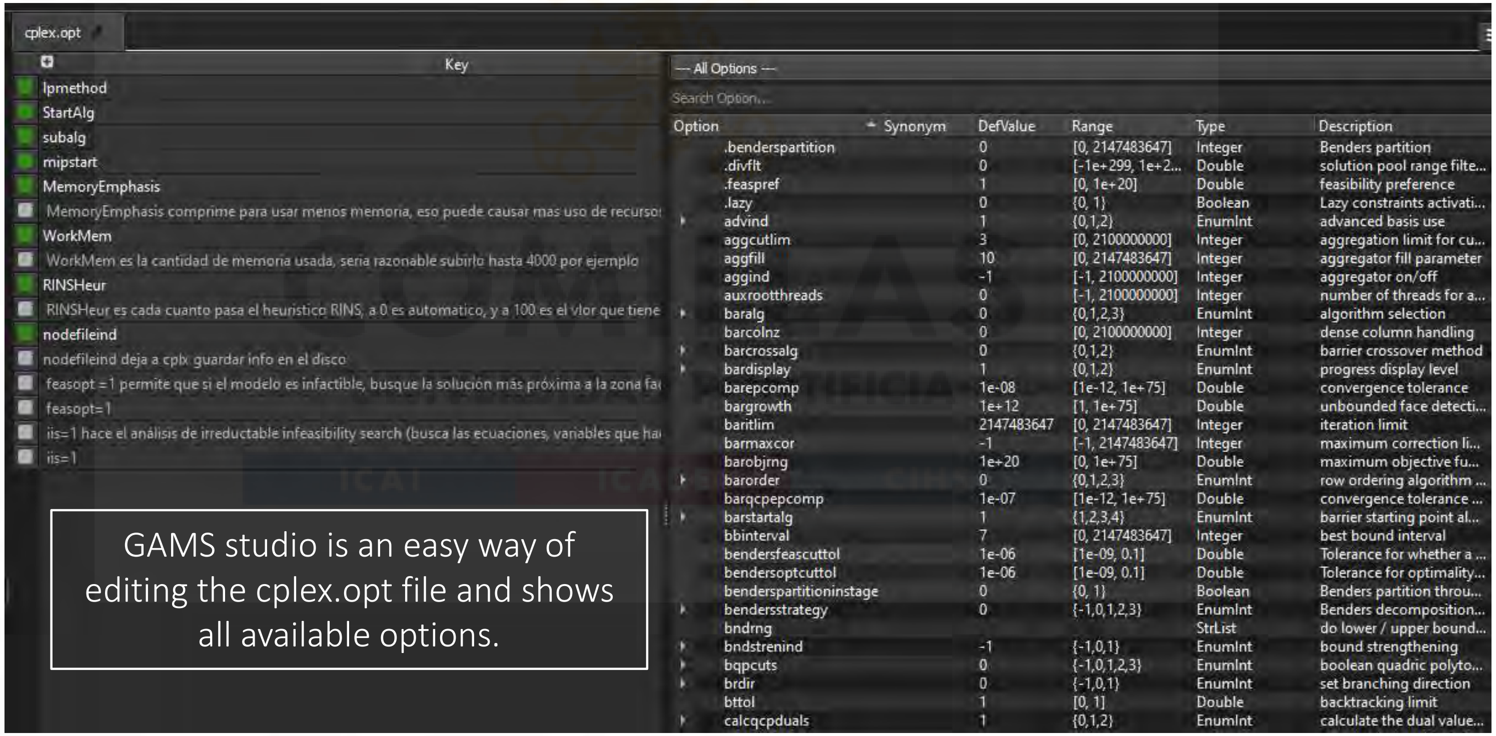
* Branch and bound relative tolerance
modelo.optcr     = 0.01;
* Limit execution time
modelo.reslim    = 2*60;
* Ommit fixed variables
modelo.holdfixed = 1;
* Tolerance to take two numbers as equal.
* It is often useful to set a very small value but different from 0 when numerical errors due to rounding or decimal precisions occur.
* The typical error that pops up is "equation infeasible due to rhs" and when we go to see the error, 0 = very low value like 10e-13
* Then we set the infeasibility tolerance slightly above that value to tell it that in those cases it assumes that they are the same.
modelo.tolInfeas = 0.00001;
```

It can be useful to define models containing a specific set of equations and the include them in larger models. Here all equations from “rampas” are included in “modelo”.

There is also the possibility to eliminate an equation by using the symbol “-”. The equation “EQ_Dummy” is included in “rampas”, and therefore included in “modelo”, but as we don’t want that equation in “modelo” we use: “- EQ_Dummy”

# Example model 2: solver option file

```
* Use cplex.opt option file
modelo.OptFile    = 1;
```




GAMS studio is an easy way of editing the cplex.opt file and shows all available options.

# Example model 2: data input, model solving and data output

```
*Loading input data from a GDX file
$GDXin 'entrada.gdx'
$loaddc g
$loaddc t
$loaddc p_Gcaco
$loaddc p_Gcvar
$loaddc p_Gcarr
$loaddc p_Gcpar
$loaddc p_Gpmn
$loaddc p_Gpmx
$loaddc p_Gei
$loaddc p_GPini
$loaddc p_Grs
$loaddc p_Grb
$loaddc p_Precio
$GDXin;

*activate the complete sets to make a full run with everything
ga(g) = yes;
ta(t) = yes;

*Variable limit and fixed values should be performed here

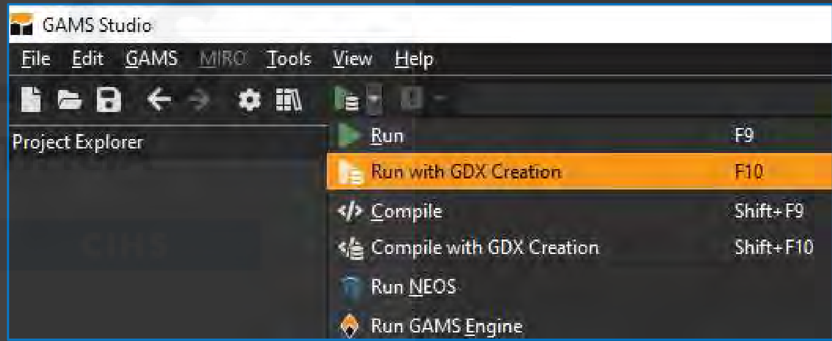
SOLVE modelo  minimizing v_fob using MIP;

*generate GDX with certain output data from the model
*no need to specify parameter/variable sets, it includes them automatically
execute_unloaddi 'salida.gdx',
v_t
v_ct
;
```

Use GDX as input and output data to isolate the model from the data processing

Limit the active set elements when you are debugging, and activate them all for standard executions

If you run the model with GDX creating a GDX file with the same name as the GAMS file is created containing all the information



# Example model 2: solver related information

```
*Useful solve information:
* in complex codes this information can be used to condition the development of the program, for example with multiple executions or loops).
*Total time
display modelo.etSolve;
*Solver time
display modelo.resUsd;
*Solver termination
* 1-Normal Completion      6-Capability Problems       11-Internal Solver Failure
* 2-Iteration Interrupt    7-Licensing Problems        13-System Failure
* 3-Resource Interrupt     8-User Interrupt
* 4-Terminated By Solver   9-Setup Failure
* 5-Evaluation Interrupt  10-Solver Failure
display modelo.solveStat;
*Model status code
* 1-Optimal                6-Intermediate Infeasible   11-Licensing Problem     16-Solved
* 2-Locally Optimal        7-Intermediate Nonoptimal   12-Error Unknown         17-Solved Singular
* 3-Unbounded              8-Integer Solution          13-Error No Solution     18-Unbounded - No Solution
* 4-Infeasible             9-Intermediate Non-Integer  14-No Solution Returned  19-Infeasible - No Solution
* 5-Locally Infeasible    10-Integer Infeasible        15-Solved Unique
display modelo.modelStat;
*Number of discrete variables of the problem
display modelo.numDVar;
*Number of equations
display modelo.numEqu;
*Number of variables
display modelo.numVar;
scalars
s_optcr       "Optcr achieved"
;
*GAMS Optcr
s_optcr = (100 * abs(modelo.objest - modelo.objval) / max(abs(modelo.objest),abs(modelo.objval)))$[max(abs(modelo.objest),abs(modelo.objval))];
display s_optcr;
*Cplex Optcr
s_optcr = (100 * abs(modelo.objest - modelo.objval) /(1e-10+abs(modelo.objval)))$[(1e-10+abs(modelo.objval))];
display s_optcr;
```

# Example model 2: equation documentation (outside the model)

```
*Objective function: cost-income minimization
$EQ_FObj$
```math
    v_fob = \sum_{g \in ga, t \in ta}[v_ct_{g,t} - v_t_{g,t} *  p_Precio_{t}]
```

*Generation cost
$EQ_CostT: g \in ga, t \in ta$
```math
    v_ct_{g,t} =   p_Gcarr_{g} * v_y_{g,t}
                  + p_Gcpar_{g} * v_z_{g,t}
                  + p_Gcaco_{g} * v_v_{g,t}
                  + p_Gcvar_{g} * v_p_{g,t}
```

*Generation units total power output
$EQ_PotAcoT: g \in ga, t \in ta$
```math
    v_t_{g,t} = p_Gpmn_{g} * v_v_{g,t} + v_p_{g,t}
```

*Commitment: stop at minimum power
$EQ_AcoParPmn: g \in ga, t \in ta \ if\ [t-1 \in ta]$
```math
    v_p_{g,t-1} \leq (p_Gpmx_{g} - p_Gpmn_{g}) * (v_v_{g,t-1} - v_z_{g,t})
```

*Commitment: start at minimum power
$EQ_AcoArrPmn: g \in ga, t \in ta$
```math
    v_p_{g,t} \leq (p_Gpmx_{g} - p_Gpmn_{g}) * (v_v_{g,t} - v_y_{g,t})
```

*Coherence between commitment status and start up and shut down decisions
$EQ_AcoPar: g \in ga, t \in ta$
```math
    v_y_{g,t} - v_v_{g,t} - v_z_{g,t} + v_v_{g,t-1}\$[t.ord>1]
                                          + p_Gei_{g}  \$[t.ord=1] = 0
```

*Limit increase in power output
$EQ_RampSub: g \in ga, t \in ta$
```math
    + v_p_{g,t} - v_p_{g,t-1}\$[t.ord>1]
                 - p_GPini_{g}\$[t.ord=1] \leq p_Grs_{g}
```

*Limit decrease in power output
$EQ_RampBaj: g \in ga, t \in ta$
```math
    - v_p_{g,t} + v_p_{g,t-1}\$[t.ord>1]
                 + p_GPini_{g}\$[t.ord=1] \leq p_Grb_{g}
```

Documentation of the equations in markdown using visual studio code

*Objective funciton: cost-income minimization

$EQ_FObj$

$$v\_fob = \sum_{g \in ga, t \in ta} [v\_ct_{g,t} - v\_t_{g,t} * p\_Precio_t]$$

*Generation cost

$EQ_CostT : g \in ga, t \in ta$

$$v\_ct_{g,t} = p\_Gcarr_g * v\_y_{g,t} + p\_Gcpar_g * v\_z_{g,t} + p\_Gcaco_g * v\_v_{g,t} + p\_Gcvar_g * v\_p_{g,t}$$

*Generation units total power output

$EQ_PotAcoT : g \in ga, t \in ta$

$$v\_t_{g,t} = p\_Gpmn_g * v\_v_{g,t} + v\_p_{g,t}$$

*Commitment: stop at minimum power

$EQ_AcoParPmn : g \in ga, t \in ta\; if\; [t-1 \in ta]$

$$v\_p_{g,t-1} \leq (p\_Gpmx_g - p\_Gpmn_g) * (v\_v_{g,t-1} - v\_z_{g,t})$$

*Commitment: start at minimum power

$EQ_AcoArrPmn : g \in ga, t \in ta$

$$v\_p_{g,t} \leq (p\_Gpmx_g - p\_Gpmn_g) * (v\_v_{g,t} - v\_y_{g,t})$$

*Coherence between commitment status and start up and shut down decisions

$EQ_AcoPar : g \in ga, t \in ta$

$$v\_y_{g,t} - v\_v_{g,t} - v\_z_{g,t} + v\_v_{g,t-1}\$[t.ord > 1] + p\_Gei_g\$[t.ord = 1] = 0$$

*Limit increase in power output

$EQ_RampSub : g \in ga, t \in ta$

$$+v\_p_{g,t} - v\_p_{g,t-1}\$[t.ord > 1] - p\_GPini_g\$[t.ord = 1] \leq p\_Grs_g$$

*Limit decrease in power output

$EQ_RampBaj : g \in ga, t \in ta$

$$-v\_p_{g,t} + v\_p_{g,t-1}\$[t.ord > 1] + p\_GPini_g\$[t.ord = 1] \leq p\_Grb_g$$

$EQ_Dummy : g \in ga, t \in ta$

$$v\_ct_{g,t} \geq 0$$
```

# Example model 2: time considerations for equation definitions and performing partial executions

```
*The equation has v_p and v_v of one time period and v_z of the next, the straightforward definition would be:
EQ_AcoParPmn(g,ta(t))$[t.ord < card(t)]....
    v_p[g,t] <= (p_Gpmx[g] - p_Gpmn[g]) * (v_v[g,t] - v_z[g,t+1]);
*However, it may be a better idea to use past time indices instead of future time indices and write the equation as follows:
EQ_AcoParPmn{ga{g},ta{t}}$[ta(t-1)]..
    v_p{g,t-1} =L= (p_Gpmx{g} - p_Gpmn{g}) * (v_v{g,t-1} - v_z{g,t});
The reason is that if the entire period being executed were split into several sequential runs, for the executions that were not the first one, the value of the variables v_p and v_v would be available in the last period of the previous execution. That would avoid the need to fix the value of the variable v_z in the first period of the subsequent execution.
```

```
sets
ej           "Executions in which the time horizon is to be split" /ej1*ej3/
ejt (ej,t)   "Time periods of each execution"
;

ejt(ej,t) = yes$[(ej.ord-1)*card(t)/card(ej)<t.ord and t.ord<=ej.ord*card(t)/card(ej)];

ga(g) = yes;
loop(ej,
    ta(t) = yes$[ejt(ej,t)];
    SOLVE modelo  minimizing v_fob using MIP;
    v_p.fx {g,ta{t}} = v_p.l {g,t};
    v_t.fx {g,ta{t}} = v_t.l {g,t};
    v_ct.fx{g,ta{t}} = v_ct.l{g,t};
    v_v.fx {g,ta{t}} = v_v.l {g,t};
    v_y.fx {g,ta{t}} = v_y.l {g,t};
    v_z.fx {g,ta{t}} = v_z.l {g,t};
);
```

1. Define active periods
2. Solve the model
3. Fix variables for optimized periods

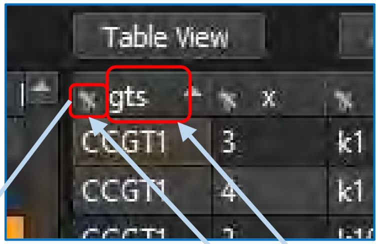
# Data in and out of the model

- Data is one of the primary sources of problems. During development, an excellent practice isolates the model from the data processing.
- Data processing outside the model
  - Construct parameters from other parameters
  - Scaling parameters to standardize units. All parameters should be in the same units, and all scaling should be performed previously. Never hardcode scaling in equations!!
- Data processing inside model
  - Scaling parameter for the execution particularities. For example, the associated cost of a monthly decision can be scaled to 7/30 if the model executes just a week horizon.
- If you are forced to perform data processing in the model, do it together and in a separate file (use an include).

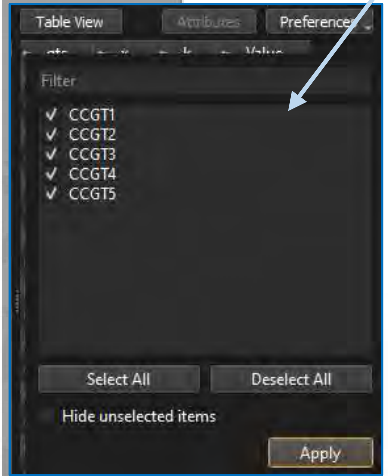
Personal recommendation: if you need data processing in GAMS, do it in a separate file and a separate process (using $call) that builds a GDX to be read from the main file. This way, you write the input data to disk and reread it. There are more efficient ways. However, having a single file that is easy to consult with all the input data is convenient for debugging.

A good option is parametrizing the process so that when you are debugging, that GDX file is created, and in standard executions, a more direct data input method is used.

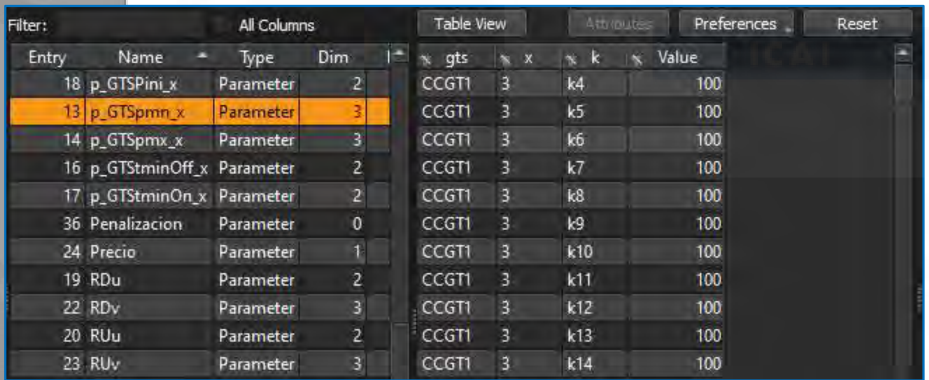
# GAMS GDX viewer

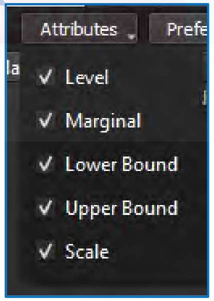


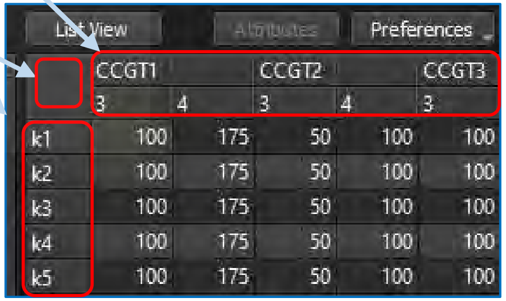

| a | b |
|---|---|
| 1 | 001 |
| 10 | 002 |
| 100 | 003 |
| 2 | 010 |
| 3 | 100 |




- In list view, you can order by different sets (be aware that all order is alphabetical and not numerical, therefore, to have proper numerical order, your sets need to be defined with zeros on the left)
- In the list view, you can set filters to display just some elements
- In table view, you can click and drag the rows and columns to change the display order
- Select the entire table (click in the corner)to copy and paste it into an Excel file
- When looking at variables, you can use the attributes option to display or hide the levels, limits, and marginal.



# Rule of thumb for selecting an LP optimization algorithm

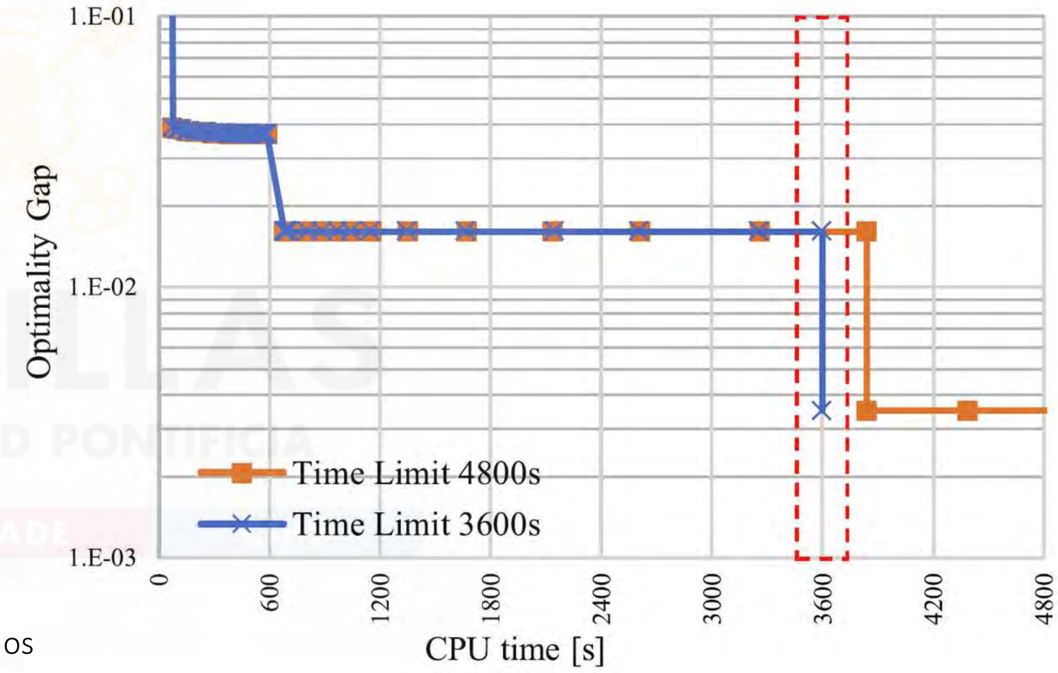
- Simplex (or dual simplex) method can be the best choice for moderate size (up to 100000 x 100000)
- Interior point method is usually the most efficient for huge and difficult problems
  - It is the most *numerically sensitive* algorithm. Numerical issues can cause crossover to stall
  - It can be *threaded* quite efficiently (compared to simplex)


- Difference in solution time can reach

  one order of magnitude

# Solution time for MIP problems

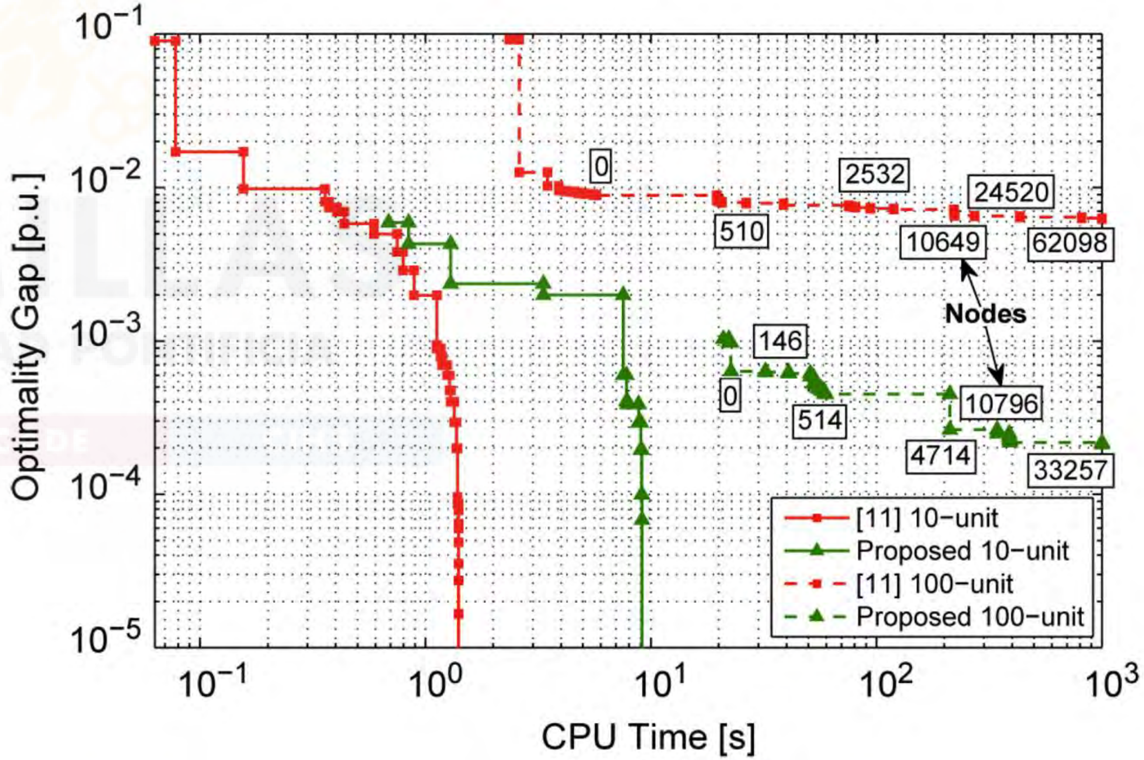
- Solution time depends on the CPU time limit




D.A. Tejada-Arango, S. Lumbreras, P. Sánchez-Martín, and A. Ramos *Which Unit-Commitment Formulation is Best? A Systematic Comparison* IEEE Transactions on Power Systems 35 (4): 2926-2936 Jul 2020 10.1109/TPWRS.2019.2962024

# Solution time for MIP problems

- Optimality gap is stepwise with the solution time
- Solution time strongly depends on the formulation quality




G. Morales-España, A. Ramos, and J. Garcia-Gonzalez *[An MIP Formulation for Joint Market-Clearing of Energy and Reserves Based on Ramp Scheduling](https://doi.org/10.1109/TPWRS.2013.2259601)* IEEE Transactions on Power Systems 29 (1): 476-488, Jan 2014 [10.1109/TPWRS.2013.2259601](https://doi.org/10.1109/TPWRS.2013.2259601)

# Algorithm improvements

- For solving LP/MILP, computer hardware got about 20 times faster, and the algorithms improved by a factor of about 9 for LP and around 50 for MILP, which gives a total speed-up of about 180 and 1,000 times, respectively

Th. Koch, T. Berthold, J. Pedersen, Ch. Vanaret "Progress in mathematical programming solvers from 2001 to 2020" *EURO Journal on Computational Optimization* 10 (2022) 100031 https://doi.org/10.1016/j.ejco.2022.100031

# Debugging an optimization model

- Grammar error
  - Read the error and click on the red line of the log file
- Infeasibility detection
  - Soft (elastic) constraints
    - Introduce a deficit or surplus variables in each equation and penalize it in the objective function. Be careful with the penalty parameter (FeasOpt in Gurobi/CPLEX)
  - Detect the smallest core of infeasible constraints by the LP solver (option *Irreducible Infeasible Subsets iis* in solvers)
    - Once known, they must be deleted or modified

# Options

| Options | **Description** |
|---|---|
| LimRow | Number of rows to show |
| LimCol | Number of columns to show |
| SolPrint | Solution output |
| SolveOpt | Replace |
| Decimals | Number of decimals in displaying values |
| IterLim | Maximum number of solver iterations |
| ResLim | Maximum solution time |
| Profile | Time profiling |
| ProfileTol | Profile threshold |
| Seed | Initialize seed for random numbers |

# $ Directives

| $ Directives | Description |
|---|---|
| $OnEmpty | Allow introduction of empty sets |
| $OnMulti | Allow redeclaration of sets |
| $OffListing | Suppress listing of the code |

# Variable attributes (varName.Attribute)

| Attribute | Description |
|---|---|
| lo | lower bound |
| up | upper bound |
| fx | fixes the variable to a constant |
| Range | range of the variable |
| l | initial value before and optimal value after |
| m | marginal value (reduced cost) |
| Scale | numerical scale factor |
| Prior | branching priority in a MIP model ($\infty \rightarrow$ not discrete) |
| SlackUp | slack from upper bound |
| SlackLo | slack from lower bound |
| Infeas | infeasibility out of bounds |

# Equation attributes (equationName.Attribute)

| Attribute | **Description** |
|---|---|
| lo | lower bound |
| up | upper bound |
| l | initial value before and optimal value after |
| m | marginal value (dual variable or shadow price) |
| Scale | numerical scaling factor |

# Model

```
model ModelName1 / Equation1 Equation3 Equation5 Equation7 /
model ModelName2 / all /
model ModelName3 / ModelName1 – Equation5 + Equation8 /
```

# Model attributes (modelName.Attribute)

| Attribute | Description | Attribute | **Description** |
|---|---|---|---|
| ResLim | Resource limit | IterUsd | Number of iterations |
| SolveOpt | Replace/merge/clear in consecutive solves | ResUsd | Resource used |
| SolSlack | Show slack variables | BRatio | Basis ratio controls the use of previous basis |
| SolvePrint | 0, 1, 2 (to remove the detailed solution from the .lst file) | HoldFixed | Fix and eliminate variables |
| TryLinear | Try linear model first | IterLim | Iteration limit |
| ModelStat | Model status | NodLim | Node limit |
| SolveStat | Solve status | OptCA | Absolute optimality tolerance |
| NumEqu | Number of equations | OptCR | Relative optimality tolerance |
| NumVar | Number of variables | OptFile | Use of an option file |
| NumDVar | Number of discrete variables | PriorOpt | Use of priority |
| NumNz | Number of non zeros | | |

# GAMS Call Options

| GAMS Options | Description |
|---|---|
| Suppress | Suppress echo of the code listing |
| PW | Page width |
| PS | Page size |
| RF | Shows all the symbols |
| Charset | Allows international characters |
| U1..U10 | User parameter |

For example, InterfaceName, SolverSelection, SkipExcelInput, SkipExcelOutput,

```
u1=“Excel_Interface_Name" u2=0 u3=0 u4=1 --NumberCores=4
```

# Boosting performance


- Threads
  - Use of multiple cores of a computer by the solver
- GUSS (Gather-Update-Solve-Scatter)
  - Use of sensitivity analysis for solving many similar problems
- Grid and Multi-Threading Solve Facility
  - Send many problems to solve and collect them after solved
- You can launch several GAMS processes simultaneously, being careful with conflicting filenames

# Scenario analysis of the transportation problem solved with GUSS

```
sets
   I  origins
   J  destinations
   SC scenarios

parameters
   pA    (i      ) origin capacity
   pB    (j      ) destination demand
   pC    (    i,j) per unit transportation cost
   pBS   (sc,   j) stochastic destination demand
   pX    (sc,i,j) stochastic units transported
   pCost (sc     ) stochastic transportation cost
   pPrice(sc,   j) stochastic spot price

variables
   vX(i,j) units transported
   vCost   transportation cost

positive variable vX

equations
   eCost          transportation cost
   eCapacity(i)   maximum capacity of each origin
   eDemand   (j)  demand supply at destination ;

eCost ..          sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) ..   sum[    j ,           vX(i,j)] =l= pA(i) ;
eDemand   (j) ..  sum[ i,               vX(i,j)] =g= pB(j) ;

model mTransport / all /
```

```
set scen_dem stochastic demand scenario dictionary /
   sc       . scenario . ''

*  update    the LHS with values of the RHS
   pB       . param    . pBS

*  store in the RHS with values of the LHS
   vX       . level    . pX
   vCost    . level    . pCost
   eDemand  . marginal . pPrice                     /
*************************************************************
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /
   SC scenarios   / sc000*sc999 /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pBS(sc,j) stochastic destination demand
      / sc000 . MADRID    400
        sc000 . BARCELONA 450
        sc000 . VALENCIA  150 / ;

* lazy input, feeding data for all the scenarios with random demand
pBS(sc,j) = pBS('sc000',j) * [1+uniform(-0.05,0.05)] ;

table pC(i,j) per unit transportation cost
           MADRID BARCELONA VALENCIA
VIGO        0.06     0.12     0.09
ALGECIRAS   0.05     0.15     0.11 ;
*************************************************************

* initialization of the destination demand
pB(j) = pBS('sc000',j) ;

solve mTransport using LP minimizing vCost scenario scen_dem
```

# Scenario analysis of the transportation problem solved with Grid computing and GUSS (i)

```
sets
   I  origins
   J  destinations
   SC scenarios

parameters
   pA    (i      ) origin capacity
   pB    (j      ) destination demand
   pC    (   i,j) per unit transportation cost
   pBS   (sc,  j) stochastic destination demand
   pX    (sc,i,j) stochastic units transported
   pCost (sc     ) stochastic transportation cost
   pPrice(sc,  j) stochastic spot price

variables
   vX(i,j) units transported
   vCost   transportation cost

positive variable vX

equations
   eCost          transportation cost
   eCapacity(i) maximum capacity of each origin
   eDemand   (j) demand supply at destination ;

eCost ..         sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) .. sum[   j ,           vX(i,j)] =l= pA(i) ;
eDemand   (j) .. sum[ i,              vX(i,j)] =g= pB(j) ;

model mTransport / all /
```

```
sets
   gs(sc)   scenarios per GUSS run
   sh       solution headers / System.GUSSModelAttributes /
   scen_dem stochastic demand scenario dictionary /
      sc          . scenario . ''
      scen_optn . opt      . st_report_o

*     update   the LHS with values of the RHS
      pB          . param    . pBS

*     store in the RHS with values of the LHS
      vX          . level    . pX
      vCost       . level    . pCost
      eDemand     . marginal . pPrice              /

***************************************************************
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /
   SC scenarios   / sc000*sc999 /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pBS(sc,j) stochastic destination demand
      / sc000 . MADRID    400
        sc000 . BARCELONA 450
        sc000 . VALENCIA  150 / ;

* lazy input, feeding data for all the scenarios with random demand
pBS(sc,j) = pBS('sc000',j) * [1+uniform(-0.05,0.05)] ;

table pC(i,j) per unit transportation cost
          MADRID  BARCELONA  VALENCIA
VIGO       0.06      0.12      0.09
ALGECIRAS  0.05      0.15      0.11 ;
```

# Scenario analysis of the transportation problem solved with Grid computing and GUSS (ii)

```
sets
*  using four cores and assignment of scenarios to cores
   core             grid jobs to run   / core001*core004        /
   coresc(core,sc) cores to scenario / core001.(sc000*sc249)
                                        core002.(sc250*sc499)
                                        core003.(sc500*sc749)
                                        core004.(sc750*sc999) /

parameter
   scen_optn          scenario options / OptFile 2, LogOption 1, SkipBaseCase 1,
                                         UpdateType 1, RestartType 1, NoMatchLimit 999 /
   st_report_o(sc,sh) status report
   pGridHandle(core)  grid handles ;
* initialization of the destination demand
pB(j) = pBS('sc000',j) ;

mTransport.SolveLink = %SolveLink.AsyncGrid% ;

* Sending Loop
loop (core,
   gs(sc) = coresc(core,sc)
   if (sum[gs(sc), 1] > 0,
      solve mTransport using LP minimizing vCost scenario scen_dem ;
      pGridHandle(core) = mTransport.Handle ;
   ) ;
) ;

* Recovering Loop
repeat
   loop (core $HandleCollect(pGridHandle(core)),
      display $HandleDelete (pGridHandle(core)) 'Trouble deleting handles' ;
      pGridHandle(core) = 0 ;
   ) ;
until card(pGridHandle) = 0 or TimeElapsed > 1000 ;
mTransport.SolveLink = %SolveLink.LoadLibrary% ;

display st_report_o
```

It could be a good idea to include a copy of the recovering loop within the sending loop to ensure that the amount of solves being executed is not larger than a certain number (for example, the number of cores):

```
loop(
    send
    repeat
        recover
    until handles < cores
)
repeat
    recover
until handles = 0
```

# Releasing memory


a) Define a dummy solve

b) Clear parameters

c) Run dummy model

```
option profile=10

set i / 1 * 10000000 /
parameter pp(i) ;

pp(i) = 33 ;

* dummy optimization problem used for releasing memory
variable vDummy
equation eDummy ; eDummy .. vDummy =e= 0 ;
model    mDummy / eDummy / ;

* only parameters that are no longer used can be cleared
option Clear=pp

* solve a dummy optimization problem to release memory usage
solve mDummy using LP minimizing vDummy
```

# GAMS to LaTeX

```
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pB(j) destination demand
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

table pC(i,j) per unit transportation cost
          MADRID BARCELONA VALENCIA
VIGO       0.06     0.12     0.09
ALGECIRAS  0.05     0.15     0.11

variables
   vX(i,j) units transported
   vCost   transportation cost

positive variable vX

equations
   eCost        transportation cost
   eCapacity(i) maximum capacity of each origin
   eDemand  (j) demand supply at destination ;

eCost        .. sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) .. sum[   j ,            vX(i,j)] =l= pA(i) ;
eDemand  (j) .. sum[ i   ,            vX(i,j)] =g= pB(j) ;

model mTransport / all /
solve mTransport using LP minimizing vCost
```

$$\min_x \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} \le a_i \quad \forall i$$

$$\sum_i x_{ij} \ge b_j \quad \forall j$$

$$x_{ij} \ge 0$$

Generate the doc file

GAMS transport.gms

**DocFile**=transport

Write the tex file

model2tex transport

## Symbols

### Sets

| Name | Domains | Description |
|---|---|---|
| I | * | origins |
| J | * | destinations |

### Parameters

| Name | Domains | Description |
|---|---|---|
| pA | I | origin capacity |
| pB | J | destination demand |
| pC | I, J | per unit transportation cost |

### Variables

| Name | Domains | Description |
|---|---|---|
| vX | I, J | units transported |
| vCost | | transportation cost |

### Equations

| Name | Domains | Description |
|---|---|---|
| eCost | | transportation cost |
| eCapacity | I | maximum capacity of each origin |
| eDemand | J | demand supply at destination |

## Equation Definitions

**eCost**

$$\sum_{I,J}(\mathrm{pC}_{I,J} \cdot \mathrm{vX}_{I,J}) = \mathrm{vCost}$$

**eCapacity**$_I$

$$\sum_{J}(\mathrm{vX}_{I,J}) \le \mathrm{pA}_I \qquad \forall I$$

**eDemand**$_J$

$$\sum_{I}(\mathrm{vX}_{I,J}) \ge \mathrm{pB}_J \qquad \forall J$$

$$\mathrm{vX}_{I,J} \ge 0 \ \forall I, J$$

3

1. Programming Style
2. GAMS Code
3. Embedded Python
4. Connect
5. Performance Issues
6. Advanced Algorithms

# Embedded Python

# GAMS Embedded Code Facility: Python

- From GAMS, it is possible to execute external code in Python.
  - Can be useful to perform actions that GAMS cannot (print figures) or that are more complex (complex data processing with functions and loops)
  - Example: Print figures during an iterative process to keep track of it
- GAMS Studio is not an excellent debugging option for Python code. The proposed example provides a generic structure that allows the Python code to be executed during the GAMS execution. It also allows its standalone execution from a more convenient tool like VSCode.
  - GAMS execution from GAMS Studio: no additional concerns are needed, just to run the GAMS code
  - Standalone execution from VSCode: select as python interpreter the one included in the GAMS installation located in *"GAMSdirectory"/GMSPython/python.exe*
  - The example is prepared for a particular directory structure:

```
🗁 pythonExample                  Project main directory
 ├🖹 mainGAMSfile.gms             GAMS source code
 └🗁 Python                        Specific directory for python code and data
    ├🖹 codigoPython.py           Python source code
    ├🖴 entradaPython.gdx         Python input data for standalone execution
    └🖴 salidaPython.gdx          Python output data
```

# Python embedded code: GAMS code

```
set
nameset /i1*i3/
;

parameters
*value of eps needed in python
s_eps /eps/
*parameters for the example
p_parameter (nameset)
p_parameterB(nameset)
p_parameterC(nameset)
;

p_parameter(nameset)=3;

*sum eps to everything that is going to be send to python so that
* all records are contained in the GDX
p_parameter(nameset) = p_parameter(nameset) + eps;

*GDX file with the input data to python.
*This is only needed when you plan to execute the python file as standalone,
* otherwise, all data are accessible from the GAMS memory
execute_unloaddi 'Python/entradaPython.gdx',
s_eps
nameset
p_parameter
;
```

Create a parameter with the "eps" value

Add "eps" to all data

GDx with all data used in python. **Only needed to run python as standalone**

```
*look for the path of the current file
$setnames "%GAMS.i%" filepath filename fileextension
*Inicia Python
embeddedCode Python:
import traceback
try:
    import pathlib
    import sys

    #insert the current path in the system path
    path=r'%filepath% '
    sys.path.insert(0,str(pathlib.Path(path.strip())))

    #import the function codigo_python and call it sending the GAMS memory,
    # and entornoGams=1 so the function knows it has been called from GAMS
    from Python.codigoPython import codigo_python
    codigo_python(GAMS=GAMS,entornoGams=1)

except Exception as e:
    traceback.print_exc()
    raise e
endembeddedCode p_parameterB

*subtract eps from all the data sent to python
* to restore their original values.
p_parameter(nameset) = p_parameter(nameset) - eps;

*unload python results from GDX
execute_loaddc  'Python/salidaPython.gdx', p_ parameterC;
```

Load python results

Restore data

# Python embedded code: Python code

```python
#%%
import pathlib
#path to this file
path_algoritmo = pathlib.Path(__file__).parent.absolute()
class ParProcessError(Exception):
    pass

#%% Python Main Function to be called from GAMS
def codigo_python(GAMS,entornoGams=1):
    if entornoGams == 1:
        printGams = lambda msn: GAMS.printLog(str(msn))
    elif entornoGams == 0:
        printGams = lambda msn: print(str(msn))
    else:
        raise Exception("no esta definido el entono de GAMS correctamnte")

    GAMS.epsAsZero=True
    #remove pyomo warnings from the log so that you do not see
    #  the precision loss warnings when writing the problem in text
    import logging
    logging.getLogger('pyomo.core').setLevel(logging.ERROR)

    #import libraries
    import GDXpds
    import pandas as pd
    #Load sets. Option depending on set dimensions and use in the code
    #nameset   = pd.DataFrame(     GAMS.get("nameset"))
    nameset   =               list(GAMS.get("nameset"))
    #nameset   =                set (GAMS.get("nameset"))
    #Load parameters/variables(variables have level upper, lower etc)
    p_parameter              = pd.Series    (dict(GAMS.get("p_parameter"     )))

    printGams('Data loaded')
```

Load data

```python
    #When used outside GAMS, eps becomes 5.0e+300 and must be set to 0.
    if entornoGams == 0:
        ss_eps = list(GAMS.get("s_eps"))
        s_eps=ss_eps[0]
        if s_eps>1:
            s_eps = 0.9*s_eps
            p_parameter [p_parameter > s_eps] = 0

    #Python may have decimal error when loading from GAMS, therefore,
    # for binary data we round to the unit and then convert the type to int
    p_parameter = p_parameter.round(0).astype('int')

    #Main Python code

    p_parameterB= pd.Series(p_parameter, p_parameter.index)
    GAMS.set("p_parameterB",list(p_parameterB.items()))

    p_parameterC= pd.DataFrame(p_parameterB, p_parameter.index)
    p_parameterC.columns= ['Value']
    p_parameterC.index.name = 'nameset'
    p_parameterC.reset_index(level=['nameset'], inplace=True)
    Datos = {'p_parameterC'   : p_parameterC,
            }
    GDX_file = str(path_algoritmo.joinpath('salidaPython.gdx'))
    GDX = GDXpds.to_GDX(Datos, GDX_file)

#%%When this file is executed directly without being imported by another (when
not called from GAMS)
if __name__ == "__main__":
    # Import a GDX simulating GAMS memory
    from GAMSemb import ECGAMSDatabase
    GAMS = ECGAMSDatabase(r' ', 'entradaPython.gdx')
    GAMS.arguments = '-a c -b -db abc'
    codigo_python(GAMS,entornoGams=0)
```

Data correction

Main code

Prepare output data

# Python/Pyomo

Pyomo is a Python library that allows defining optimization models using an algebraic language like the one used by GAMS.

Pros:
- Open source
- Active development by a vast community
- Processing of data and results can benefit from Python libraries, graphical functions, etc.

Cons:
- GAMS has been developed for a longer time. Some options that are easy to use in GAMS are not so trivial with Pyomo (or may not exist yet)
- Worse documentation

A simple but detailed (following good practices) example that can be used as a base is available in the:
https://gitlab001.iit.comillas.edu/pdeotaola/Ejemplo_Optimizacion_Python-Pyomo

4

# Connect (data input and output)

# Connect

GAMS Connect allows reading and writing data directly from/to:

- Excel
- CSV
- GDX

It uses yaml for a user-friendly programming code

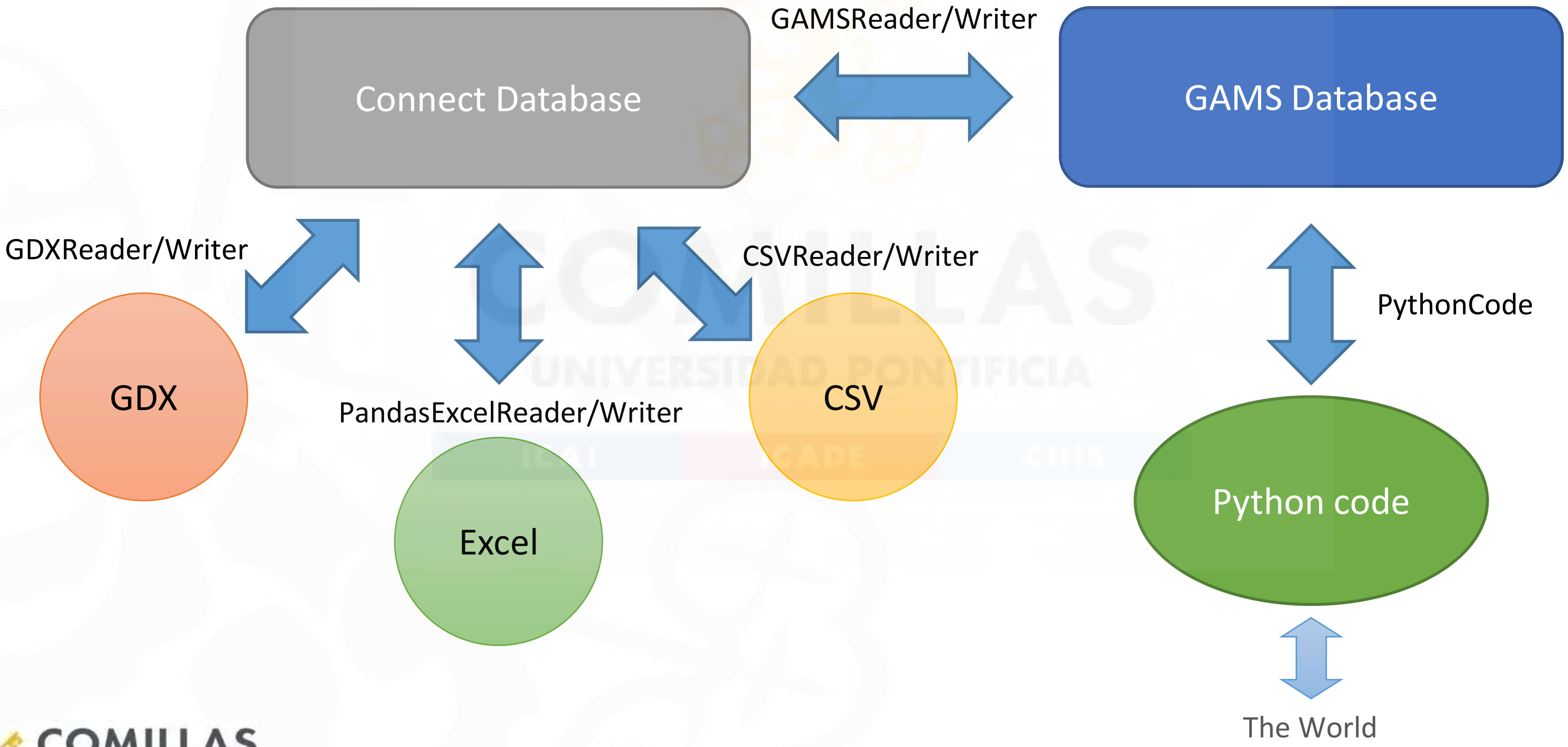
The code can be run with the **embedded code facility** or with a command line parameter:

- **ConnectIn='scriptfile'**: executes the instructions in 'file' at the beginning of the GAMS execution. As the sets and parameters are not defined, it can be used, for example, to build a single GDX from several input files that will be available for latter
- **ConnectOut ='scriptfile'**: executes the instructions in 'file' at the end of the GAMS execution. It can be used to store all the required data from the model

When using the embedded code facility is also possible to use **python code.**

https://www.GAMS.com/latest/docs/UG_GAMSCONNECT.html

# Connect



GAMSReader/Writer
Connect Database
GAMS Database
GDXReader/Writer
CSVReader/Writer
PythonCode
GDX
CSV
PandasExcelReader/Writer
Excel
Python code
The World

# Connect

| Connect agent | Description |
|---|---|
| CSVReader | Allows reading a symbol from a specified CSV file into the Connect database. |
| CSVWriter | Allows writing a symbol in the Connect database to a specified CSV file. |
| GAMSReader | Allows reading symbols from the GAMS database into the Connect database. |
| GAMSWriter | Allows writing symbols in the Connect database to the GAMS database. |
| GDXReader | Allows reading symbols from a specified GDX file into the Connect database. |
| GDXWriter | Allows writing symbols in the Connect database to a specified GDX file. |
| Options | Allows to set more general options that can affect the Connect database and other Connect agents. |
| PandasExcelReader | Allows reading symbols from a specified Excel file into the Connect database. |
| PandasExcelWriter | Allows writing symbols in the Connect database to a specified Excel file. |
| Projection | Allows index reordering and projection onto a reduced index space of a GAMS symbol. |
| PythonCode | Allows executing arbitrary Python code. |
| RawExcelReader | Allows reading unstructured data from a specified Excel file into the Connect database. |

```
*Out.yaml

- GAMSReader:
    symbols:
        - name: p_A

- PandasExcelWriter:
    file: myworkbook.xlsx
    symbols:
        - name: p_A
          range: out!A1
          rowDimension: 1
```

At the end of the GAMS execution

Read from GAMS to connect

Write from connect to excel

| In | A | B | C | D | F | G |
|---|---|---|---|---|---|---|
| 1 | i | j | k | | p_B | |
| 2 | | | | | j1 | j2 |
| 3 | i1 | j1 | k1 | | | |
| 4 | i2 | j2 | k2 | i1 | 1 | 4 |
| 5 | i3 | | k3 | i2 | 2 | 5 |
| 6 | | | k4 | i3 | 3 | 6 |

| Out | A | B | C | D | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | j1 | | | | j2 | | | |
| 2 | | k1 | k2 | k3 | k4 | k1 | k2 | k3 | k4 |
| 3 | | | | | | | | | |
| 4 | i1 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 5 | i2 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 6 | i3 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

**Data.xlsx**

```
*Command line parameters: connectOut=out.yaml
sets
i (*)
j (*)
k (*);
parameters
p_A(i,j,k)
p_B(i,j)
p_C(i,j);

$onEmbeddedCode Connect:
- PandasExcelReader:
    file: Data.xlsx
    symbols:
        - name: i
          type: set
          range: In!A2:A5
          rowDimension: 1
          columnDimension: 0
        - name: j
          type: set
          range: In!B2:B4
          rowDimension: 1
          columnDimension: 0
        - name: k
          type: set
          range: In!C2:C6
          rowDimension: 1
          columnDimension: 0
        - name: p_B
          range: In!D2:F6
          rowDimension: 1
          columnDimension: 1
- GAMSWriter:
    writeAll: True
- PythonCode:
    code: |
      import numpy as np
      import pandas as pd
      df = pd.Series(dict(GAMS.get('p_B')))
      df *=  2
      GAMS.set('p_C', list(df.items()))
$offEmbeddedCode
p_A(i,j,k)=10;
```

Read from excel to connect

Write from connect to GAMS

Python: read from GAMS, modify and write back to GAMS

4



# Performance Issues

# Finding and Fixing Execution Errors and Performance Problems

- Resolving Execution Errors
- Small to Large: Aid in Development and Debugging
- Increasing Efficiency: Reducing GAMS Execution Time
- Increasing Efficiency: Reducing Memory Use

https://www.gams.com/latest/docs/UG_ExecErrPerformance.html

GUROBI OPTIMIZATION

# Guidelines for Numerical Issues

I strongly recommend going through these web pages to improve the numerical properties of the optimization model.

- Avoid rounding of input
- Real numbers are not real
- Tolerances and user-scaling
  - Models at the edge of infeasibility
  - Gurobi tolerances and the limitations of double-precision arithmetic
  - Why scaling and geometry is relevant
  - Recommended ranges for variables and constraints
  - Improving ranges for variables and constraints
  - Advanced user scaling
  - Avoid hiding large coefficients
  - Dealing with big-M constraints

- Does my model have numerical issues?
- Solver parameters to manage numerical issues
  - Presolve
  - Choosing the right algorithm
  - Making the algorithm less sensitive

- Instability and the geometry of optimization problems
  - The case of linear systems:
  - The geometry of linear optimization problems
  - Multiple optimal solutions
  - Dealing with epsilon-optimal solutions
  - Thin feasible regions
  - Optimizing over the circle:
  - Optimizing over thin regions:
  - Stability and convergence

https://www.gurobi.com/documentation/current/refman/guidelines_for_numerical_i.html

# LP Performance issues and their suggested resolution

| LP performance issue | Suggested resolution |
|---|---|
| Numerical instability | • Calculate and input model data in double precision<br>• Eliminate nearly-redundant rows and/or columns of *A a priori*<br>• Avoid mixtures of large and small numbers:<br>(i) Be suspicious of $\kappa$ between $10^{10}$ and $10^{14}$;<br>(ii) Avoid data leading to $\kappa$ greater than $10^{14}$<br>• Use alternate scaling (in the model formulation or optimizer settings)<br>• Increase the Markowitz threshold<br>• Employ the numerical emphasis parameter (if available) |
| Lack of objective function improvement under degeneracy | • Try all other algorithms (and variants)<br>• Perturb data either *a priori* or using algorithmic settings |
| Primal degeneracy | • Use either dual simplex or interior point on primal problem |
| Dual degeneracy | • Employ either primal simplex or interior point on primal problem |
| Both primal and dual degeneracy | • Execute interior point on primal or dual problem |
| Excessive time per iteration | • Try all other algorithms (and variants)<br>• Use algorithmic settings to conserve memory or purchase more externally<br>• Try less expensive pricing settings if using simplex algorithms |
| Excessive simplex algorithm iterations | • Try Steepest edge or Devex variable selection |
| Multiple bound shift removals or significant infeasibilities after removing shifts | • Reduce feasibility and optimality tolerances |
| Barrier algorithm iterations with little or no progress | • Increase barrier convergence tolerance in order to initiate crossover earlier |
| Too much time in crossover | • Reduce barrier convergence tolerance in order to provide a better starting point for crossover |

E. Klotz, A.M. Newman *Practical guidelines for solving difficult linear programs* Surveys in Operations Research and Management Science 18 (1-2), 1-17, Oct 2013 10.1016/j.sorms.2012.12.001

# Preprocessing by Gurobi

```
--- StarNetLite_TEPM_Iceland.gms(15000) 986 Mb
---   1,167,736 rows  2,004,441 columns  8,140,314 non-zeroes
---   0 nl-code  0 nl-non-zeroes
---   7 discrete-columns
***   63,652 relaxed-columns WARNING
--- StarNetLite_TEPM_Iceland.gms(15000) 984 Mb
--- Executing GUROBI: elapsed 0:00:20.917
--- StarNetLite_TEPM_Iceland.gms(15000) 984 Mb  3 secs

Gurobi          24.6.1 r55820 Released Jan 18, 2016 WEI x86 64bit/MS
Windows

Gurobi link license.
Gurobi library version 6.5.0
Reading parameter(s) from "C:\Users\aramos\Desktop\Aramos\TEPES\gurobi.opt"
>>  Method    2
>>  IntFeasTol 1e-9
>>  OptimalityTol 1e-9
>>  FeasibilityTol 1e-9
>>  IIS    1
>>  RINS      100
>>  DisplayInterval 1
>>  NumericFocus       1
>>  Kappa    1
>>  MarkowitzTol 0.999
>>  UseBasis 0
>>  CrossOver 0
>>  Names  0
>>  AggFill 0
>>  GomoryPasses 0
>>  Heuristics 0.001
>>  MipFocus 3
>>  *PreDual 0
>>  *PrePasses 3
>>  *PreSolve 2
>>  *PreSparsify 1
```

40 % reduction in rows,
38 % in columns and
30 % in nonzeros

```
Finished reading from "C:\Users\aramos\Desktop\Aramos\TEPES\gurobi.opt"
Starting Gurobi...
Optimize a model with 1167735 rows, 2004440 columns and 8140308 nonzeros
Coefficient statistics:
  Matrix range    [7e-09, 1e+03]
  Objective range [1e+00, 1e+00]
  Bounds range    [4e-07, 5e+00]
  RHS range       [1e-18, 1e+01]
Presolve removed 257457 rows and 497937 columns (presolve time = 2s) ...
Presolve removed 259520 rows and 500000 columns (presolve time = 2s) ...
Presolve removed 470416 rows and 710896 columns (presolve time = 3s) ...
Presolve removed 470616 rows and 711132 columns (presolve time = 4s) ...
Presolve removed 470666 rows and 753485 columns (presolve time = 5s) ...
Presolve removed 470705 rows and 753535 columns (presolve time = 6s) ...
Presolve removed 470705 rows and 753535 columns
Presolve time: 6.23s
Presolved: 697030 rows, 1250905 columns, 5645696 nonzeros
```

# Preprocessing by Gurobi

```
Case SEP2030
Read LP format model from file openTEPES_SEP2030sto.lp
Reading time = 106.19 seconds
eTotalTCost: 2999548 rows, 3513436 columns, 11508142 nonzeros
Statistics for model eTotalTCost :
  Linear constraint matrix    : 2999548 Constrs, 3513436 Vars, 11508142 NZs
  Variable types              : 2858236 Continuous, 655200 Integer (655200 Binary)
  Matrix coefficient range    : [ 0.00092, 5000 ]
  Objective coefficient range : [ 1, 1 ]
  Variable bound range        : [ 0.00120252, 4307.64 ]
  RHS coefficient range       : [ 0.00120223, 2765.6 ]
Presolve removed 445826 rows and 519558 columns (presolve time = 6s) ...
Presolve removed 728231 rows and 831354 columns (presolve time = 10s) ...
Presolve removed 927371 rows and 840000 columns (presolve time = 15s) ...
Presolve removed 932122 rows and 845683 columns (presolve time = 20s) ...
Presolve removed 941267 rows and 858479 columns (presolve time = 25s) ...
Presolve removed 952315 rows and 872117 columns (presolve time = 30s) ...
Presolve removed 969568 rows and 892522 columns (presolve time = 35s) ...
Presolve removed 974714 rows and 902410 columns (presolve time = 44s) ...
Presolve removed 974731 rows and 902410 columns (presolve time = 53s) ...
Presolve removed 974731 rows and 902410 columns
Presolve time: 53.20s
Statistics for model eTotalTCost_pre :
  Linear constraint matrix    : 2024817 Constrs, 2611026 Vars, 8347459 NZs
  Variable types              : 2392626 Continuous, 218400 Integer (218400 Binary)
  Matrix coefficient range    : [ 0.0319163, 5000 ]
  Objective coefficient range : [ 0.00092, 2 ]
  Variable bound range        : [ 0.0002, 4307.64 ]
  RHS coefficient range       : [ 0.000141, 3281.46 ]
```

```
Case ES2030
eTotalTCost: 5162243 rows, 6832942 columns, 21554828 nonzeros
Statistics for model eTotalTCost :
  Linear constraint matrix    : 5162243 Constrs, 6832942 Vars, 21554828 NZs
  Matrix coefficient range    : [ 0.000107523, 3554.92 ]
  Objective coefficient range : [ 1, 1 ]
  Variable bound range        : [ 6.08628e-06, 4307.64 ]
  RHS coefficient range       : [ 0.00483795, 2844.48 ]
Presolve removed 547789 rows and 739037 columns (presolve time = 8s) ...
Presolve removed 1386313 rows and 1577561 columns (presolve time = 10s) ...
Presolve removed 1387761 rows and 1579009 columns (presolve time = 16s) ...
Presolve removed 1389569 rows and 1611781 columns (presolve time = 20s) ...
Presolve removed 1389569 rows and 1614605 columns
Statistics for model eTotalTCost_pre :
  Linear constraint matrix    : 3772674 Constrs, 5218337 Vars, 15088689 NZs
  Matrix coefficient range    : [ 0.0002813, 2488.05 ]
  Objective coefficient range : [ 0.000107523, 163.402 ]
  Variable bound range        : [ 6.08628e-06, 4307.64 ]
  RHS coefficient range       : [ 0.00258675, 2844.48 ]
```

# Preprocessing by Gurobi Python shell

- Before and after presolve can help you in detecting improvements in the formulation
- Allows getting the optimization problem after the presolve

```
ModelName          = read("OriginalProblem.lp")
ModelNamePresolved = ModelName.presolve()
ModelNamePresolved.write("PresolvedProblem.lp")
```

- Gurobi Model Analyzer (`gurobi_modelanalyzer`) allows to detect numerical problems

# Some tips for MIP

- Think about lazy constraints (only in GAMS/CPLEX/Gurobi)
- Avoid introducing symmetry (totally equal decision variables). GAMS/CPLEX has a symmetry-breaking cut parameter
  - Symmetry-breaking constraint $\boldsymbol{x_i \geq x_{i+1}}$
- Avoid the use of big $M$ parameters or put tight (lowest upper bound) values for the big $M$
- GAMS/CPLEX/Gurobi supports the use of an indicator constraint $x \leq My$

$$\begin{array}{c} \min Fy + Vx \\ x \leq My \\ x \geq 0 \\ y \in \{0,1\} \end{array} \longrightarrow \begin{array}{c} \min Fy + Vx \\ x \leq 0 \\ x \geq 0 \\ y \in \{0,1\} \end{array}$$

Write in the file cplex.opt

```
indic constraint$y 0
```

# Reformulation in MIP problems

STRONGER

- Most MIP problems can be formulated in different ways
- In MIP problems, a good formulation is crucial to solve the model
- How good is a MIP formulation?
  - Integrality gap: the difference between the objective function of the MIP and LP relaxation solutions
- Given two equivalent MIP formulations, one is *stronger* (*tighter/better*) than the other if the feasible region of the linear relaxation is strictly contained in the feasible region of the other. The integrality gap is lower.

E. Klotz, A.M. Newman *Practical guidelines for solving difficult mixed integer linear programs* Surveys in Operations Research and Management Science 18 (1-2), 18-32, Oct 2013 10.1016/j.sorms.2012.12.001

# Warehouse location problem (no limits) (i)

- Choose where to locate warehouses among a set of locations and assign clients to the warehouses, minimizing the total cost. No limits mean that there is no limit on the number of clients assigned to a warehouse.
- Data

$j$ locations

$i$ clients

$c_j$ localization cost in $j$

$h_{ij}$ cost of satisfying the demand of client $i$ from $j$

- Variables

$$y_j = \begin{cases} 1 & \text{warehouse located in } j \\ 0 & \text{otherwise} \end{cases}$$

$x_{ij}$ fraction of demand of client $i$ met from $j$

# Warehouse location problem (no limits) (ii)

Formulation #1

$$\min \sum_j c_j y_j + \sum_{ij} h_{ij} x_{ij}$$
$$\sum_j x_{ij} = 1 \quad \forall i$$
$$x_{ij} \le y_j \quad \forall ij$$
$$y_j \in \{0,1\}, x_{ij} \in [0,1]$$

Number of constraints: $I + IJ$

Formulation #2

$$\min \sum_j c_j y_j + \sum_{ij} h_{ij} x_{ij}$$
$$\sum_j x_{ij} = 1 \quad \forall i$$
$$\sum_i x_{ij} \le M y_j \quad \forall j$$
$$y_j \in \{0,1\}, x_{ij} \in [0,1]$$

Number of constraints: $I + J$

- Both formulations are MIP equivalent. However, formulation #1 is stronger
- Intuitively the fewer constraints the better. That's true in LP. However, in many MIP problems, the more constraints, the better.

# Production problem with fixed and inventory costs (i)

- Data

$t$ time period

$c_t$ fixed cost, $p_t$ variable cost, $h_t$ inventory cost

$d_t$ demand

- Variables

$$y_t = \begin{cases} 1 \text{ to produce} \\ 0 \text{ not produce} \end{cases}$$

$x_t$ amount produced

$s_t$ inventory at the end of the period

- Formulation #1

$$\min \sum_t \left( c_t y_t + p_t x_t + h_t s_t \right)$$

$$s_{t-1} + x_t = d_t + s_t \quad \forall t$$

$$x_t \le M y_t \quad \forall t$$

$$s_0 = s_T = 0$$

$$x_t, s_t \ge 0, y_t \in \{0,1\}$$

Number of constraints: $2T$

Number of variables: $3T$

# Production problem with fixed and inventory costs (ii)

- Variables

$$y_t = \begin{cases} 1 \text{ to produce} \\ 0 \text{ not produce} \end{cases}$$

$q_{it}$ quantity produced in period $i$ to meet the demand in period $t \geq i$

- Formulation #2

$$\min \sum_{t=1}^{T} \sum_{i=1}^{t} \left( p_i + h_i + h_{i+1} + \cdots + h_{t-1} \right) q_{it} + \sum_{t=1}^{T} c_t y_t$$

$$\sum_{i=1}^{t} q_{it} = d_t \quad \forall t$$

$$q_{it} \leq d_t y_i \quad \forall it$$

$$q_{it} \geq 0, y_t \in \{0,1\}$$

Number of constraints: $T + T^2/2$

Number of variables: $T + T^2/2$

- Formulation #2 is better. However, it has a greater number of constraints and variables.

# Tight and compact unit commitment


- D.A. Tejada-Arango, S. Lumbreras, P. Sánchez-Martín, and A. Ramos *Which Unit-Commitment Formulation is Best? A Systematic Comparison* IEEE Transactions on Power Systems 35 (4): 2926-2936 Jul 2020 10.1109/TPWRS.2019.2962024
- G. Gentile, G. Morales-España and A. Ramos *A Tight MIP Formulation of the Unit Commitment Problem with Start-up and Shut-down Constraints* EURO Journal on Computational Optimization 5 (1), 177–201 March 2017 10.1007/s13675-016-0066-y
- G. Morales-España, C.M. Correa-Posada, A. Ramos *Tight and Compact MIP Formulation of Configuration-Based Combined-Cycle Units* IEEE Transactions on Power Systems 31 (2), 1350-1359, March 2016 10.1109/TPWRS.2015.2425833
- G. Morales-España, J.M. Latorre, and A. Ramos *Tight and Compact MILP Formulation for the Thermal Unit Commitment Problem* IEEE Transactions on Power Systems 28 (4): 4897–4908, Nov 2013 10.1109/TPWRS.2012.2222938
- G. Morales-España, J.M. Latorre, and A. Ramos *Tight and Compact MILP Formulation of Start-Up and Shut-Down Ramping in Unit Commitment* IEEE Transactions on Power Systems 28 (2): 1288-1296, May 2013 10.1109/TPWRS.2012.2222938

# Ramp constraints

$$P_{nt} - P_{n-1,t} \le rup_t$$
$$P_{n-1,t} - P_{nt} \le rdw_t$$

Classical

$$P_{nt} - P_{n-1,t} \le rup_t \, UC_{nt}$$
$$P_{n-1,t} - P_{nt} \le rdw_t (UC_{nt} + SD_{nt})$$

Tighter

- Ramp equations are considered in the periods only when the unit is connected

$P_{nt}$ : output above the minimum load
$rup_t$: upwards ramp limit for generator $t$
$rdw_t$: downwards ramp limit for generator $t$
$UC_{nt}$: 1 if generator $t$ is connected in hour $n$, 0 otherwise
$SU_{nt}$: 1 if generator $t$ is started in hour $n$
$SD_{nt}$: 1 if generator $t$ is shutdown in hour $n$

# Why the constraint is tighter?

Given that the constraint uses the output above the minimum load, it can only be applied when the unit is committed until the following period the unit was committed.

| | $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $UC_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| | $SU_n$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | $SD_n$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| RampUp | $UC_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| RampDw | $UC_n + SD_n$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

comillas.edu

COMILLAS UNIVERSIDAD PONTIFICIA ICAI ICADE CIHS

# Reformulation of an NLP problem

$$\min \sum_{i=1}^{n} \sum_{j=i+1}^{n} q_{ij} x_i x_j$$
$$\sum_{j=1}^{n} x_j = 1$$
$$\sum_{j=1}^{n} r_j x_j = r_0$$

$$\min \sum_{i=1}^{n} x_i \sum_{j=i+1}^{n} q_{ij} x_j$$
$$\sum_{j=1}^{n} x_j = 1$$
$$\sum_{j=1}^{n} r_j x_j = r_0$$

$$\min \sum_{i=1}^{n} x_i w_i$$
$$w_i = \sum_{j=i+1}^{n} q_{ij} x_j$$
$$\sum_{j=1}^{n} x_j = 1$$
$$\sum_{j=1}^{n} r_j x_j = r_0$$

- Formulation #2 is better than the #1. The evaluation of the objective function in #1 requires $2n^2/2$ multiplications. In #2 only $n + n^2/2$
- Formulation #3 has essentially the same number of multiplications, but they appear in linear constraints. The number of constraints is bigger, but all of them are linear. Linear algebra is much more efficient. Formulation #3 is the most efficient

# Reformulation of an NLP problem

$$\min \frac{x+y}{\sum_i z_i}$$

$$\min \frac{u}{v}$$
$$u = x + y$$
$$v = \sum_i z_i$$
$$v \geq \varepsilon$$

- Formulation #1 has a lot of nonlinear variables, and it is not protected against division by zero
- Formulation #2 has only 2 nonlinear variables; the remaining ones appear in linear equations, and the denominator is lower bounded to avoid division by zero. The model is easier to solve and more robust

# Product of two variables $x_1 x_2$

$$x_1 x_2 = y_1^2 - y_2^2$$

Quadratic separable form

$$y_1 = (x_1 + x_2)/2$$
$$y_2 = (x_1 - x_2)/2$$
$$l_1 \leq x_1 \leq u_1$$
$$l_2 \leq x_2 \leq u_2$$
$$\frac{1}{2}(l_1 + l_2) \leq y_1 \leq \frac{1}{2}(u_1 + u_2)$$
$$\frac{1}{2}(l_1 - u_2) \leq y_2 \leq \frac{1}{2}(u_1 - l_2)$$

Bradley, Hax, and Magnanti *Applied Mathematical Programming* Addison-Wesley, 1977

# Solving large-scale problems

- MIP
  - Solve with a sensible relative optimality tolerance
  - Provide an initial solution based on specific knowledge of the model or use the solution from a previous solve
- NLP
  - Introduce sensible bounds on variables AND
  - Provide a good enough starting point AND
  - Scale the problem

# MIP models. Gurobi parameters

- Most important parameters
  - Threads, MIPFocus
- Solution Improvement
  - ImproveStartTime, ImproveStartGap
- Termination
  - TimeLimit
  - MIPGap, MIPGapAbs
  - NodeLimit, IterationLimit, SolutionLimit
  - Cutoff
- Speeding Up The Root Relaxation
  - Method

- ❑ Numerical issues
  - ✓ Presolve, PrePasses, Aggregate, AggFill, PreSparsify, PreDual, PreDepRow
  - ✓ NumericFocus
- ❑ Heuristics
  - ✓ Heuristics, SubMIPNodes, MinRelNodes, PumpPasses, ZeroObjNodes
  - ✓ RINS 100
- ❑ Cutting Planes
  - ✓ Cuts, GomoryPasses, FlowCoverCuts, MIRCuts

https://www.gurobi.com/documentation/current/refman/mip_models.html

# CPLEX Performance Tuning for MIP

- Names no
- NodeFileInd 3
- NodeSel 0
- VarSel 3
- StartAlg 4
- MemoryEmphasis 1
- WorkMem 1000
- MIPEmphasis 2
- MIPSearch 2
- SolveFinal 0
- Solution Polishing
- Solution pool
- FlowCovers
- FeasOptMode 2
- FeasOpt 1
- tuning cplex.opt
- RINSHeur 100
- FpHeur 2

*Pure branch and bound*

- Cuts -1
- HeurFreq -1

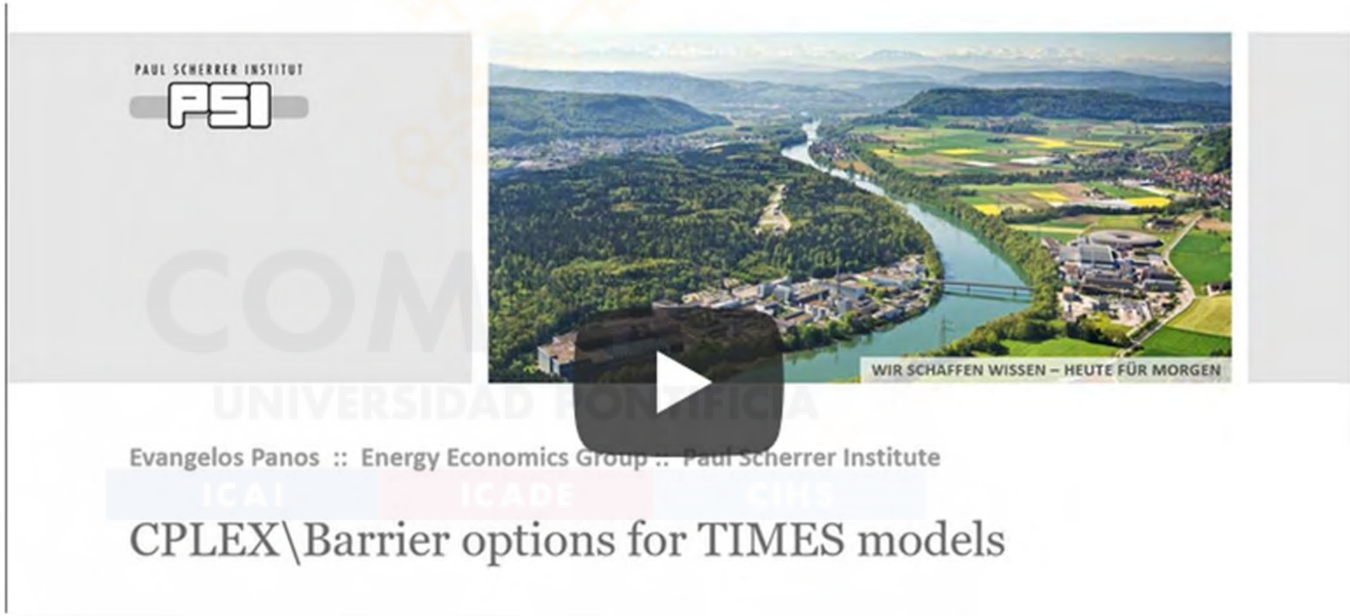
*Presolve*

- PreInd, PrePass

- Solution method of LP problem
  - ✓ First iteration (interior point or simplex method)
  - ✓ Successive iterations (primal or dual simplex)
- Priority for variable selection
  - ✓ Select variables that impact the most in the o.f. (e.g., investment vs. operation variables)
- Initial cutoff or incumbent
  - ✓ Initial valid bound of the o.f. estimated by the user

https://www.ibm.com/support/pages/cplex-performance-tuning-mixed-integer-programs

# How to Tune CPLEX Options for TIMES models




https://iea-etsap.org/webinar/CPLEX%20options%20for%20running%20TIMES%20models.pdf

5



# Advanced Algorithms

# Fixed-Charge Transportation Problem (FCTP)

***Flows***
(second stage)

***Investment decisions***
(first stage)

Capacity of each origin

Demand of each destination

Flow can pass only for installed connections

Complete problem

$$\min_{x_{ij},y_{ij}} \sum_{ij} (f_{ij}y_{ij} + c_{ij}x_{ij})$$
$$\sum_j x_{ij} \le a_i \quad \forall i$$
$$\sum_i x_{ij} \ge b_j \quad \forall j$$
$$x_{ij} \le M_{ij}y_{ij} \quad \forall ij$$
$$x_{ij} \ge 0, y_{ij} \in \{0,1\}$$

- Bd Relaxed Master

$$\min_{y_{ij},\theta} \sum_{ij} (f_{ij}y_{ij}) + \theta$$
$$\delta^l\theta - \theta^l \ge \sum_{ij} \pi_{ij}^l M_{ij}(y_{ij}^l - y_{ij}) \quad l = 1, \dots, k$$
$$y_{ij} \in \{0,1\}$$

O.F. of the subproblem at iteration $l$

Dual variables of linking constraints at iteration $l$

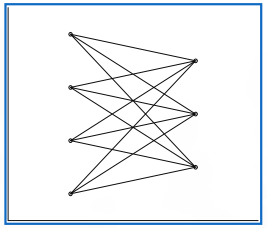
Master proposal at iteration $l$

- Bd Subproblem

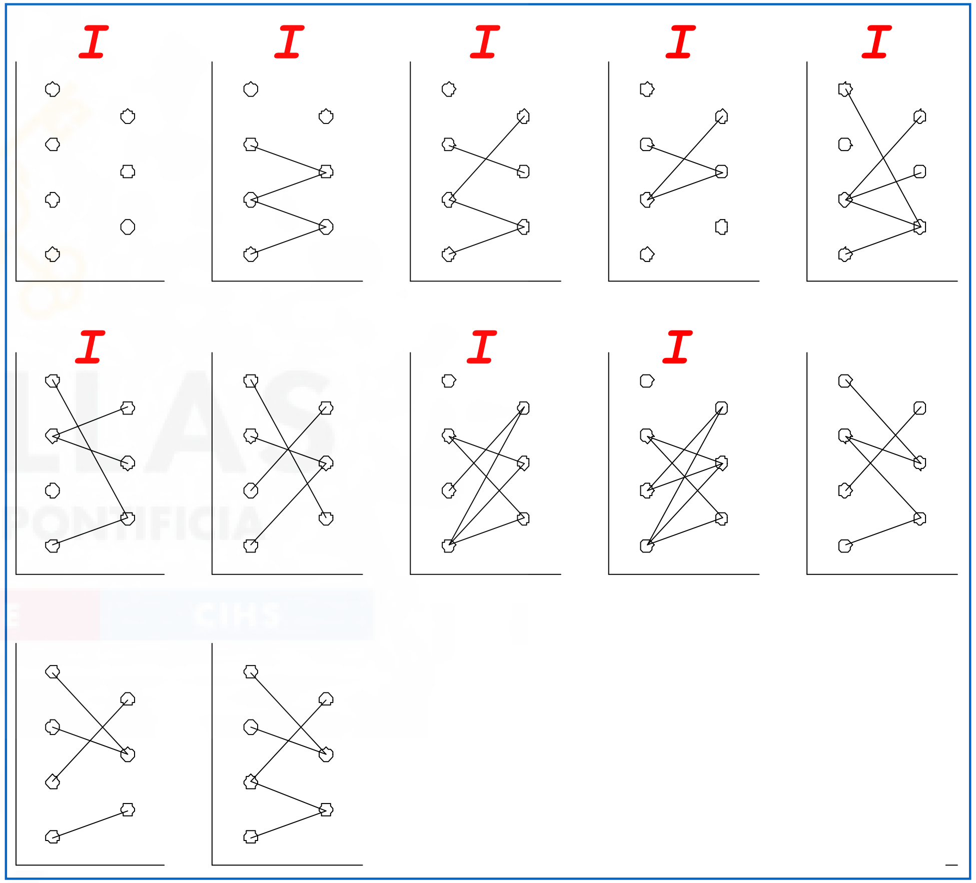
$$\theta^k = \min_{x_{ij}} \sum_{ij} (c_{ij}x_{ij})$$
$$\sum_j x_{ij} \le a_i \quad \forall i$$
$$\sum_i x_{ij} \ge b_j \quad \forall j$$
$$x_{ij} \le M_{ij}y_{ij}^k \quad \forall ij \quad : \pi_{ij}^k$$
$$x_{ij} \ge 0$$

# Fixed-Charge Transportation Problem. Bd Solution

- Possible arcs

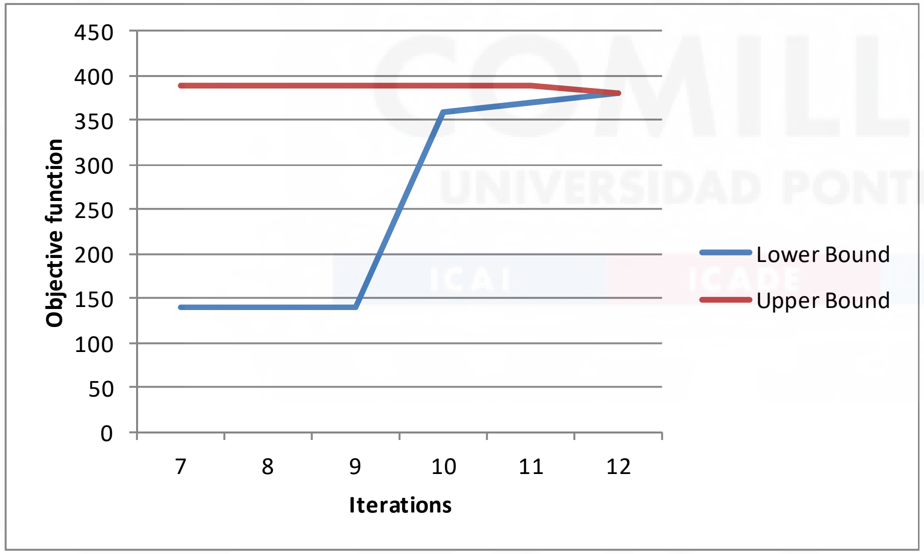


- Solutions along Benders decomposition iterations



# Fixed-Charge Transportation Problem. Bd Convergence

| *Iteration* | *Lower Bound* | *Upper Bound* |
|---|---|---|
| *1 a 6* | $-\infty$ | $\infty$ |
| *7* | *140* | *390* |
| *8* | *140* | *390* |
| *9* | *140* | *390* |
| *10* | *360* | *390* |
| *11* | *370* | *390* |
| *12* | *380* | *380* |



# FCTP solved by Benders decomposition (i)

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/BendersDecomposition_FCTP.gms

```
$Title Fixed-charge transportation problem (FCTP) solved by Benders decomposition

* relative optimality tolerance in solving MIP problems
option OptcR = 0

sets
   L         iterations        / l1 * l20 /
   LL(l)     iterations subset
   I         origins           / i1 * i4  /
   J         destinations      / j1 * j3  /

* Begin problem data

parameters
   A(i)      product offer
             / i1 10, i2 30, i3 40, i4 20 /
   B(j)      product demand
             / j1 20, j2 50, j3 30 /

table C(i,j) per unit variable transportation cost
       j1  j2  j3
   i1   1   2   3
   i2   3   2   1
   i3   2   3   4
   i4   4   3   2

table F(i,j) fixed transportation cost
       j1  j2  j3
   i1  10  20  30
   i2  20  30  40
   i3  30  40  50
   i4  40  50  60

* End problem data

abort $(sum[i, A(i)] < sum[j, B(j)]) 'Infeasible problem'

parameters
   BdTol        relative Benders tolerance / 1e-6 /
   Z_Lower      lower bound        / -inf /
   Z_Upper      upper bound        /  inf /
   Y_L  (l,i,j) first stage variables values               in iteration l
   PI_L (l,i,j) dual variables of second stage constraints in iteration l
   Delta(l)     cut type (feasibility 0 optimality 1)      in iteration l
   Z2_L (l)     subproblem objective function value        in iteration l

positive variable
   X(i,j)       arc flow

binary   variable
   Y(i,j)       arc investment decision

variables
   Z1           first  stage objective function
   Z2           second stage objective function
   Theta        recourse function
```

# FCTP solved by Benders decomposition (ii)

```
equations
   EQ_Z1           first  stage     objective function
   EQ_Z2           second stage     objective function
   EQ_OBJ          complete problem objective function
   Offer    (i  ) offer  at origin
   Demand   (  j) demand at destination
   FlowLimit(i,j) arc flow limit
   Bd_Cuts    (ll) Benders cuts ;

EQ_Z1          .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + Theta ;

EQ_Z2          .. Z2 =e=                             sum[(i,j), C(i,j)*X(i,j)] ;

EQ_OBJ         .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(i,j), C(i,j)*X(i,j)] ;

Offer    (i  ) .. sum[j, X(i,j)] =l= A(i) ;

Demand   (  j) .. sum[i, X(i,j)] =g= B(j) ;

FlowLimit(i,j) .. X(i,j) =l= min[A(i),B(j)] * Y(i,j) ;

Bd_Cuts(ll)    .. Delta(ll) * Theta =g= Z2_L(ll) -
                  sum[(i,j), PI_L(ll,i,j) * min[A(i),B(j)] * (Y_L(ll,i,j) - Y(i,j))] ;

model Master_Bd     / EQ_Z1  Bd_Cuts                /
model Subproblem_Bd / EQ_Z2  Offer Demand FlowLimit /
model Complete      / EQ_OBJ Offer Demand FlowLimit / ;

X.up(i,j) = min[A(i),B(j)]

* to allow CPLEX correctly detect rays in an infeasible problem
* only simplex method can be used and no preprocessing neither scaling options
* optimality and feasibility tolerances are very small to avoid primal degeneration

file COPT / cplex.opt /
put  COPT putclose 'ScaInd -1' / 'LPMethod 1' / 'PreInd 0' / 'EpOpt 1e-9' / 'EpRHS 1e-9' / ;

Subproblem_Bd.OptFile = 1 ;
```

# FCTP solved by Benders decomposition (iii)

```
* parameter initialization

LL      (l) = no ;
Delta   (l) =  0 ;
Z2_L    (l) =  0 ;
PI_L(l,i,j) =  0 ;
Y_L (l,i,j) =  0 ;

* Benders algorithm iterations
Theta.fx    =  0 ;
loop (l $(abs(1-Z_Lower/Z_Upper) > BdTol),

*  solving master problem
   solve Master_Bd using MIP minimizing Z1 ;

*  storing the master solution
   Y_L(l,i,j) = Y.l(i,j) ;

*  fixing first-stage variables and solving subproblem
   Y.fx( i,j) = Y.l(i,j) ;

*  solving subproblem
   solve Subproblem_Bd using RMIP minimizing Z2 ;

*  storing parameters to build a new Benders cut
   if (Subproblem_Bd.ModelStat = 4,
      Delta(l) = 0 ;
      Z2_L (l) = Subproblem_Bd.SumInfes ;
   else
*     updating lower and upper bound
      Z_Lower =              Z1.l ;
      Z_Upper = min(Z_Upper, Z1.l - Theta.l + Z2.l) ;

      Theta.lo = -inf ;
      Theta.up =  inf ;

      Delta(l) =    1 ;
      Z2_L (l) = Subproblem_Bd.ObjVal ;
   ) ;

   PI_L(l,i,j) = FlowLimit.m(i,j) ;

   Y.lo(  i,j) = 0 ;
   Y.up(  i,j) = 1 ;

*  increase the set of Benders cuts
   LL(l) = yes ;
) ;

solve Complete using MIP minimizing Z1
```

comillas.edu

# Stochastic FCTP

## Complete problem

***Flows***
(second stage)

***Investment decisions***
(first stage)

Capacity of each origin

Demand of each destination

Flow can go only for installed connections

$$\min_{x_{ij}^{\omega}, y_{ij}} \sum_{ij} \left( f_{ij} y_{ij} + \sum_{\omega} p_{\omega} c_{ij} x_{ij}^{\omega} \right)$$

$$\sum_{j} x_{ij}^{\omega} \le a_i \quad \forall i\omega$$

$$\sum_{i} x_{ij}^{\omega} \ge b_j^{\omega} \quad \forall j\omega$$

$$x_{ij}^{\omega} \le M_{ij}^{\omega} y_{ij} \quad \forall ij\omega$$

$$x_{ij}^{\omega} \ge 0, y_{ij} \in \{0,1\}$$

# Deterministic & Stochastic FCTP

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/FCTP_Sto.gms

```
$Title Deterministic fixed-charge transportation problem (DFCTP)

* relative optimality tolerance in solving MIP problems
option OptcR = 0

sets
   I         origins           / i1 * i4 /
   J         destinations      / j1 * j3 /


parameters
   A(i)      product offer   / i1 20, i2 30, i3 40, i4 20 /
   B(j)      product demand / j1 20, j2 50, j3 30 /


table C(i,j) per unit variable transportation cost
       j1  j2  j3
   i1   1   2   3
   i2   3   2   1
   i3   2   3   4
   i4   4   3   2

table F(i,j) fixed transportation cost
       j1  j2  j3
   i1  10  20  30
   i2  20  30  40
   i3  30  40  50
   i4  40  50  60

abort $(sum[i, A(i)] < sum[j, B(j)]) 'Infeasible problem'

positive variable
   X(i,j)       arc flow
binary   variable
   Y(i,j)       arc investment decision
variables
   Z1           objective function

equations
   EQ_OBJ         complete problem objective function
   Offer    (i  ) offer  at origin
   Demand   (  j) demand at destination
   FlowLimit(i,j) arc flow limit ;

EQ_OBJ         .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(i,j), C(i,j)*X(i,j)] ;
Offer    (i  ) .. sum[j, X(i,j)] =l= A(i) ;
Demand   (  j) .. sum[i, X(i,j)] =g= B(j) ;
FlowLimit(i,j) .. X(i,j) =l= min[A(i),B(j)] * Y(i,j) ;

model Complete      / EQ_OBJ Offer Demand FlowLimit / ;

X.up(i,j) = min[A(i),B(j)]

solve Complete using MIP minimizing Z1
```

```
$$Title Stochastic fixed-charge transportation problem (SFCTP)

* relative optimality tolerance in solving MIP problems
option OptcR = 0, Decimals = 6

sets
   I         origins             / i1 * i4 /
   J         destinations        / j1 * j3 /
   S         scenarios           / s000 * s099 /

parameters
   A(i)      product offer       / i1 20, i2 30, i3 40, i4 20 /
   B(j)      product demand      / j1 21, j2 51, j3 31 /
   P(s)      scenario probability
   BS(s,  j) product demand stochastic ;

BS(s,j) = B(j) * [1+uniform(-0.05,0.05)] ;
P (s)   = 1/card(s) ;

table C(i,j) per unit variable transportation cost
       j1  j2  j3
   i1   1   2   3
   i2   3   2   1
   i3   2   3   4
   i4   4   3   2 ;

table F(i,j) fixed transportation cost
       j1  j2  j3
   i1  10  20  30
   i2  20  30  40
   i3  30  40  50
   i4  40  50  60 ;

loop (s, abort $(sum[i, A(i)] < sum[j, BS(s,j)]) 'Infeasible problem' )

positive variable
   X(s,i,j)     arc flow
binary   variable
   Y(  i,j)     arc investment decision
variables
   Z1           objective function

equations
   EQ_OBJ           complete problem objective function
   Offer    (s,i  ) offer  at origin
   Demand   (s,  j) demand at destination
   FlowLimit(s,i,j) arc flow limit ;

EQ_OBJ           .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(s,i,j), P(s)*C(i,j)*X(s,i,j)] ;
Offer    (s,i  ) .. sum[j, X(s,i,j)] =l= A (  i) ;
Demand   (s,  j) .. sum[i, X(s,i,j)] =g= BS(s,j) ;
FlowLimit(s,i,j) .. X(s,i,j) =l= 100 * Y(i,j) ;

model Complete      / EQ_OBJ Offer Demand FlowLimit / ;

X.up(s,i,j) = 100 ;

Complete.OptFile = 1 ;

file COPT / cplex.opt / ;
put  COPT putclose 'writelp FCTP_Sto.lp' / ;

solve Complete using MIP minimizing Z1

display Z1.l, Y.l
```

# Stochastic FCTP with EMP (Extended Mathematical Programming) (https://www.GAMS.com/latest/docs/UG_EMP_SP.html)

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/FCTP_EMP.gms

```
$title Deterministic fixed-charge transportation problem (FCTP)

* relative optimality tolerance in solving MIP problems
option OptcR = 0, Decimals = 6

sets
   I         origins          / i1 * i4 /
   J         destinations     / j1 * j3 / ;

parameters
   A(i)      product offer   / i1 20, i2 30, i3 40, i4 20 /
   B(j)      product demand  / j1 21, j2 51, j3 31 / ;

table C(i,j) per unit variable transportation cost
       j1  j2  j3
   i1   1   2   3
   i2   3   2   1
   i3   2   3   4
   i4   4   3   2 ;

table F(i,j) fixed transportation cost
       j1  j2  j3
   i1  10  20  30
   i2  20  30  40
   i3  30  40  50
   i4  40  50  60 ;

positive variable
   X(i,j)       arc flow

binary   variable
   Y(i,j)       arc investment decision

variables
   Z1           objective function

equations
   EQ_OBJ         complete problem objective function
   Offer     (i  ) offer  at origin
   Demand    (  j) demand at destination
   FlowLimit(i,j) arc flow limit ;

EQ_OBJ         .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(i,j), C(i,j)*X(i,j)] ;
Offer    (i  ) .. sum[j, X(i,j)] =l= A(i) ;
Demand   (  j) .. sum[i, X(i,j)] =g= B(j) ;
FlowLimit(i,j) .. X(i,j) =l= 100 * Y(i,j) ;

model Complete / all / ;

X.up(i,j) = 100 ;
```

```
set S scenarios / s000 * s099 /
parameter
   BS(s,  j) product demand
   YS(s,i,j) arc investment decision
   XS(s,i,j) arc flow
   P (s    ) scenario probability ;

BS(s,j) = B(j) * [1+uniform(-0.05,0.05)] ;
P (s)   = 1/card(s) ;

* EMP annotations
file emp / '%emp.info%' / ; emp.pc=2 ; emp.pw=1020
* define probability and values of the stochastic parameter
put  emp '* problem %GAMS.i%' / 'jrandvar '
loop (j,
   put B.tn(j) ' '
)
loop (s,
   put P(s)
   loop (j,
      put BS(s,j) /
   )
)
* define stochastic parameter, variable and constraints of the second stage
putclose emp / 'stage 2 B X Offer Demand FlowLimit'

set dict / s . scenario . ''
           B . randvar  . BS
           X . level    . XS
           Y . level    . YS / ;

loop (s, abort $(sum[i, A(i)] < sum[j, BS(s,j)]) 'Infeasible problem') ;

file COPT / cplex.opt / ;
put  COPT putclose 'writelp FCTP_EMP.lp' / 'names 1' / ;

file DOPT / de.opt / ;
put  DOPT putclose 'subsolver CPLEX' / 'subsolveropt 1' / ;

Complete.OptFile = 1 ;

solve Complete minimizing Z1 using emp scenario dict

display Z1.l, YS
```

It seems that the LP file created by the EMP has only 2 decimals

# Stochastic FCTP solved with Benders using EMP

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/FCTP_StoBd_EMP.gms

```
$Title Fixed-charge transportation problem (FCTP) solved by Benders decomposition

* relative optimality tolerance in solving MIP problems
option OptcR = 0, Decimals = 6

sets
   L         iterations        / l001 * l200 /
   LL(l)     iterations subset
   I         origins           / i1 * i4  /
   J         destinations      / j1 * j3  /
   S         scenarios         / s000 * s099 /

parameters
   A(i)      product offer        / i1 20, i2 30, i3 40, i4 20 /
   B(j)      product demand       / j1 21, j2 51, j3 31        /
   P(s)      scenario probability
   YS(s,i,j) arc investment decision
   XS(s,i,j) arc flow
   NS(s,  j) demand not served
   PI(s,i,j) dual variables of second stage constraints
   PTY       penalty for demand not served / 1000 / ;

table C(i,j) per unit variable transportation cost
       j1  j2  j3
   i1   1   2   3
   i2   3   2   1
   i3   2   3   4
   i4   4   3   2 ;

table F(i,j) fixed transportation cost
       j1  j2  j3
   i1  10  20  30
   i2  20  30  40
   i3  30  40  50
   i4  40  50  60 ;

parameter
   BS(s,  j) product demand ;

BS(s,j) = B(j) * [1+uniform(-0.05,0.05)] ;
P (s)   = 1/card(s) ;

abort $(sum[i, A(i)] < sum[j, B(j)]) 'Infeasible problem'

parameters
   BdTol              relative Benders tolerance / 1e-6 /
   Z_Lower            lower bound                / -inf /
   Z_Upper            upper bound                /  inf /
   Y_L    (l,  i,j)   first stage variables values               in iteration l
   PI_L   (l,s,i,j)   dual variables of second stage constraints in iteration l
   Delta  (l)         cut type (feasibility 0 optimality 1)      in iteration l
   Z2_L   (l)         subproblem objective function value        in iteration l

positive variable
   X(i,j)       arc flow
   N(  j)       demand not served

binary   variable
   Y(i,j)       arc investment decision

variables
   Z1           first  stage objective function
   Z2           second stage objective function
   Theta        recourse function
```

```
equations
   EQ_Z1          first  stage     objective function
   EQ_Z2          second stage     objective function
   EQ_OBJ         complete problem objective function
   Offer    (i  ) offer  at origin
   Demand   (  j) demand at destination
   FlowLimit(i,j) arc flow limit
   Bd_Cuts     (l) Benders cuts ;

EQ_Z1          .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + Theta ;
EQ_Z2          .. Z2 =e=                             sum[(i,j), C(i,j)*X(i,j)] + sum[j, PTY*N(j)] ;
EQ_OBJ         .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(i,j), C(i,j)*X(i,j)] + sum[j, PTY*N(j)] ;
Offer    (i  ) .. sum[j, X(i,j)]         =l= A(i) ;
Demand   (  j) .. sum[i, X(i,j)] + N(j) =g= B(j) ;
FlowLimit(i,j) .. X(i,j) =l= 100 * Y(i,j) ;
Bd_Cuts(ll)    .. Delta(ll) * Theta =g= Z2_L(ll) + sum[(s,i,j), (Y(i,j)-Y_L(ll,i,j)) * 100 * PI_L(ll,s,i,j)] ;

model Master_Bd     / EQ_Z1  Bd_Cuts                  / ;
model Subproblem_Bd / EQ_Z2  Offer Demand FlowLimit / ;
model Complete      / EQ_OBJ Offer Demand FlowLimit / ;

X.up(i,j) = 100 ;

* EMP annotations
file emp / '%emp.info%' / ; emp.pc=2 ; emp.pw=1020
* define probability and values of the stochastic parameter
put  emp '* problem %GAMS.i%' / 'jrandvar '
loop (j,
   put B.tn(j) ' '
)
loop (s,
   put P(s)
   loop (j,
      put BS(s,j) /
   )
)
* define stochastic parameter, variable and constraints of the second stage
putclose emp / 'stage 2 B X Offer Demand FlowLimit'

set dict / s         . scenario . ''
           B         . randvar  . BS
           X         . level    . XS
           N         . level    . NS
           Y         . level    . YS
           FlowLimit . marginal . PI / ;

loop (s, abort $(sum[i, A(i)] < sum[j, BS(s,j)]) 'Infeasible problem' ) ;

* to allow CPLEX correctly detect rays in an infeasible problem
* only simplex method can be used and no preprocessing neither scaling options
* optimality and feasibility tolerances are very small to avoid primal degeneration

Subproblem_Bd.OptFile = 1 ;

file COPT / cplex.opt / ;
put  COPT putclose 'writelp DEP.lp' / 'names 1' / ;

file DOPT / de.opt / ;
put  DOPT putclose 'subsolver CPLEX' / 'subsolveropt 1' / ;
```

```
* parameter initialization

LL   (l)       = no ;
Delta(l)       =  0 ;
Z2_L (l)       =  0 ;
PI_L (l,s,i,j) =  0 ;
Y_L  (l,  i,j) =  0 ;

* Benders algorithm iterations
Theta.fx     =  0 ;
loop (l $(abs(1-Z_Lower/Z_Upper) > BdTol),

*  solving master problem
   solve Master_Bd using MIP minimizing Z1 ;

*  storing the master solution
   Y_L(l,i,j) = Y.l(i,j) ;

*  fixing first-stage variables and solving subproblem
   Y.fx( i,j) = Y.l(i,j) ;

*  solving subproblem

   solve Subproblem_Bd minimizing Z2 using emp scenario dict

*  storing parameters to build a new Benders cut
   if (Subproblem_Bd.ModelStat = 4,
      Delta(l) = 0 ;
      Z2_L (l) = Subproblem_Bd.SumInfes ;
   else
*     updating lower and upper bound
      Z_Lower =                Z1.l ;
      Z_Upper = min(Z_Upper, Z1.l - Theta.l + Z2.l) ;

      Theta.lo = -inf ;
      Theta.up =  inf ;

      Delta(l) = 1 ;
      Z2_L (l) = Z2.l ;
   ) ;

   PI_L(l,s,i,j) = PI(s,i,j) ;

   Y.lo(    i,j) = 0 ;
   Y.up(    i,j) = 1 ;

*  increase the set of Benders cuts
   LL(l) = yes ;
) ;

display Z_Lower, Z_Upper, Y.l
```

# Stochastic FCTP solved with Benders using Guss

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/FCTP_StoBd_Guss.gms

```
$Title Fixed-charge transportation problem (FCTP) solved by Benders decomposition

* relative optimality tolerance in solving MIP problems
option OptcR = 0, Decimals = 6

sets
   L        iterations        / l001 * l200 /
   LL(l)    iterations subset
   I        origins           / i1 * i4  /
   J        destinations      / j1 * j3  /
   S        scenarios         / s000 * s099 /

parameters
   A(i)       product offer        / i1 20, i2 30, i3 40, i4 20 /
   B(j)       product demand       / j1 21, j2 51, j3 31        /
   BS(s,  j) product demand
   P (s)      scenario probability
   YS(s,i,j) arc investment decision
   XS(s,i,j) arc flow
   NS(s,  j) demand not served
   PI(s,i,j) dual variables of second stage constraints
   PTY       penalty for demand not served / 1000 / ;

table C(i,j) per unit variable transportation cost
      j1  j2  j3
   i1  1   2   3
   i2  3   2   1
   i3  2   3   4
   i4  4   3   2 ;

table F(i,j) fixed transportation cost
      j1  j2  j3
   i1 10  20  30
   i2 20  30  40
   i3 30  40  50
   i4 40  50  60 ;

BS(s,j) = B(j) * [1+uniform(-0.05,0.05)] ;
P(s) = 1/card(s) ;

abort $(sum[i, A(i)] < sum[j, B(j)]) 'Infeasible problem'

parameters
   BdTol              relative Benders tolerance / 1e-6 /
   Z_Lower            lower bound                / -inf /
   Z_Upper            upper bound                /  inf /
   Y_L    (l,  i,j) first stage variables values              in iteration l
   PI_L   (l,s,i,j) dual variables of second stage constraints in iteration l
   Delta  (l)        cut type (feasibility 0 optimality 1)     in iteration l
   Z2_L   (l,s)      subproblem objective function value       in iteration l
   Z2S    (s)        subproblem objective function value       in iteration l

positive variable
   X(i,j)        arc flow
   N(  j)        demand not served

binary   variable
   Y(i,j)        arc investment decision

variables
   Z1            first  stage objective function
   Z2            second stage objective function
   Theta         recourse function
```

```
equations
   EQ_Z1           first  stage     objective function
   EQ_Z2           second stage     objective function
   EQ_OBJ          complete problem objective function
   Offer    (i  ) offer  at origin
   Demand   (  j) demand at destination
   FlowLimit(i,j) arc flow limit
   Bd_Cuts     (l) Benders cuts ;

EQ_Z1          .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + Theta ;
EQ_Z2          .. Z2 =e=                              sum[(i,j), C(i,j)*X(i,j)] + sum[j, PTY*N(j)] ;
EQ_OBJ         .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(i,j), C(i,j)*X(i,j)] + sum[j, PTY*N(j)] ;
Offer    (i  ) .. sum[j, X(i,j)]         =l= A(i) ;
Demand   (  j) .. sum[i, X(i,j)] + N(j) =g= B(j) ;
FlowLimit(i,j) .. X(i,j) =l= 100 * Y(i,j) ;
Bd_Cuts(ll)    .. Delta(ll) * Theta =g= sum[s, P(s) * (Z2_l(ll,s) + sum[(i,j), (Y(i,j)-Y_L(ll,i,j)) * 100 *
PI_L(ll,s,i,j)])] ;

model Master_Bd     / EQ_Z1  Bd_Cuts                / ;
model Subproblem_Bd / EQ_Z2  Offer Demand FlowLimit / ;
model Complete      / EQ_OBJ Offer Demand FlowLimit / ;

X.up(i,j) = 100 ;

set
   scen_dem stochastic demand scenario dictionary /
   s         . scenario . ''

*  update   the LHS with values of the RHS
   B         . param    . BS

*  store in the RHS with values of the LHS
   X         . level    . XS
   N         . level    . NS
   Y         . level    . YS
   Z2        . level    . Z2S
   FlowLimit . marginal . PI / ;

parameter
   scen_optn         scenario options / OptFile 2, LogOption 1, SkipBaseCase 1,
                                        UpdateType 1, RestartType 1, NoMatchLimit 999 /

loop (s, abort $(sum[i, A(i)] < sum[j, BS(s,j)]) 'Infeasible problem' ) ;

* to allow CPLEX correctly detect rays in an infeasible problem
* only simplex method can be used and no preprocessing neither scaling options
* optimality and feasibility tolerances are very small to avoid primal degeneration

file COPT / cplex.opt / ;
put  COPT putclose 'ScaInd -1' / 'LPMethod 1' / 'PreInd 0' / 'EpOpt 1e-9' / 'EpRHS 1e-9' / ;

Subproblem_Bd.OptFile = 1 ;
```

```
* parameter initialization
LL   (l)       = no ;
Delta(l)       =  0 ;
Z2_L (l,s)     =  0 ;
PI_L (l,s,i,j) =  0 ;
Y_L  (l,  i,j) =  0 ;

* Benders algorithm iterations
Theta.fx    =  0 ;
loop (l $(abs(1-Z_Lower/Z_Upper) > BdTol),

*  solving master problem
   solve Master_Bd using MIP minimizing Z1 ;

*  storing the master solution
   Y_L(l,i,j) = Y.l(i,j) ;

*  fixing first-stage variables and solving subproblem
   Y.fx( i,j) = Y.l(i,j) ;

*  initialization of the destination demand
   B(j) = BS('s000',j) ;

*  solving subproblem
   solve Subproblem_Bd minimizing Z2 using RMIP scenario scen_dem ;

*  storing parameters to build a new Benders cut
   if (Subproblem_Bd.ModelStat = 4,
      Delta(l) = 0 ;
      Z2_L (l,s) = Subproblem_Bd.SumInfes ;
   else
*     updating lower and upper bound
      Z_Lower =                 Z1.l ;
      Z_Upper = min(Z_Upper, Z1.l - Theta.l + sum[s, P(s) * Z2S(s)]) ;

      Theta.lo = -inf ;
      Theta.up =  inf ;

      Delta(l  ) = 1 ;
      Z2_L (l,s) = Z2S(s) ;
   ) ;

   PI_L(l,s,i,j) = PI(s,i,j) ;

   Y.lo(    i,j) = 0 ;
   Y.up(    i,j) = 1 ;

*  increase the set of Benders cuts
   LL(l) = yes ;
) ;

display Z_Lower, Z_Upper, Y.l
```

# Stochastic FCTP solved with Benders using Guss&Grid

https://github.com/IIT-EnergySystemModels/Fixed-Charge-Transportation-Problem-Benders-Decomposition/blob/main/FCTP_StoBd_GG.gms

```
$Title Fixed-charge transportation problem (FCTP) solved by Benders decomposition

* relative optimality tolerance in solving MIP problems
option OptcR = 0, Decimals = 6

sets
   L         iterations        / l001 * l200 /
   LL(l)     iterations subset
   I         origins           / i1 * i4  /
   J         destinations      / j1 * j3  /
   S         scenarios         / s000 * s099 /

parameters
   A(i)      product offer        / i1 20, i2 30, i3 40, i4 20 /
   B(j)      product demand       / j1 21, j2 51, j3 31        /
   BS(s,  j) product demand
   P (s)     scenario probability
   YS(s,i,j) arc investment decision
   XS(s,i,j) arc flow
   NS(s,  j) demand not served
   PI(s,i,j) dual variables of second stage constraints
   PTY       penalty for demand not served / 1000 / ;

table C(i,j) per unit variable transportation cost
       j1  j2  j3
   i1   1   2   3
   i2   3   2   1
   i3   2   3   4
   i4   4   3   2 ;

table F(i,j) fixed transportation cost
       j1  j2  j3
   i1  10  20  30
   i2  20  30  40
   i3  30  40  50
   i4  40  50  60 ;

BS(s,j) = B(j) * [1+uniform(-0.05,0.05)] ;
P(s) = 1/card(s) ;

abort $(sum[i, A(i)] < sum[j, B(j)]) 'Infeasible problem'

parameters
   BdTol               relative Benders tolerance / 1e-6 /
   Z_Lower             lower bound                / -inf /
   Z_Upper             upper bound                /  inf /
   Y_L    (l,   i,j) first stage variables values              in iteration l
   PI_L   (l,s,i,j) dual variables of second stage constraints in iteration l
   Delta  (l)        cut type (feasibility 0 optimality 1)     in iteration l
   Z2_L   (l,s)      subproblem objective function value       in iteration l
   Z2_S   (s)        subproblem objective function value       in iteration l

positive variable
   X(i,j)       arc flow
   N(  j)       demand not served

binary   variable
   Y(i,j)       arc investment decision

variables
   Z1           first  stage objective function
   Z2           second stage objective function
   Theta        recourse function
```

```
equations
   EQ_Z1             first  stage     objective function
   EQ_Z2             second stage     objective function
   EQ_OBJ            complete problem objective function
   Offer    (i  ) offer  at origin
   Demand   (  j) demand at destination
   FlowLimit(i,j) arc flow limit
   Bd_Cuts    (l) Benders cuts ;

EQ_Z1           .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + Theta ;
EQ_Z2           .. Z2 =e=                                sum[(i,j), C(i,j)*X(i,j)] + sum[j, PTY*N(j)] ;
EQ_OBJ          .. Z1 =e= sum[(i,j), F(i,j)*Y(i,j)] + sum[(i,j), C(i,j)*X(i,j)] + sum[j, PTY*N(j)] ;
Offer    (i  ) .. sum[j, X(i,j)]        =l= A(i) ;
Demand   (  j) .. sum[i, X(i,j)] + N(j) =g= B(j) ;
FlowLimit(i,j) .. X(i,j) =l= 100 * Y(i,j) ;
Bd_Cuts(ll)    .. Delta(ll) * Theta =g= sum[s, P(s) * (Z2_L(ll,s) + sum[(i,j), (Y(i,j)-Y_L(ll,i,j)) * 100 *
PI_L(ll,s,i,j))])] ;

model Master_Bd     / EQ_Z1  Bd_Cuts                / ;
model Subproblem_Bd / EQ_Z2  Offer Demand FlowLimit / ;
model Complete      / EQ_OBJ Offer Demand FlowLimit / ;

X.up(i,j) = 100 ;

set
   gs(s)    scenarios per GUSS run
   sh       solution headers / System.GUSSModelAttributes /
   scen_dem stochastic demand scenario dictionary /
   s         . scenario . ''
*   scen_optn . opt      . st_report_o

*  update   the LHS with values of the RHS
   B         . param    . BS

*  store in the RHS with values of the LHS
   X         . level    . XS
   N         . level    . NS
   Y         . level    . YS
   Z2        . level    . Z2_s
   FlowLimit . marginal . PI /

sets
*  using four cores and assignment of scenarios to cores
   core            grid jobs to run   / core001*core004     /
   coresc(core,s) cores to scenario / core001.(s000*s024)
                                      core002.(s025*s049)
                                      core003.(s050*s074)
                                      core004.(s075*s099) /

parameter
*   scen_optn         scenario options / OptFile 2, LogOption 1, SkipBaseCase 1,
*                                        UpdateType 1, RestartType 1, NoMatchLimit 999 /
*   st_report_o(s,sh) status report
   pGridHandle(core) grid handles ;

loop (s, abort $(sum[i, A(i)] < sum[j, BS(s,j)]) 'Infeasible problem' ) ;

* to allow CPLEX correctly detect rays in an infeasible problem
* only simplex method can be used and no preprocessing neither scaling options
* optimality and feasibility tolerances are very small to avoid primal degeneration

file COPT / cplex.opt / ;
put  COPT putclose 'ScaInd -1' / 'LPMethod 1' / 'PreInd 0' / 'EpOpt 1e-9' / 'EpRHS 1e-9' / ;
```

```
Subproblem_Bd.OptFile = 1 ;

* parameter initialization
LL   (l)        = no ;
Delta(l)        =  0 ;
Z2_L (l,s)      =  0 ;
PI_L (l,s,i,j) =  0 ;
Y_L  (l,  i,j) =  0 ;

* Benders algorithm iterations
Theta.fx     =  0 ;
loop (l $(abs(1-Z_Lower/Z_Upper) > BdTol),

*  solving master problem
   solve Master_Bd using MIP minimizing Z1 ;

*  storing the master solution
   Y_L(l,i,j) = Y.l(i,j) ;

*  fixing first-stage variables and solving subproblem
   Y.fx( i,j) = Y.l(i,j) ;

*  initialization of the destination demand
   B(j) = BS('s000',j) ;

*  solving subproblem

   Subproblem_Bd.SolveLink = %SolveLink.AsyncGrid% ;
*  Sending loop
   loop (core,
      gs(s) = coresc(core,s)
      if (sum[gs(s), 1] > 0,
         solve Subproblem_Bd minimizing Z2 using RMIP scenario
scen_dem ;
         pGridHandle(core) = Subproblem_Bd.Handle ;
      ) ;
   ) ;
*  Recovering loop
   repeat
      loop (core $HandleCollect(pGridHandle(core)),
         display $HandleDelete (pGridHandle(core)) 'Trouble
deleting handles' ;
         pGridHandle(core) = 0 ;
      ) ;
   until card(pGridHandle) = 0 or TimeElapsed > 1000 ;
   Subproblem_Bd.SolveLink = %SolveLink.LoadLibrary% ;

*  storing parameters to build a new Benders cut
   if (Subproblem_Bd.ModelStat = 4,
      Delta(l) = 0 ;
      Z2_L (l,s) = Subproblem_Bd.SumInfes ;
   else
*     updating lower and upper bound
      Z_Lower =               Z1.l ;
      Z_Upper = min(Z_Upper, Z1.l - Theta.l + sum[s, P(s) *
Z2_s(s)]) ;

      Theta.lo = -inf ;
      Theta.up =  inf ;

      Delta(l  ) = 1 ;
      Z2_L (l,s) = Z2_S(s) ;
   ) ;

   PI_L(l,s,i,j) = PI(s,i,j) ;

   Y.lo(    i,j) = 0 ;
   Y.up(    i,j) = 1 ;

*  increase the set of Benders cuts
   LL(l) = yes ;
) ;

display Z_Lower, Z_Upper, Y.l
```

# Transportation problem solved as MCP (KKT conditions)

```
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pB(j) destination demand
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

table pC(i,j) per unit transportation cost
          MADRID BARCELONA VALENCIA
VIGO       0.06     0.12      0.09
ALGECIRAS  0.05     0.15      0.11

variables
   vX(i,j) units transported
   vCost   transportation cost

positive variable vX

equations
   eCost          transportation cost
   eCapacity(i) maximum capacity of each origin
   eDemand   (j) demand supply at destination ;

eCost          .. sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) .. sum[   j ,          vX(i,j)] =l= pA(i) ;
eDemand   (j) .. sum[ i   ,          vX(i,j)] =g= pB(j) ;

model mTransport / all /
solve mTransport using LP minimizing vCost
```

$$\min_{x_{ij}} \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} \le a_i \quad \forall i$$

$$\sum_i x_{ij} \ge b_j \quad \forall j$$

$$x_{ij} \ge 0$$

$$\mathcal{L} = \sum_{ij} c_{ij} x_{ij} + \alpha_i \left( \sum_j x_{ij} - a_i \right) + \beta_j \left( b_j - \sum_i x_{ij} \right)$$

$$\frac{\partial L}{\partial x_{ij}} \rightarrow$$

$$c_{ij} + \alpha_i \ge \beta_j \qquad : x_{ij} \quad \forall ij$$

$$-\sum_j x_{ij} \ge -a_i \qquad : \alpha_i \quad \forall i$$

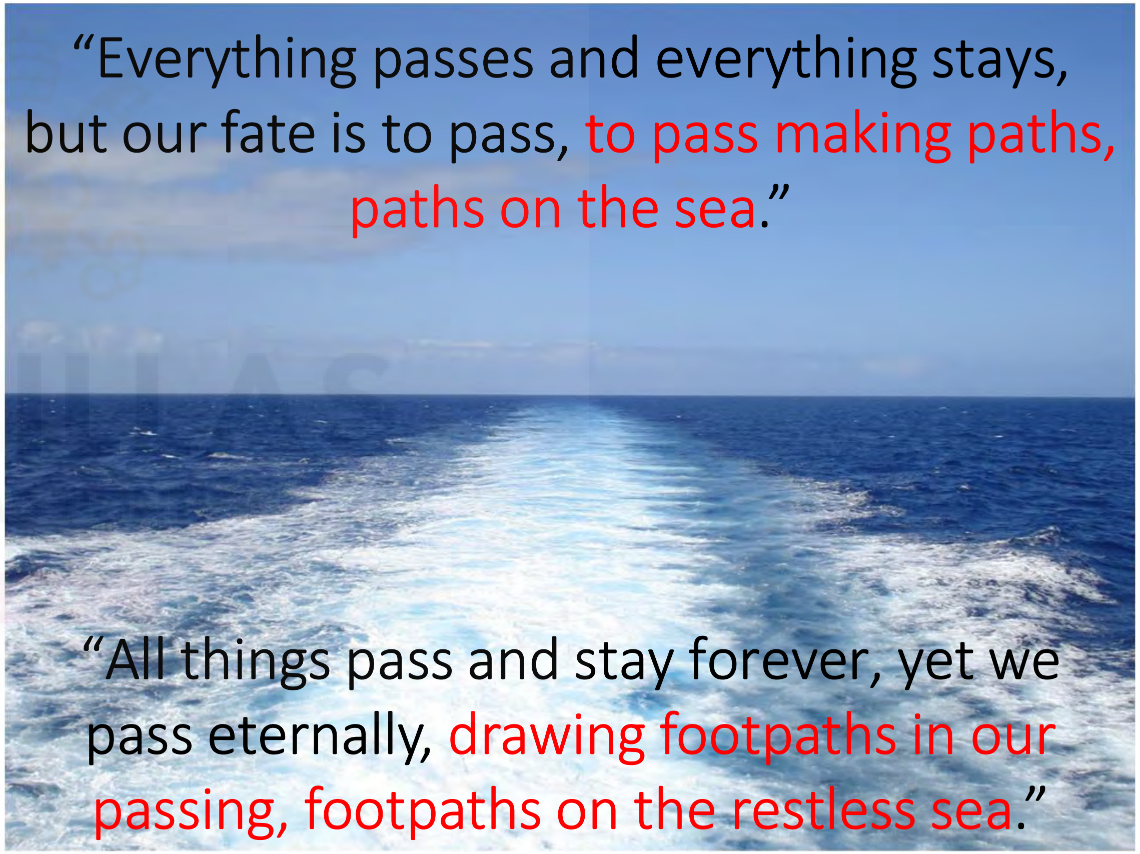
$$\sum_i x_{ij} \ge b_j \qquad : \beta_j \quad \forall j$$

$$x_{ij}, \alpha_i, \beta_j \ge 0$$

```
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pB(j) destination demand
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

table pC(i,j) per unit transportation cost
          MADRID BARCELONA VALENCIA
VIGO       0.06     0.12      0.09
ALGECIRAS  0.05     0.15      0.11

variables
   vX(i,j) units transported
   vA(i  ) Lagrange multiplier of capacity constraint
   vB(  j) Lagrange multiplier of demand   constraint

positive variables vX, vA, vB

equations
   eProfit(i,j) marginal cost >= marginal profit
   eCapacity(i) maximum capacity of each origin
   eDemand   (j) demand supply at destination ;

eProfit(i,j) ..  vA(i) + pC(i,j)  =g=  vB(j) ;
eCapacity(i) .. -sum[j,  vX(i,j)] =g= -pA(i) ;
eDemand   (j) ..  sum[i,  vX(i,j)] =g=  pB(j) ;

model mTransport / eProfit.vX eCapacity.vA eDemand.vB /
solve mTransport using MCP
```

# Antonio Machado. Cantares

“Todo pasa y todo queda,
pero lo nuestro es pasar,
pasar haciendo caminos,
caminos sobre el mar.”



Enjoy Formulating, writing and solving optimization models

*Thank you for your attention*

Prof. Andres Ramos

https://www.iit.comillas.edu/aramos/

Andres.Ramos@comillas.edu

Pedro de Otaola

Pedro.Otaola@comillas.edu