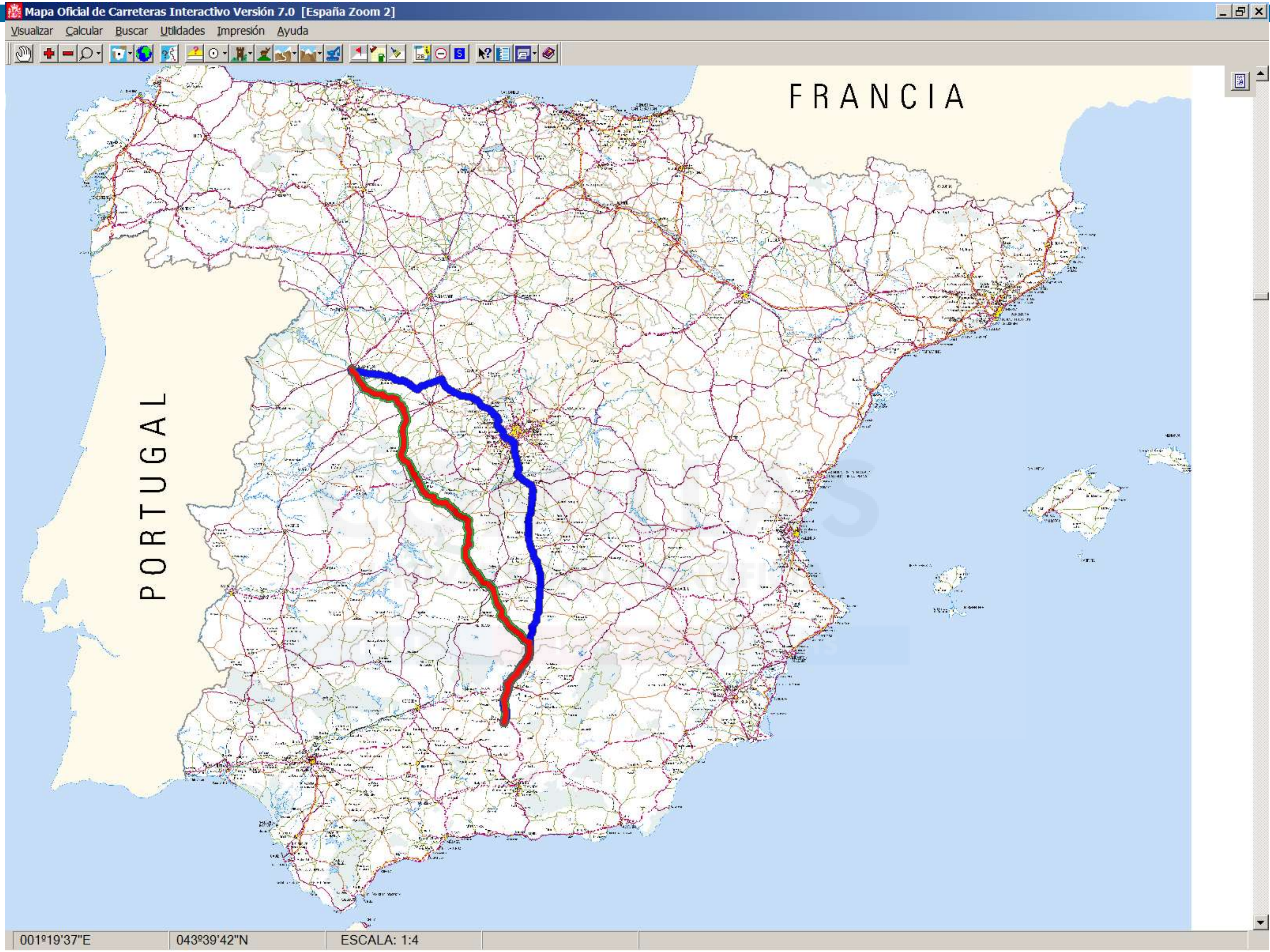
Algebraic modeling languages

Andrés Ramos
Universidad Pontificia Comillas
https://pascua.iit.comillas.edu/aramos/
Andres.Ramos@comillas.edu

Operations Research (OR) definition

- Application of scientific methods in improving operations, decisions, and management effectiveness.
  - Design and improvements in operations and decisions
  - Problem-solving and support in management, planning, or forecasting functions
  - Provide knowledge and help in making decisions
- Tasks:
  - Collect and analyze data
  - Develop and test mathematical models
  - Propose solutions and recommendations
  - Interpret the information
  - Help to implement improvement actions
- Results: computer applications, systems, services, or products.

Shortest path problem



OR History

- Origin at the beginning of World War II (due to the urgent assignment of scarce resources in military operations, in tactical and strategic problems). The same techniques were then applied to companies.
- Rapid initial algorithmic progress (many techniques –LP, DP– were developed before 1960).
    - Game Theory: von Neumann y Morgenstern 1944
    - Simplex Method: Dantzig 1947
    - Optimality Principle: Bellman 1957
- Constant relation with computer hardware advance. Today, solving an LP problem with 1.000.000 equations and 1.000.000 variables in a PC is possible.

What is optimization?

- To find the value of the *variables* that make optimal the *objective function* satisfying the set of *constraints*.

Components of an optimization problem

- *Objective function*
  - Quantitative performance function (fitness) of a system that we want to maximize or minimize
- *Variables*
  - Decisions that influence the objective function
- *Constraints*
  - Set of relations that variables are forced to satisfy

Alternatives for developing optimization models

- General purpose programming languages (C, C++, Java, Visual Basic, FORTRAN 90)
  - C (CPLEX from IBM ILOG, , Gurobi from Gurobi Optimization, Xpress-Optimizer from FICO)
  - C++ (ILOG Concert from IBM, LINDO API from LINDO Systems, OptiMax 2000 from Maximal Software, FLOPC++ from Universidade de Aveiro)
  - Public domain [GNU Linear Programming Toolkit GLPK (www.gnu.org/software/glpk), Computational Infrastructure for Operations Research COIN-OR (www.coin-or.org), LP solver SoPlex (http://soplex.zib.de) and MIP framework SCIP (http://scip.zib.de)]
- Numerical or symbolic languages or environments (spreadsheet, Matlab, Mathematica)
- Algebraic modeling languages [GAMS (https://www.gams.com/), Pyomo (https://pyomo.readthedocs.io/en/stable/), AMPL, IBM ILOG CPLEX Optimization Studio, AIMMS, XPRESS-MP, MPL, Zimpl (http://zimpl.zib.de), JuMP (https://jump.readthedocs.org/en/latest/), OptimJ]
- YALMIP, CVX (both are toolboxes for Matlab)
- Once a year in *OR/MS Today* (www.orms-today.com), there are summary papers on the different optimization environments and their characteristics

Algebraic modeling languages advantages (i)

- High-level languages for the compact formulation of large-scale and complex models
- Easy prototype development
- Improve modelers' productivity
- Structure good modeling habits
- Separation of interface, data, model and solver
- Formulation independent of model size
- Model independent of solvers

Algebraic modeling languages advantages (ii)

- Easy continuous reformulation
- Documentation is made simultaneously with modeling
- Allow to build large maintainable models that can be adapted quickly to new situations
- Allow advanced algorithm implementation
- Easy implementation of NLP, MIP, and MCP
- Open architecture with interfaces to other systems
- Platform independence and portability among platforms and operating systems (MS Windows, Linux, macOS)

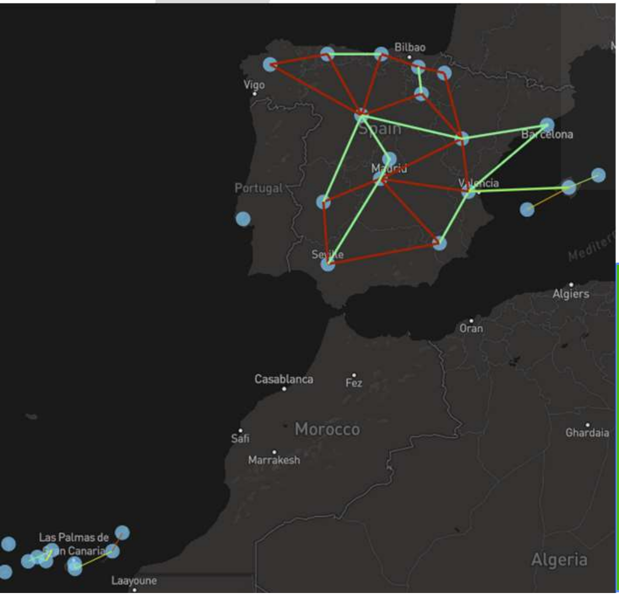
Algebraic modeling languages drawbacks

- Not adequate for sporadic use with small problems
- Not adequate for direct resolution of very large-scale problems (100.000.000 x 100.000.000)

Real applications

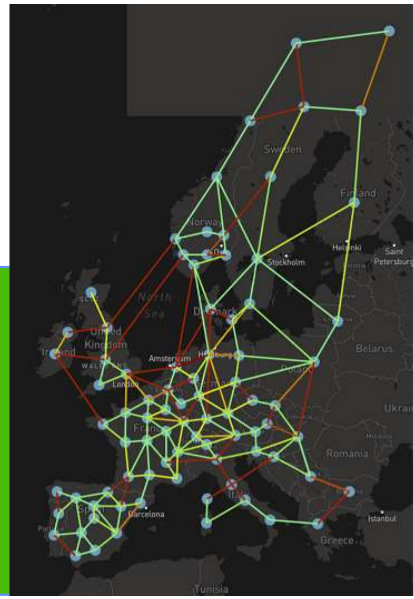
- GAMS is been used extensively to develop decision support models. Commercial since1987. More than 10.000 users in 100 countries
- Problems with up to 22 million constraints, 27 million variables, and 74 million non-zero elements easily solved in a PC with 256 GB RAM

Spain 2030



Case SEP2030
2999548 rows, 3513436 columns, 11508142 nonzeros
Case SEPE2030
5165034 rows, 5454024 columns (1310400 binary) and 21855526 nonzeros
Case ES2030
5162243 rows, 6832942 columns, 21554828 nonzeros
Case TYNDP DE2050
22157904 rows, 26966415 columns, 73814159 nonzeros

Europe TF2030



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 minimum cost.

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

Mathematical formulation

- Objective function

$$\min_{x_{ij}} \sum_i \sum_j c_{ij} x_{ij}$$

- Production limit for each factory *i*

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

- Consumption in each market *j*

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

- Quantity to send from each factory *i* to each market *j*

$$x_{ij} \ge 0 \quad \forall i \to j$$

Transportation model in GAMS (i)

```
SETS
   I fábricas de envasado / VIGO, ALGECIRAS /
   J mercados de consumo  / MADRID, BARCELONA, VALENCIA /

PARAMETERS
   A(i) capacidad de producción de la fábrica i [cajas]
       / VIGO      350
         ALGECIRAS 700 /

   B(j) demanda del mercado j [cajas]
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

TABLE C(i,j) coste transporte entre i y j [€ por caja]
         MADRID BARCELONA VALENCIA
VIGO       0.06      0.12      0.09
ALGECIRAS  0.05      0.15      0.11
```

Transportation model in GAMS (ii)

```
VARIABLES
   X(i,j) cajas transportadas entre fábrica i y mercado j [cajas]
   CT     coste de transporte                             [€]

POSITIVE VARIABLE X

EQUATIONS
   COSTE        coste total de transporte             [€]
   CAPACIDAD(i) capacidad máxima de cada fábrica i    [cajas]
   DEMANDA(j)   satisfacción demanda de cada mercado j [cajas] ;

COSTE ..        CT =E= SUM((i,j), C(i,j) * X(i,j)) ;

CAPACIDAD(i) .. SUM(j, X(i,j)) =L= A(i) ;

DEMANDA(j) ..   SUM(i, X(i,j)) =G= B(j) ;

MODEL TRANSPORTE / COSTE, CAPACIDAD, DEMANDA /

SOLVE TRANSPORTE USING LP MINIMIZING CT
```

Problem library (+ 250 models)

GAMS Studio > GAMS > Model Library Explorer (F6)

https://pascua.iit.comillas.edu/aramos/simio/apuntes/a_casos.pdf

- Production management
- Agricultural economics
- Chemical engineering
- Forest engineering
- International commerce
- Economic development
- Micro and macroeconomics
- General equilibrium models
- Energy economics
- Finance
- Statistics, econometrics
- Management science and operations research

GAMS (General Algebraic Modeling System)

- Development environment GAMS Studio

- Documentation

  https://www.gams.com/latest/docs/UG_MAIN.html#UG_Language_Environment
- Solvers guide

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

- Model: file_name.gms
- Results: file_name.lst
- Process log: file_name.log

General structure of GAMS sentences

- Lines with * in the first column are comments
- No distinction between uppercase and lowercase letters
- Parenthesis (), square bracket [], or braces {} can be used indistinctly to distinguish levels
- Language-reserved words appear in bold
- Sentences end with a ; (that can be suppressed when the following word is a reserved one)

General structure of a model

- Set declaration. Initialization.
- Include and manipulate input data and parameters.

- Variables
- Equations
- Model

- Bounds and initialization of variables
- Solve the optimization problem
- Output of the results

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`

- Values of `inf`, `eps` are valid as data

variables

- There must always be a free variable to represent the value of the objective function. The values of the variables are always kept.
- Type:

`free` (by default)	$-\infty$ to $+\infty$
`positive`	0 to $+\infty$
`negative`	$-\infty$ to 0
`binary`	0 or 1
`integer`	0 to 100

- Suffixes:
  - var_name.`lo` lower bound
  - var_name.`up` upper bound
  - var_name.`l` initial value before and optimal value after
  - var_name.`m` marginal value (reduced cost)
  - var_name.`fx` fixes a variable to a constant

equations

- Blocks:
  - Declaration with explanatory comment
  - Mathematical expressions

- Types: `=e=` =, `=l=` ≤, `=g=` ≥

- Suffixes:
  - eq_name.`lo` lower bound
  - eq_name.`up` upper bound
  - eq_name.`l` initial value before and optimal value after
  - eq_name.`m` marginal value (dual variable or shadow price)

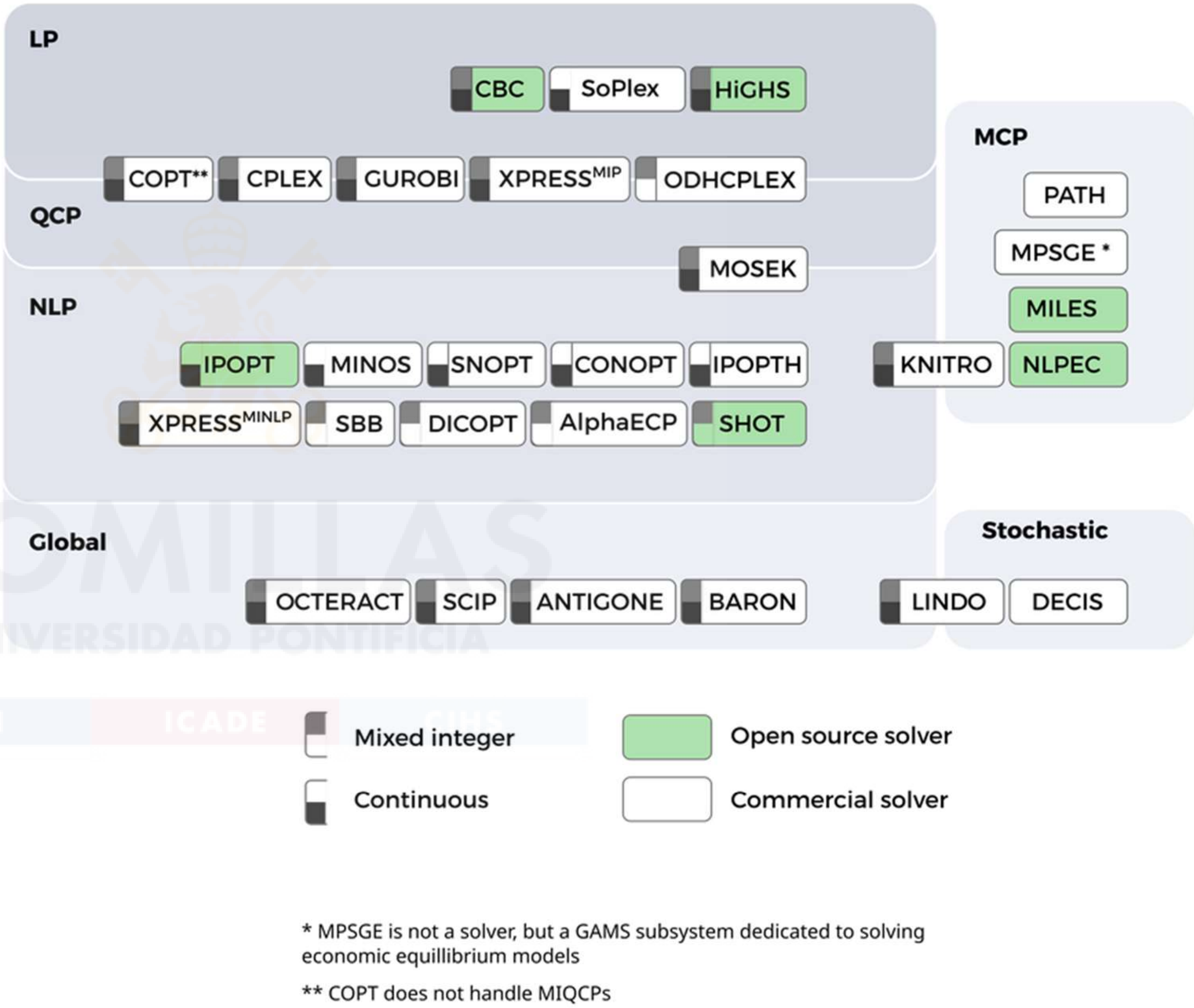
mode and solve

- `model` model_name / equation_names /

  `model` model_name / `all` /

- `solve` model_name `using` problem_type `minimizing` (`maximizing`) o.f._variable

Types of problems and solvers



GAMS Solvers (https://www.gams.com/latest/docs/S_MAIN.html)

$ Operator in assignments, summations, constraints

- Sets a condition

  $(VALUE > 0) $(NUMBER1 <> NUMBER2)

- On the left of an assignment: it does the assignment **ONLY if** the condition is satisfied

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

- On the right of an assignment: 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
);
```

Relational operator

- `lt` <, `gt` >, `eq` =, `ne` <>, `le` <=, `ge` >=

- `not`, `and`, `or`, `xor`

- `diag(set_element,set_element)`={1,0}
- `sameas(set_element,set_element)`={T,F}

Functions

- Elementary: `+`, `-`, `*`, `/`, `**` or `power(x,n)`
- `ord`, `card` ordinal and cardinal of a set
- With indexes: `sum`, `prod`, `smax`, `smin`
- Other functions: `abs, arctan, sin, cos, ceil, floor, exp, log, log10, max, min, mod, round, sign, sqr, sqrt, trunc, normal, uniform`
- Time functions: `gyear, gmonth, gday, ghour, gminute, gsecond, gdow, gleap, jdate, jnow, jstart, jtime`

Dynamic sets

- Subsets of static sets whose content may change by assignments

```
sets M months / 1 * 12 /
     MP(m) even months ;
display m ;
MP(m) $[MOD(ord(m),2) = 0] = YES ;
display mp ;
MP('3') = yes ;
display mp ;
MP(m) $(ord(m) = 3) = NO ;
display mp ;
```

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

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)
```

Transshipment model

- Let's have a set of nodes connected by arcs. A node doesn't have to be connected with the remaining nodes. A node can generate, demand, or transship according to its function. Total offer exceeds total demand. Maximum offer capacity and maximum demand in each node are assumed to be known. Also, each product's unit transportation cost for each arc is known. We want to supply the demand while minimizing the total transportation cost.

- Extend the former problem by adding non-directed arcs.

Index shifting. Lag and lead

- `t=J,F,M,A,MA,J,JU,AU,S,O,N,D`

  `vReserve(t)+pInflow(t)-vOutflow(t) =e= vReserve(t+1)`

- Vector values out of the domain are 0

  `vReserve('D') + pInflow('D') - vOutflow('D') =e= 0`

- Circular sequence of an index (++, --)

  `t=1,...,12`

  `vReserve(t)+pInflow(t)-vOutflow(t) =e= vReserve(t++1)`

  `vReserve('12')+pInflow('12')-vOutflow('12') =e= vReserve('1')`

- Inverted order sequence of `PP` index even though `i` is traversed in increasing order

  `PP(t+[card(t)-2*ord(t)+1])`

Tensor product of two matrices

$$\Phi_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \; ; \; \Phi_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\Phi_1 \otimes \Phi_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 & 4 \end{bmatrix}$$

Underground sellers

The sales booths of an underground station need the following people for 24 hours each day

Interval	Sellers
00 – 06	2
06 – 10	8
10 – 12	4
12 – 16	3
16 – 18	6
18 – 22	5
22 – 24	3

Each seller works 8 hours in two blocks of 4 hours with 1 free hour after the first block. The shift may begin every hour. Determine the minimum number of salespersons to be hired.

Lag and lead exercises

- Distances among crosses
  - Suppose a city with a complete square grid with a unit length on each side. Compute analytically the distance between any two crosses of the city
- Maximum number of knights (queens, castles)
  - Determine by an optimization problem the maximum number of knights (queens, castles) that can be in a chessboard without threat among them

Control flow

- loop (set,

) ;
- while (condition,

) ;
- repeat

  until condition;
- if (condition,

  else

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

) ;

Data input/output

- Data input from a file

  `$include` file_name

- `display` identifier_name (shows its content or value)

- Data output to a file

  `file` internal_name / external_name /

  `put` internal_name

  `put` identifier_name

  `putclose` internal_name

- There are specific options to control the output format

TABLE (i)

- Table with multiple columns

```
sets i / MAD, BCN /
     j / A1, A2, A3, A4, A5, A6 /

table CAPACITY(i,j) maximum capacity
          A1  A2  A3
MAD       1   0   3
BCN       2   1   2
+
          A4  A5  A6
MAD       2   1   3
BCN       3   2   2
```

TABLE (ii)

- Tables with more than 2 dimensions

```
sets i / MAD, BCN /
     j / A1, A2, A3, A4, A5, A6 /
     K / A, B, C /
 table CAPACITY(i,j,k) maximum capacity
          A  B  C
MAD.A1    1  0  3
MAD.A2    2  1  2

 table CAPACITY(i,j,k) maximum capacity
          A1.A  A1.B  A1.C  A2.A  A2.B
MAD         1     0     3     6     8
BCN         2     1     2     2     4
```

Interesting GAMS features

- Profile, ProfileTol
- Threads
- GRID computing
  - Use of **multiple cores** of a computer
- GUSS (Gather-Update-Solve-Scatter)
  - Use of **sensitivity analysis** for solving many similar problems

- You can **launch several GAMS processes simultaneously,** being careful with conflicting filenames

Andrés Ramos

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

Andres.Ramos@comillas.edu