UNIVERSIDAD PONTIFICIA
ICAI ICADE
COMILLAS
MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
DEPARTAMENTO DE ORGANIZACIÓN INDUSTRIAL

Lenguaje GAMS

José María Ferrer Caja
Universidad Pontificia Comillas

Alternativas para desarrollar modelos

- Lenguajes de programación de propósito general
  - ✓ C, C++, Java, Visual Basic, FORTRAN
- Lenguajes o entornos de cálculo numérico o simbólico
  - ✓ Hojas de cálculo, MATLAB, Mathematica
- Lenguajes algebraicos de modelado
  - ✓ GAMS, OPL Studio, AMPL, AIMMS, XPRESS-MP, MPL, Zimpl

Lenguajes de propósito general

- Ventajas
  - ✓ Versatilidad para crear modelos de gran complejidad y/o tamaño
  - ✓ Ejecución frecuente
  - ✓ Uso de algoritmos específicos de optimización
- Inconvenientes
  - ✓ Dificultad en la programación
  - ✓ Mantenimiento costoso del modelo

Lenguajes de cálculo numérico o simbólico

- Ventajas
  - ✓ Facilidad de manejo de los optimizadores existentes
  - ✓ Familiaridad con el entorno
  - ✓ Visualización cómoda de los resultados
  - ✓ Buena alternativa para problemas pequeños
- Inconvenientes
  - ✓ No inducen una buena práctica de programación
  - ✓ No permiten modelar problemas complejos o de gran tamaño
  - ✓ Presentan dificultades en el desarrollo, verificación, validación, actualización y documentación de los modelos

Lenguajes algebraicos de modelado

- Ventajas
  - ✓ Formulación compacta de modelos grandes y complejos
  - ✓ Estructuran buenos hábitos de modelado
  - ✓ Separan datos de estructura matemática del modelo
  - ✓ Modelo independiente de optimizadores
  - ✓ Documentación simultánea al modelo
  - ✓ Mantenimiento y reformulación cómodos
  - ✓ Portabilidad entre plataformas y sistemas operativos
- Inconvenientes
  - ✓ No recomendables para uso esporádico con problemas de pequeño tamaño
  - ✓ No son adecuados para resolución directa de problemas de muy gran tamaño

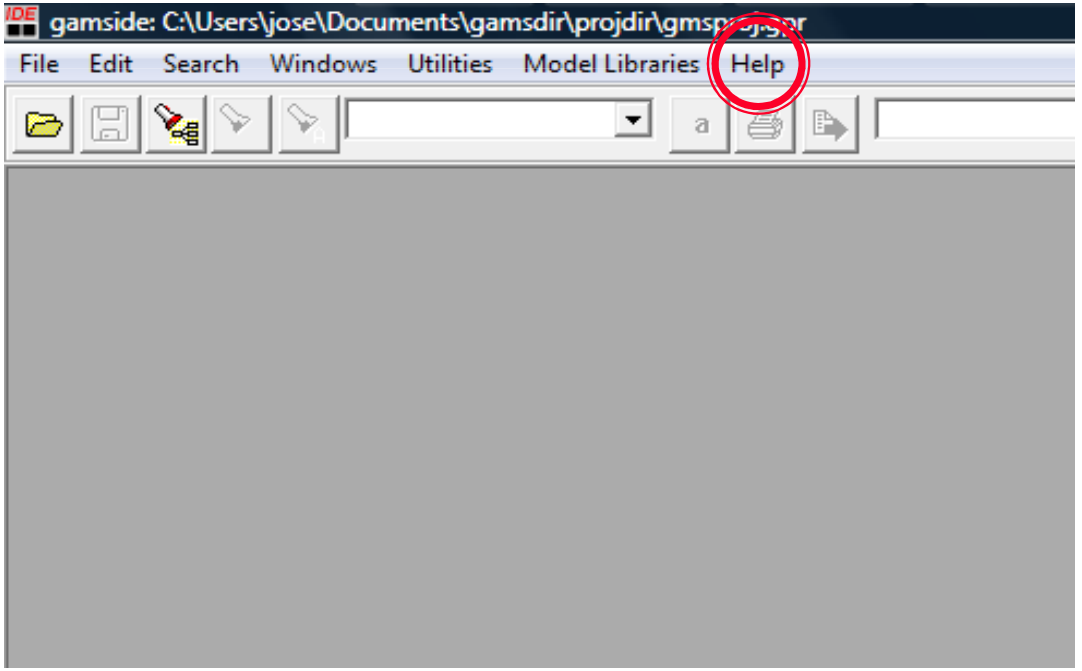
GAMS (General Algebraic Modeling System)

- Lenguaje algebraico de modelado
- Creado en 1987 en EEUU
- Más de 10000 usuarios en 100 países
- Compatible con multitud de optimizadores

- Descarga e instalación de la versión estudiante
  - ✓ Enlace en el portal de recursos
  - ✓ Instalada en los ordenadores de la escuela
  - ✓ Permite resolver problemas de tamaño pequeño
  - ✓ Para resolver un problema mayor se puede enviar al enlace que aparece en el portal de recursos

Manuales

- Ejecutar la aplicación GAMS
- Para un manual de GAMS: Seleccionar en el menú superior Help → GAMS Users Guide
- Para manuales de los optimizadores disponibles: Seleccionar en el menú superior Help → Solver Manual



Formato general de las instrucciones GAMS

- ✓ Para incluir un comentario se comienza la línea con *. Si el comentario ocupa varias líneas se puede intercalar entre las instrucciones $Ontext y $Offtext
- ✓ No se distingue entre mayúsculas y minúsculas
- ✓ El paréntesis (), el corchete [] o la llave {} se pueden utilizar indistintamente para separar niveles
- ✓ Las instrucciones han de acabar con ; (puede omitirse si la siguiente palabra es reservada)
- ✓ Las palabras reservadas del lenguaje GAMS tienen un uso específico, el código las reconoce y las resalta (en azul). No se pueden utilizar fuera de su uso

Palabras reservadas

```
abort
acronym
acronyms
alias
all
and
assign
binary
card
diag
display
else
eps
eq
equation
equations
file
files
for
free
ge
gt
if
inf
integer
le
loop
lt
maximizing
minimizing
model
models
na
ne
negative
no
not
option
options
or
ord
parameter
parameters
positive
prod
putpage
puttl
repeat
sameas
scalar
scalars
semicont
semiint
set
sets
smax
smin
solve
sos1
sos2
sum
system
table
then
until
using
variable
variables
while
xor
yes
```

Estructura general de un modelo

- Declaración de conjuntos. Asignación de valores
- Inclusión y manipulación de datos de entrada y parámetros auxiliares.
- Variables
- Ecuaciones
- Modelo
- Acotación e inicialización de variables
- Resolución del problema de optimización
- Presentación de resultados

Bloques de un modelo en GAMS

- Obligatorios
  - ✓ VARIABLES
  - ✓ EQUATIONS
  - ✓ MODEL
  - ✓ SOLVE

- Opcionales
  - ✓ SETS: (ALIAS)
  - ✓ DATA: SCALARS, PARAMETERS, TABLE

Bloque VARIABLES

- Se recomienda el uso de comentarios explicativos
- La función objetivo se declara como variable (libre)
- Tipos
  - ✓ `FREE` (por omisión) $-\infty$ a $+\infty$
  - ✓ `POSITIVE` 0 a $+\infty$
  - ✓ `NEGATIVE` $-\infty$ a 0
  - ✓ `BINARY` 0 ó 1
  - ✓ `INTEGER` 0 a 100
- Sufijos
  - ✓ `.LO` cota inferior
  - ✓ `.UP` cota superior
  - ✓ `.L` valor inicial antes y valor óptimo después
  - ✓ `.M` valor marginal (coste reducido)
  - ✓ `.FX` fija una variable a un valor

Bloque EQUATIONS

- Se asigna un nombre a cada tipo de ecuación
- Se recomienda el uso de comentarios explicativos
- Tipos
  - ✓ `=E=` $=$
  - ✓ `=L=` $\leq$
  - ✓ `=G=` $\geq$
- Sufijos
  - ✓ `.LO` cota inferior
  - ✓ `.UP` cota superior
  - ✓ `.L` valor inicial antes y valor óptimo después
  - ✓ `.M` valor marginal (variable dual o precio en la sombra) .
  - ✓ `.FX` fija una variable a un valor

Bloques MODEL y SOLVE

- Se pueden definir varios modelos y resolverlos simultáneamente
- `MODEL` nombre_modelo1 / nombre_ecuaciones /

  `MODEL` nombre_modelo2 / nombre_ecuaciones /

  …

- `SOLVE` nombre_modelo1 `USING` tipo_problema

  `MINIMIZING (MAXIMIZING)` variable_objetivo

  …

Bloque SETS

- Se utiliza para introducir conjuntos y subconjuntos de índices
  - ✓ `SETS`

    índice1 comentario / elementos del conjunto1 /

    índice2 comentario / elementos del conjunto2 /

    …
- En los conjuntos numéricos se puede usar `*` como puntos suspensivos
- Se utiliza `ALIAS(i,j,…)` para crear copias del índice `i` definido con anterioridad

Entrada de datos

- Para parámetros unidimensionales
  - ✓ `PARAMETER`

    nombre (índice) comentario / elemento 1 valor1, elemento 2 valor2,…/

    …
  - ✓ Se pueden definir parámetros mediante fórmulas
- Para parámetros bidimensionales
  - ✓ `TABLE`

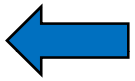
    nombre (índices) comentario

	j_1	j_2	…
i_1	$valor_{11}$	$valor_{12}$	…
i_2	$Valor_{21}$	$Valor_{22}$	…
…	…	…	…
- Se pueden importar datos de un fichero externo mediante

  `$include` nombre_del_fichero

Ejemplo de parámetro tridimensional

```
SETS i / MAD, BCN /
     j / A1, A2, A3, A4, A5, A6 /
     k / A, B, C /
TABLE CAPACIDAD(i,j,k) capacidad máxima
          A   B   C
MAD.A1    1   0   3
MAD.A2    2   1   2
BCN.A1    4   3   3
BCN.A2    0   2   2
```



2 alternativas

```
TABLE CAPACIDAD(i,j,k) capacidad máxima
        A1.A  A1.B  A1.C  A2.A  A2.B  A2.C
MAD        1     0     3     2     1     2
BCN        4     3     3     0     2     2
```

Funciones y operadores

- Elementales: `+`, `-`, `*` , `/`, `**` ó `POWER(x,n)`
- `ORD`, `CARD`
  - ✓ Ordinal y cardinal de un conjunto
- `SUM`, `PROD`, `SMAX`, `SMIN`
  - ✓ Con índices
- Otras funciones: `ABS`, `SIN`, `COS`, `FLOOR`, `EXP`, `LOG`, `LOG10`, `MAX`, `MIN`, `MOD`, `SIGN`, `SQRT`...

- Operadores lógicos: `NOT`, `AND`, `OR`, `XOR`

- Operadores relacionales: `LT`, `GT`, `EQ`, `NE`, `LE`, `GE`

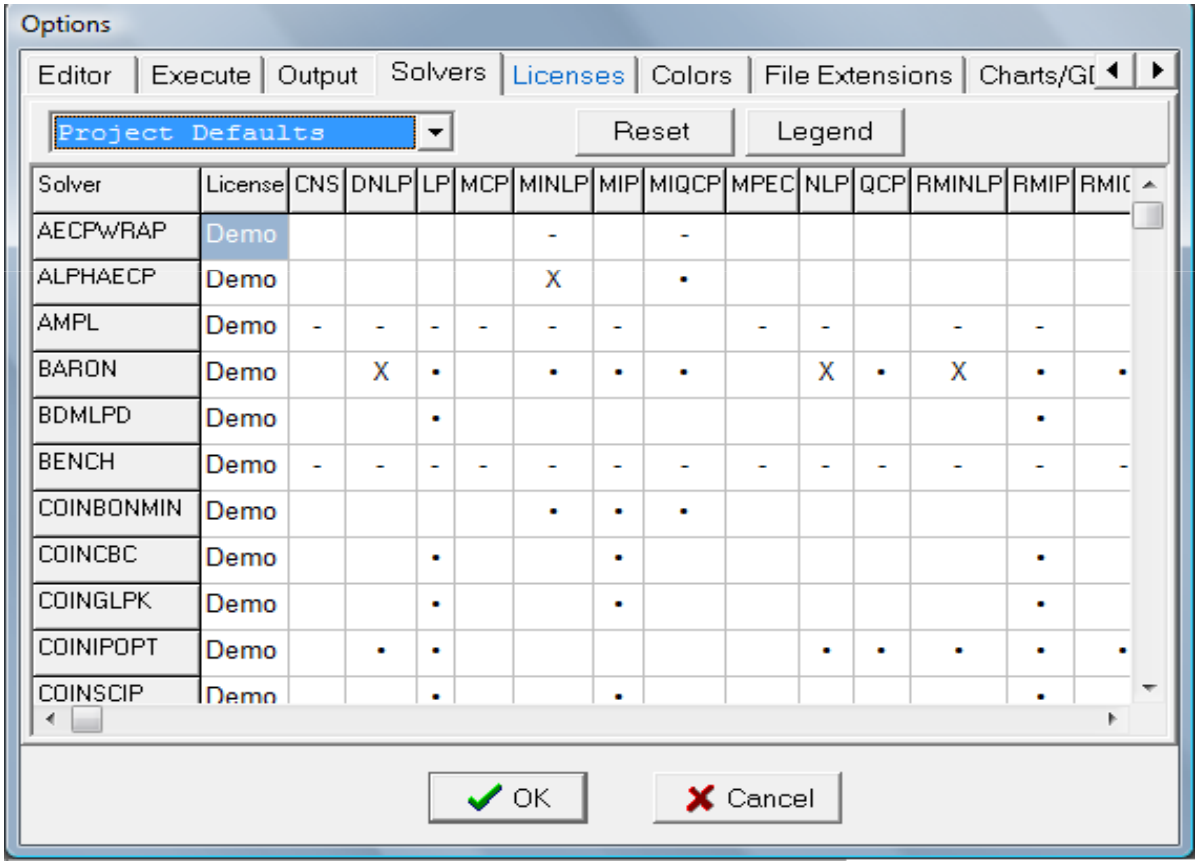
Tipos de problemas y optimizadores

- LP, RMIP (programación lineal): BDMLP, CPLEX…
- MIP (programación lineal entera mixta): CPLEX, OSL, XA, XPRESS…
- NLP (programación no lineal): CONOPT, MINOS, SNOPT, PATHNLP, LGO, MOSEK…
- DNLP (programación no lineal con derivadas no continuas): CONOPT, MINOS, SNOPT, BARON, LGO, OQNLP, MOSEK…
- MINLP (programación no lineal entera mixta): DICOPT, SBB, BARON, OQNLP…
- SP (programación estocástica): DECIS, OSLSE…
- MCP (problema mixto complementario): MILES, PATH, NLPEC…
- MPEC (programación matemática con restricciones de equilibrio): NLPEC…
- CNS (sistemas no lineales restringidos): CONOPT, PATH…

…

Elección de los optimizadores

- En menú inicio:
  - ✓ File → Options → Solvers
  - ✓ Pinchar en el optimizador deseado



Compilación y resolución

- Para compilar sólo
  - ✓ File → Compile
- Para compilar y resolver
  - ✓ File → Run o pinchando en el botón directo o F9
- Al compilar se crean 2 nuevos archivos
  - ✓ nombre_fichero.gms → Archivo del código creado en el editor
  - ✓ nombre_fichero.lst → Archivo de resultados
  - ✓ nombre_fichero.log → Archivo del registro del proceso

Ejemplo de transporte: Código (1)

```
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
```

Ejemplo de transporte: Código (2)

```
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
```

Ejemplo de transporte: Resultados (1)

```
MODEL STATISTICS

BLOCKS OF EQUATIONS           3     SINGLE EQUATIONS            6
BLOCKS OF VARIABLES           2     SINGLE VARIABLES            7
NON ZERO ELEMENTS            19


GENERATION TIME      =        1.264 SECONDS      4 Mb  WIN228-228 Jul 26, 2008


EXECUTION TIME       =        1.264 SECONDS      4 Mb  WIN228-228 Jul 26, 2008
GAMS Rev 228  x86/MS Windows                          10/23/08 12:07:12 Page 5
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE TRANSPORTE Using LP From line 34


               S O L V E      S U M M A R Y

     MODEL   TRANSPORTE          OBJECTIVE  CT
     TYPE    LP                  DIRECTION  MINIMIZE
     SOLVER  CPLEX               FROM LINE  34

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE               93.5000

 RESOURCE USAGE, LIMIT          0.078      1000.000
 ITERATION COUNT, LIMIT         5         10000

ILOG CPLEX       Aug  1, 2008 22.8.1 WIN 5924.6015 VIS x86/MS Windows
Cplex 11.1.1, GAMS Link 34
```

Ejemplo de transporte: Resultados (2)

```
---- EQU DEMANDA  satisfacción demanda de cada mercado j [cajas]

             LOWER      LEVEL      UPPER     MARGINAL

MADRID       400.000    400.000      +INF       0.050
BARCELONA    450.000    450.000      +INF       0.150
VALENCIA     150.000    150.000      +INF       0.110

---- VAR X  cajas transportadas entre fábrica i y mercado j [cajas]

                       LOWER      LEVEL      UPPER     MARGINAL

VIGO      .MADRID          .          .        +INF       0.040
VIGO      .BARCELONA       .        350.000    +INF         .
VIGO      .VALENCIA        .          .        +INF       0.010
ALGECIRAS.MADRID           .        400.000    +INF         .
ALGECIRAS.BARCELONA        .        100.000    +INF         .
ALGECIRAS.VALENCIA         .        150.000    +INF         .

                       LOWER      LEVEL      UPPER     MARGINAL

---- VAR CT             -INF       93.500     +INF         .

  CT  coste de transporte                                 [€]


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
```

Secuenciación de trabajos: Planteamiento

- Hay que realizar 5 trabajos en una máquina (en cualquier orden). El tiempo de ejecución de cada trabajo es

TR1	TR2	TR3	TR4	TR5
15	13	14	12	16

- El tiempo de ajuste de la máquina para pasar de ejecutar el trabajo (fila) a ejecutar el trabajo (columna) aparece en la siguiente tabla

	TR1	TR2	TR3	TR4	TR5
TR1		2	5	1	6
TR2	3		4	2	5
TR3	4	2		3	4
TR4	5	3	6		5
TR5	4	4	4	3	

- Determinar cómo secuenciar los trabajos para que el tiempo empleado sea mínimo. Considerar que la secuencia se ha de repetir indefinidamente

Secuenciación de trabajos: Código (1)

```
$TITLE Secuenciación de órdenes de trabajo

SETS
   I trabajos que se van a ejecutar / TR1 * TR5 /

ALIAS (i,j)

TABLE C(i,j) tiempo de ajuste para pasar del trabajo i al trabajo j
         TR1    TR2    TR3    TR4    TR5
TR1              2      5      1      6
TR2      3              4      2      5
TR3      4      2              3      4
TR4      5      3      6              5
TR5      4      4      4      3

* Es indiferente el tiempo de ejecución de los trabajos

VARIABLES
   X(i,j) paso del trabajo i al trabajo j
   TT     tiempo total en completar los trabajos

BINARY VARIABLE X
```

Secuenciación de trabajos: Código (2)

```
EQUATIONS
   TIEMPO        tiempo total de trabajo
   ANTERIOR(i)   de cada trabajo se parte una vez
   POSTERIOR(j)  a cada trabajo se llega una vez
   PAREJAS(i,j)  suma de los trabajos por parejas ;

TIEMPO        .. TT =E= SUM[(i,j) $(NOT SAMEAS(i,j)), C(i,j)*X(i,j)] ;
ANTERIOR(i)   .. SUM[j $(NOT SAMEAS(i,j)), X(i,j)] =E= 1 ;
POSTERIOR(j)  .. SUM[i $(NOT SAMEAS(i,j)), X(i,j)] =E= 1 ;
PAREJAS(i,j) $(ORD(i) < ORD(j)) .. X(i,j) + X(j,i) =L= 1 ;

* El segundo modelo impide subciclos de parejas de trabajos

MODEL AJUSTE1 / TIEMPO, ANTERIOR, POSTERIOR          /
MODEL AJUSTE2 / TIEMPO, ANTERIOR, POSTERIOR, PAREJAS /

SOLVE AJUSTE1 USING MIP MINIMIZING TT
SOLVE AJUSTE2 USING MIP MINIMIZING TT
```

Secuenciación de trabajos: Resultados (1)

```
---- VAR X  paso del trabajo i al trabajo j

           LOWER      LEVEL      UPPER    MARGINAL

TR1.TR2      .          .         1.000      2.000
TR1.TR3      .          .         1.000      5.000
TR1.TR4      .        1.000       1.000      1.000
TR1.TR5      .          .         1.000      6.000
TR2.TR1      .        1.000       1.000      3.000
TR2.TR3      .          .         1.000      4.000
TR2.TR4      .          .         1.000      2.000
TR2.TR5      .          .         1.000      5.000
TR3.TR1      .          .         1.000      4.000
TR3.TR2      .          .         1.000      2.000
TR3.TR4      .          .         1.000      3.000
TR3.TR5      .        1.000       1.000      4.000
TR4.TR1      .          .         1.000      5.000
TR4.TR2      .        1.000       1.000      3.000
TR4.TR3      .          .         1.000      6.000
TR4.TR5      .          .         1.000      5.000
TR5.TR1      .          .         1.000      4.000
TR5.TR2      .          .         1.000      4.000
TR5.TR3      .        1.000       1.000      4.000
TR5.TR4      .          .         1.000      3.000

                      LOWER      LEVEL      UPPER    MARGINAL

---- VAR TT             -INF     15.000      +INF        .

  TT  tiempo total en completar los trabajos
```

El primer modelo no ofrece una solución válida: se forman subciclos

Secuenciación de trabajos: Resultados (2)

```
---- VAR X  paso del trabajo i al trabajo j

           LOWER      LEVEL      UPPER    MARGINAL

TR1.TR2      .          .         1.000      2.000
TR1.TR3      .          .         1.000      5.000
TR1.TR4      .        1.000       1.000      1.000
TR1.TR5      .          .         1.000      6.000
TR2.TR1      .        1.000       1.000      3.000
TR2.TR3      .          .         1.000      4.000
TR2.TR4      .          .         1.000      2.000
TR2.TR5      .          .         1.000      5.000
TR3.TR1      .          .         1.000      4.000
TR3.TR2      .        1.000       1.000      2.000
TR3.TR4      .          .         1.000      3.000
TR3.TR5      .          .         1.000      4.000
TR4.TR1      .          .         1.000      5.000
TR4.TR2      .          .         1.000      3.000
TR4.TR3      .          .         1.000      6.000
TR4.TR5      .        1.000       1.000      5.000
TR5.TR1      .          .         1.000      4.000
TR5.TR2      .          .         1.000      4.000
TR5.TR3      .        1.000       1.000      4.000
TR5.TR4      .          .         1.000      3.000

                      LOWER      LEVEL      UPPER    MARGINAL

---- VAR TT            -INF     15.000       +INF        .
```

La solución óptima es la secuencia:
TR1, TR4,TR5, TR3, TR2

Tiempo en completar un ciclo:
15 + 68 = 83