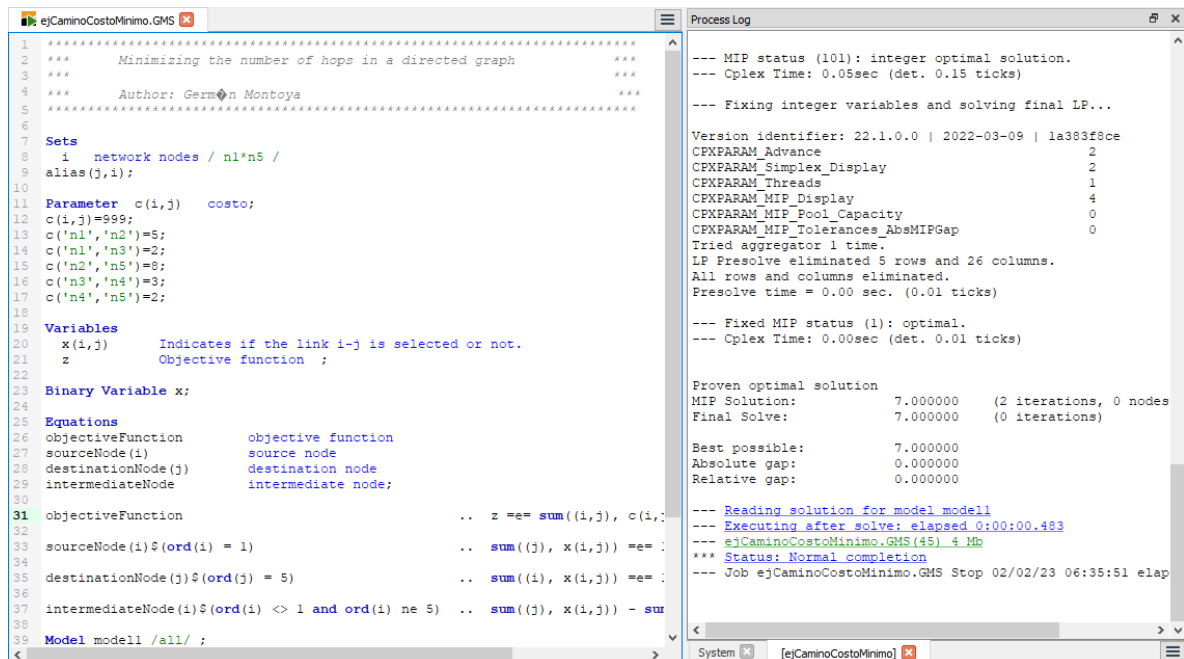


GAMS:

Licencia:

GAMS_Community_License_for_Nicolas_Klopstock_G230125|0002AO-GEN
Universidad_de los Andes, Computer_and_Systems_Engineering_____
1790724205_____
205523510C_____
CL4985_____A_COMMUNITY_____
Nicolas_Klopstock,_n.klopstock@uniandes.edu.co_____

ejCaminoCostoMínimo.GMS



The screenshot displays the GAMS software interface. The main window shows the model code for 'ejCaminoCostoMínimo.GMS'. The code is as follows:

```
1 *****
2 ***      Minimizing the number of hops in a directed graph      ***
3 ***
4 ***      Author: Germán Montoya      ***
5 *****
6
7 Sets
8   i   network nodes / n1*n5 /
9   alias(j,i);
10
11 Parameter c(i,j)  costo;
12 c(i,j)=999;
13 c('n1','n2')=5;
14 c('n1','n3')=2;
15 c('n2','n5')=8;
16 c('n3','n4')=3;
17 c('n4','n5')=2;
18
19 Variables
20   x(i,j)          Indicates if the link i-j is selected or not.
21   z               Objective function ;
22
23 Binary Variable x;
24
25 Equations
26   objectiveFunction      objective function
27   sourceNode(i)          source node
28   destinationNode(j)     destination node
29   intermediateNode       intermediate node;
30
31   objectiveFunction      .. z =e= sum((i,j), c(i,j)*x(i,j));
32
33   sourceNode(i)$(ord(i) = 1) .. sum((j), x(i,j)) =e= 1;
34
35   destinationNode(j)$(ord(j) = 5) .. sum((i), x(i,j)) =e= 1;
36
37   intermediateNode(i)$(ord(i) <> 1 and ord(i) ne 5) .. sum((j), x(i,j)) - sum((j), x(j,i)) =e= 0;
38
39 Model modell /all/ ;
```

The Process Log window on the right shows the following output:

```
--- MIP status (101): integer optimal solution.
--- Cplex Time: 0.05sec (det. 0.15 ticks)

--- Fixing integer variables and solving final LP...

Version identifier: 22.1.0.0 | 2022-03-09 | la383f8ce
CPXPARAM_Advance          2
CPXPARAM_Simplex_Display  2
CPXPARAM_Threads          1
CPXPARAM_MIP_Display      4
CPXPARAM_MIP_Pool_Capacity 0
CPXPARAM_MIP_Tolerance_AbsMIPGap 0
Tried aggregator 1 time.
LP Presolve eliminated 5 rows and 26 columns.
All rows and columns eliminated.
Presolve time = 0.00 sec. (0.01 ticks)

--- Fixed MIP status (1): optimal.
--- Cplex Time: 0.00sec (det. 0.01 ticks)

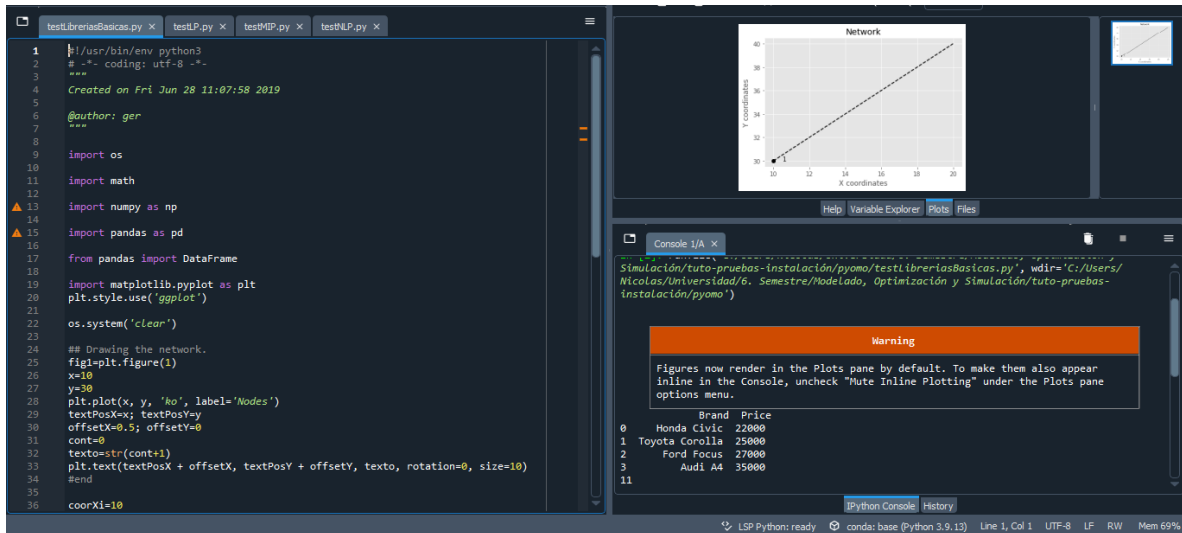
Proven optimal solution
MIP Solution:          7.000000    (2 iterations, 0 nodes)
Final Solve:          7.000000    (0 iterations)

Best possible:          7.000000
Absolute gap:           0.000000
Relative gap:           0.000000

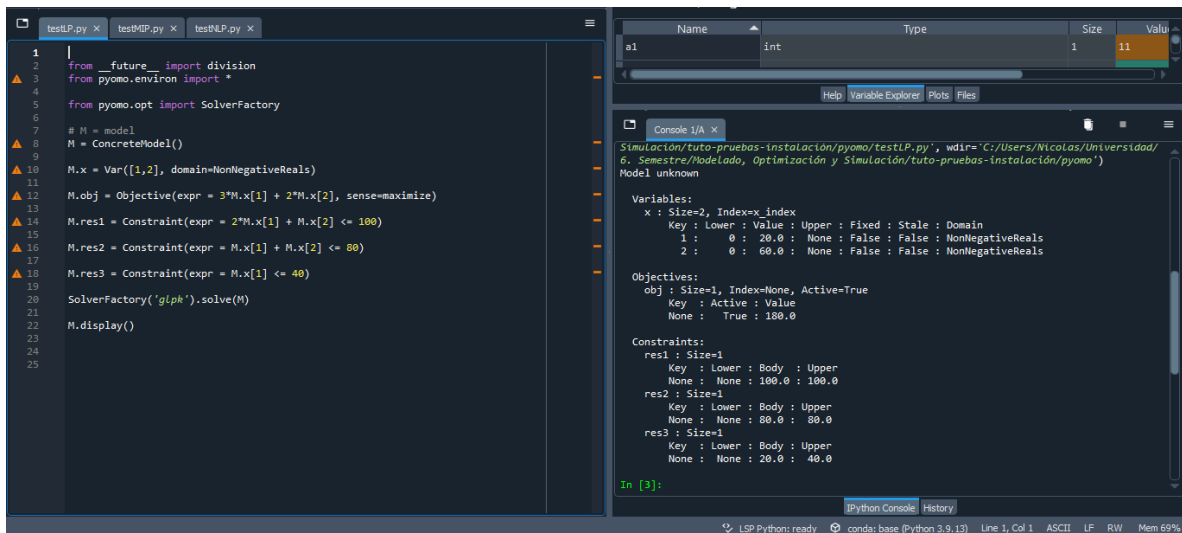
--- Reading solution for model modell
--- Executing after solve: elapsed 0:00:00.483
--- ejCaminoCostoMínimo.GMS(45) 4 Mb
*** Status: Normal completion
--- Job ejCaminoCostoMínimo.GMS Stop 02/02/23 06:35:51 elap
```

Pyomo:

testLibreriasBasicas.py



testLP.py



testMIP.py

The screenshot shows a Jupyter Notebook with a file named `testMIP.py`. The code defines a Mixed-Integer Programming (MIP) model using Pyomo. It includes imports for `division`, `pyomo.environ`, and `pyomo.opt`. The model is created using `ConcreteModel()` and includes a set of nodes (`N`) and a range set (`N-RangeSet`). The objective function is defined as the sum of costs for each node. The constraints are defined using `source rule`. The console output shows the model's status, including the objective value and the status of the constraints.

```

1 from __future__ import division
2 from pyomo.environ import *
3
4 from pyomo.opt import SolverFactory
5
6
7 import sys
8 import os
9
10 os.system("clear")
11
12 #sys.exit("Stopped")
13
14 Model = ConcreteModel()
15
16 # SETS & PARAMETERS*****
17 numNodes=5
18
19 N=RangeSet(1, numNodes)
20
21 cost={(1,1):999, (1,2):5, (1,3):2, (1,4):999, (1,5):999,\
22      (2,1):999, (2,2):999, (2,3):999, (2,4):999, (2,5):8,\
23      (3,1):999, (3,2):999, (3,3):999, (3,4):3, (3,5):999,\
24      (4,1):999, (4,2):999, (4,3):999, (4,4):999, (4,5):2,\
25      (5,1):999, (5,2):999, (5,3):999, (5,4):999, (5,5):999}
26
27
28 # VARIABLES*****
29 Model.x = Var(N,N, domain=Binary)
30
31 # OBJECTIVE FUNCTION*****
32 Model.obj = Objective(expr = sum(Model.x[i,j]*cost[i,j] for i in N for j in N))
33
34 # CONSTRAINTS*****
35 def source rule(Model,i):

```

The console output shows the model's status, including the objective value and the status of the constraints.

```

(4, 2) : 0 : 0.0 : 1 : False : False : Binary
(4, 3) : 0 : 0.0 : 1 : False : False : Binary
(4, 4) : 0 : 0.0 : 1 : False : False : Binary
(4, 5) : 0 : 1.0 : 1 : False : False : Binary
(5, 1) : 0 : 0.0 : 1 : False : False : Binary
(5, 2) : 0 : 0.0 : 1 : False : False : Binary
(5, 3) : 0 : 0.0 : 1 : False : False : Binary
(5, 4) : 0 : 0.0 : 1 : False : False : Binary
(5, 5) : 0 : 0.0 : 1 : False : False : Binary

Objectives:
obj : Size=1, Index=None, Active=True
Key : Active : Value
None : True : 7.0

Constraints:
source : Size=1
Key : Lower : Body : Upper
1 : 1.0 : 1.0 : 1.0
destination : Size=1
Key : Lower : Body : Upper
5 : 1.0 : 1.0 : 1.0
intermediate : Size=3
Key : Lower : Body : Upper
2 : 0.0 : 0.0 : 0.0
3 : 0.0 : 0.0 : 0.0
4 : 0.0 : 0.0 : 0.0

```

testNLP.py

The screenshot shows a Jupyter Notebook with a file named `testNLP.py`. The code defines a Nonlinear Programming (NLP) model using Pyomo. It includes imports for `pyomo.environ` and `pyomo.opt`. The model is created using `ConcreteModel()` and includes a set of nodes (`A`). The objective function is defined as the sum of costs for each node. The constraints are defined using `source rule`. The console output shows the model's status, including the objective value and the status of the constraints.

```

1 # -*- coding: utf-8 -*-
2
3 from pyomo.environ import *
4 from pyomo.opt import SolverFactory
5
6 Model = ConcreteModel()
7
8 A = [1,2,3,4]
9
10 xx = (1:2, 2:6, 3:2, 4:6)
11 yy = (1:1, 2:1, 3:5, 4:5)
12 demanda = (1:100, 2:200, 3:300, 4:400)
13
14 #Model.x = Var(domain=PositiveReals, initialize = 2)
15 #Model.y = Var(domain=PositiveReals, initialize = 1)
16
17 Model.x = Var(domain=PositiveReals)
18 Model.y = Var(domain=PositiveReals)
19
20
21 Model.g = Objective(expr= sum((demanda[i]*((xx[i]-Model.x)**2+(yy[i]-Model.y)**2)**(1/2)
22
23 SolverFactory('ipopt').solve(Model)
24
25 Model.display()
26

```

The console output shows the model's status, including the objective value and the status of the constraints.

```

"clear" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

In [3]: runfile('C:/Users/Nicolas/Universidad/6. Semestre/Modelado, Optimización y Simulación/tuto-pruebas-instalación/pyomo/testNLP.py', wdir='C:/Users/Nicolas/Universidad/6. Semestre/Modelado, Optimización y Simulación/tuto-pruebas-instalación/pyomo')
Model unknown

Variables:
x : Size=1, Index=None
Key : Lower : Value : Upper : Fixed : Stale : Domain
None : 0 : 5.332625897989843 : None : False : False : PositiveReals
y : Size=1, Index=None
Key : Lower : Value : Upper : Fixed : Stale : Domain
None : 0 : 4.5369430173199055 : None : False : False : PositiveReals

Objectives:
g : Size=1, Index=None, Active=True
Key : Active : Value
None : True : 2540.1450233862192

Constraints:
None

In [4]:

```