

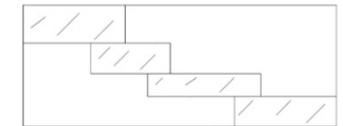
TÉCNICAS DE DESCOMPOSICIÓN EN PROGRAMACIÓN MATEMÁTICA

Dr. Víctor M. Albornoz
Departamento de Industrias.
Campus Santiago Vitacura, Chile.
Universidad Técnica Federico Santa María

Instituto de Computación, Facultad de Ingeniería, UdelaR.
Montevideo, lunes 12 de Mayo de 2025

CONTENIDO

1. Introducción a los Métodos de Descomposición.
- 2. Formulación y resolución de modelos en AMPL.**
3. Método de Benders.
4. Generación de Columnas.
5. Método de Dantzig & Wolfe.
6. Conclusiones, Extensiones y palabras finales.



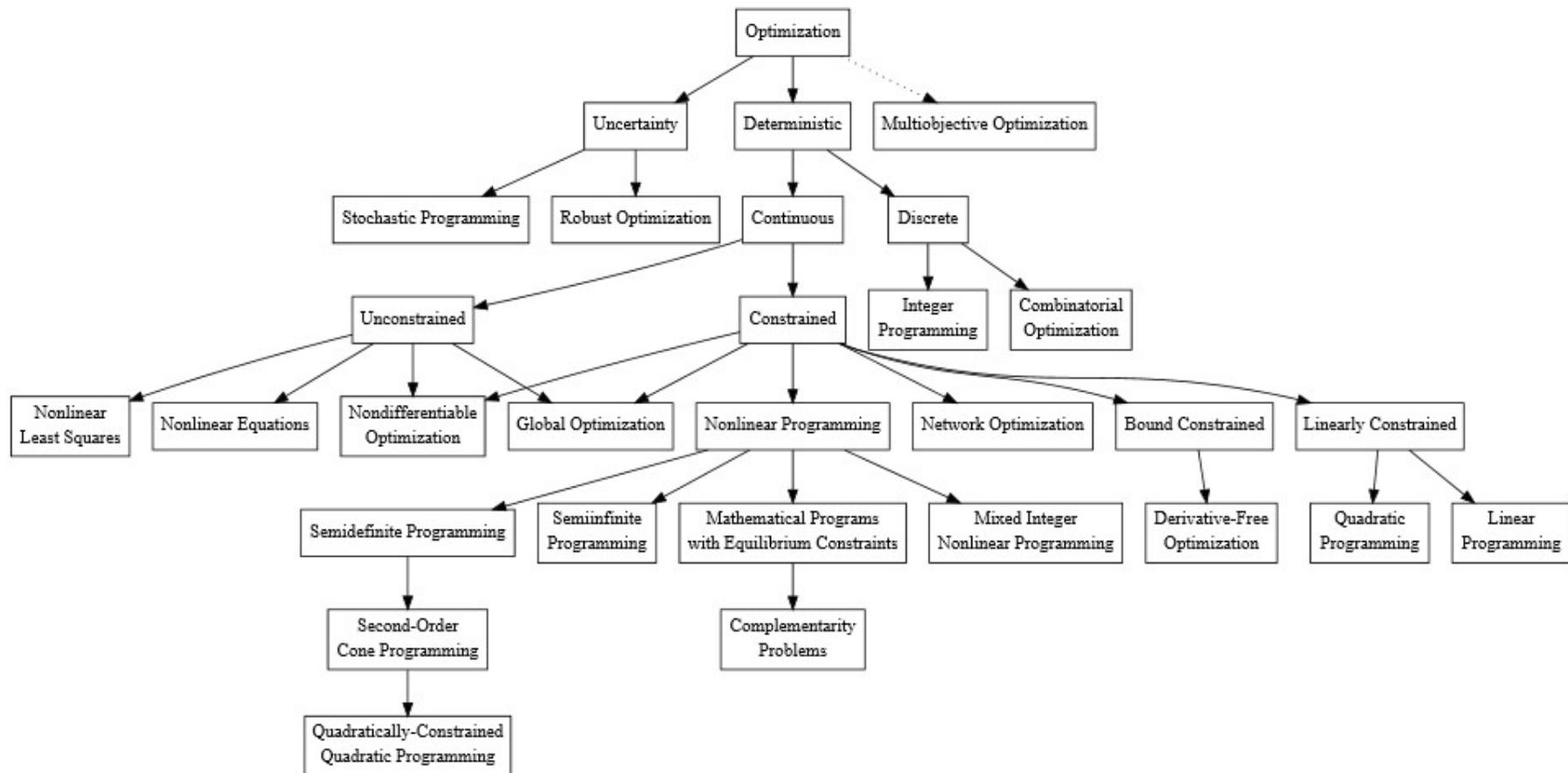
2.- Formulación y resolución de modelos en AMPL

En apoyo a la toma de decisiones en problemas de naturaleza real, la Investigación de Operaciones contempla metodologías como los *modelos de optimización* para encontrar la mejor solución a un problema o simplemente generar buenas decisiones admisibles.

OPERATIONS RESEARCH: THE SCIENCE OF BETTER

TIME-STARVED EXECUTIVES ARE MAKING BOLDER DECISIONS WITH LESS RISK AND BETTER OUTCOMES. THEIR SECRET: OPERATIONS RESEARCH.

Los *modelos de optimización* son muy variados dada la naturaleza de las decisiones y el problema abordado.



Al formular un modelo de optimización uno comúnmente expresa de manera algebraica la función objetivo y las distintas ecuaciones e inecuaciones que definen las restricciones del problema en términos de sus variables de decisión.

Set

L = the set of locations

Parameters

x_i = the x-coordinate for location i , $\forall i \in L$

y_i = the y-coordinate for the location i , $\forall i \in L$

d_i = projected demand for the next period for location i , $\forall i \in L$

s_i = number of helicopters currently assigned to location i , $\forall i \in L$

c = transportation cost per kilometer

$dist_{ij}$ = Euclidean distance between location i and location j , $\forall i \in L \forall j \in L$

$$dist_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

Variables

z_{ij} = number of helicopters to be moved from location i to j , $\forall i \in L \forall j \in L$

Objective Function

Minimize $\sum_{(i,j) \in L \times L} dist_{ij} * c * z_{ij}$

Constraints

Flow balance constraint for each location i in set L

$$\sum_{j \in L} z_{ji} + s_i = d_i + \sum_{j \in L} z_{ij}, \forall i \in L$$

Complementa lo anterior la existencia de *lenguajes de modelado algebraico* que permiten emplear la notación común para la representación de modelos y una sintaxis simple en el desarrollo de algoritmos.

Estos *lenguajes* existen desde fines de los años 70 y fueron creados considerando inicialmente problemas de programación lineal. En particular, el desarrollo de AMPL se inicia en 1985.

Algunos programas de modelado algebraico son:

AIMMS

AMPL

GAMS

JuMP

LINGO

MPL

OPL Studio

PLAM

Pyomo



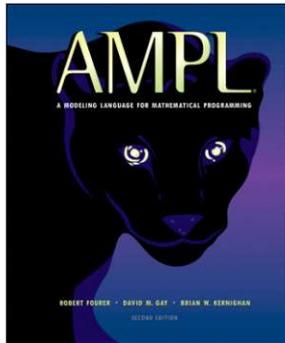
Estos software deben ser utilizados conjuntamente con un solver de acuerdo a la naturaleza del modelo a resolver, por ejemplo: **cplex, gurobi, knitro, minos y xpress.**

Sin embargo, en numerosas situaciones estos solvers pueden ser insuficientes ante problemas complejos y de gran tamaño.

Descarga versión académica de **AMPL** está disponible en la página del curso en **EVA**.

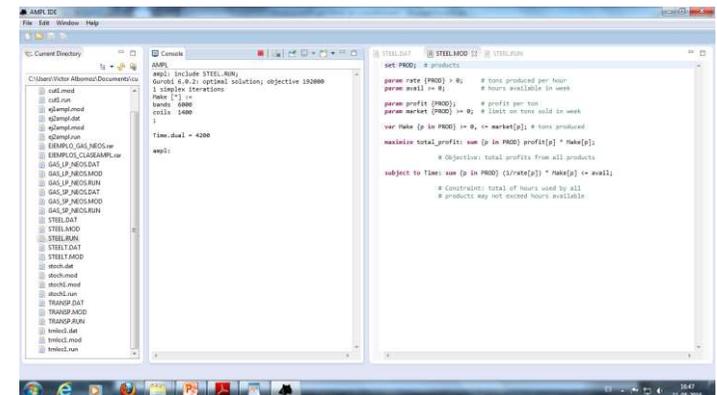
Manual [AMPL](#)

<https://ampl.com/resources/books/ampl-book/>



AMPL permite representar modelos de optimización en términos algebraicos expresados con la ayuda de conjuntos de índices y sus principales operaciones.

Toda la información contempla diferentes archivos con las extensiones: **.mod** (para el modelo), **.dat** (para los datos del modelo) y **.run** (para la ejecución del modelo).



En lo concerniente a la representación de un modelo en AMPL, este se declara en un archivo .mod, representando el mismo con los siguientes elementos y comandos:

Conjuntos de índices (set),

Parámetros (param),

Variables de decisión (var),

Función objetivo (minimize/maximize) y

Restricciones (subject to).

Ejemplo 1. Modelo de producción multi-producto:

Conjunto e índices.

P = conjunto de productos, con subíndice $p \in P$

Parámetros.

u_p = beneficio por tonelada del producto $p \in P$.

r_p = ton producidas del producto $p \in P$ por hora.

av = horas disponibles de producción.

d_p = demanda máxima del producto $p \in P$.

Variable de decisión.

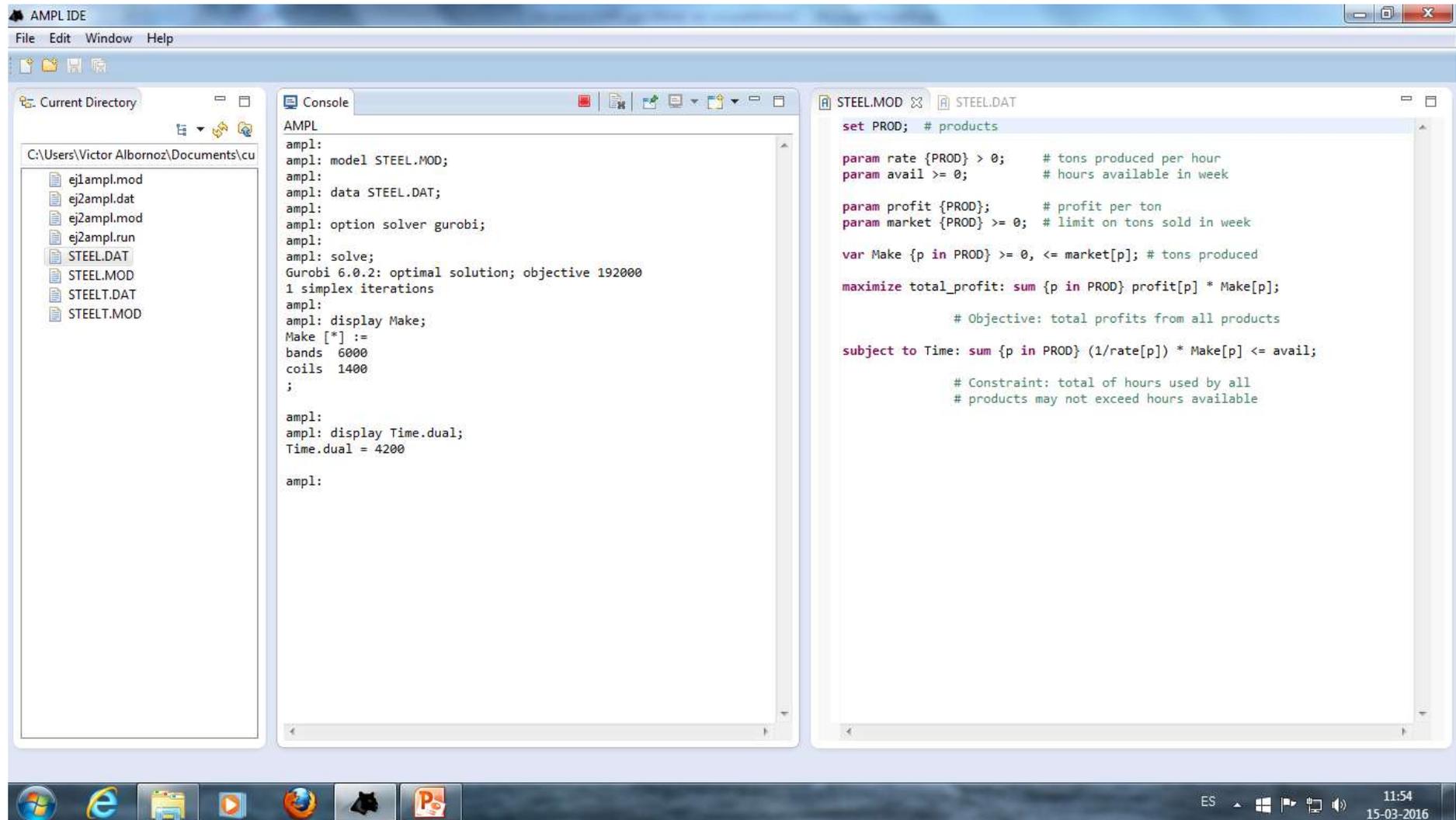
X_p = toneladas elaboradas del producto $p \in P$

$$\text{Max } \sum_{p \in P} u_p X_p$$

$$\text{s.a. } \sum_{p \in P} (1/r_p) X_p \leq av$$

$$0 \leq X_p \leq d_p \quad p \in P$$

El modelo de producción en AMPL puede revisarse en los archivos STEEL.MOD y STEEL.DAT:



The screenshot shows the AMPL IDE interface with three main panes:

- Current Directory:** Shows the file structure in `C:\Users\Victor Albornoz\Documents\cu`, including `ej1ampl.mod`, `ej2ampl.dat`, `ej2ampl.mod`, `ej2ampl.run`, `STEEL.DAT`, `STEEL.MOD`, `STEELT.DAT`, and `STEELT.MOD`.
- Console:** Displays the execution output:

```
AMPL
ampl:
ampl: model STEEL.MOD;
ampl:
ampl: data STEEL.DAT;
ampl:
ampl: option solver gurobi;
ampl:
ampl: solve;
Gurobi 6.0.2: optimal solution; objective 192000
1 simplex iterations
ampl:
ampl: display Make;
Make [*] :=
bands 6000
coils 1400
;

ampl:
ampl: display Time.dual;
Time.dual = 4200

ampl:
```
- STEEL.MOD and STEEL.DAT:** Shows the model code:

```
set PROD; # products

param rate {PROD} > 0; # tons produced per hour
param avail >= 0; # hours available in week

param profit {PROD}; # profit per ton
param market {PROD} >= 0; # limit on tons sold in week

var Make {p in PROD} >= 0, <= market[p]; # tons produced

maximize total_profit: sum {p in PROD} profit[p] * Make[p];

# Objective: total profits from all products

subject to Time: sum {p in PROD} (1/rate[p]) * Make[p] <= avail;

# Constraint: total of hours used by all
# products may not exceed hours available
```

The Windows taskbar at the bottom shows the system tray with the time 11:54 and date 15-03-2016.

Ejemplo 2. Consideramos un problema muy relevante a nivel táctico correspondiente al de *planificación agregada de la producción*.

Este problema consiste en hallar una política óptima de producción de un determinado conjunto de productos para satisfacer demandas fluctuantes en el tiempo, de modo de minimizar costos de producción e inventario, considerando la disponibilidad de diversos recursos escasos.

Parámetros.

c_{pt} = costo unitario de producción del producto p en periodo t .

h_{pt} = costo unitario de inventario del producto p en periodo t .

u_{pt} = beneficio por unidad del producto p en periodo t .

d_{pt} = demanda máxima de unidades de producto p en t .

r_p = unidades producidas del producto p por hora.

av_t = horas disponibles de producción en periodo t .

l_{p0} = inventario inicial del producto p .

Por su parte, las *variables de decisión* del modelo corresponden a:

X_{pt} = unidades elaboradas del producto p en periodo t

I_{pt} = nivel de inventario del producto p al término de periodo t

S_{pt} = unidades vendidas del producto p en periodo t

Modelo de Producción multi-producto con múltiples periodos

$$\text{Max } \sum_p \sum_t (u_{pt} S_{pt} - c_{pt} X_{pt} - h_{pt} I_{pt})$$

s.a.

$$X_{pt} + I_{pt-1} = S_{pt} + I_{pt} \quad p \in P; t=1, \dots, T$$

$$\sum_{p \in P} (1/r_p) X_{pt} \leq av_t \quad t=1, \dots, T$$

$$0 \leq S_{pt} \leq d_{pt} \quad p \in P; t=1, \dots, T$$

$$I_{pt} \geq 0, X_{pt} \geq 0, \quad p \in P; t=1, \dots, T$$

Si discute a continuación el modelo descrito y su formulación en AMPL asociada a los archivos STEEL.MOD y STEEL.DAT:

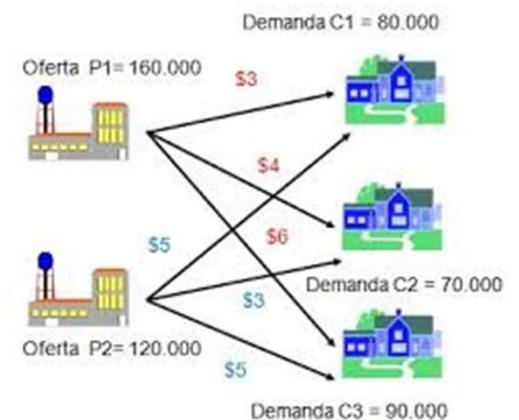
The screenshot displays the AMPL IDE interface with three main panes:

- Current Directory:** Shows the file structure for the project, including `ej1ampl.mod`, `ej2ampl.dat`, `ej2ampl.mod`, `ej2ampl.run`, `STEEL.DAT`, `STEEL.MOD`, `STEELT.DAT`, and `STEELT.MOD`.
- Console:** Shows the execution of an AMPL solve command. The output indicates an error: "Error executing 'solve' command: error processing var Make[...]: no data for set PROD". A subsequent solve attempt is successful, showing an optimal solution with an objective value of 727990 and 16 simplex iterations. The final output table is as follows:

	Make	Sell	Inv	:=
bands 0	.	.	10	
bands 1	2590	2000	600	
bands 2	8000	8500	100	
bands 3	6400	6500	0	
bands 4	6500	6500	0	
coils 0	.	.	0	
coils 1	3787	0	3787	
coils 2	0	2500	1287	
coils 3	0	1287	0	
coils 4	1050	1050	0	
- *STEELT.MOD:** Contains the AMPL model code, including parameters for products, weeks, rates, inventory, and revenue, as well as variables for production, inventory, and sales. The model is formulated to maximize total profit, subject to time and initial inventory constraints.

Ejemplo 3. *Problema de Transporte.*

Un problema muy interesante en la logística de las operaciones consiste en decidir cuántas unidades trasladar desde ciertos puntos de origen (plantas, ciudades, etc.) a ciertos puntos de destino (centros de distribución, ciudades, etc..) de modo de minimizar los costos de transporte, dada la oferta y demanda en dichos puntos.



Variables de decisión.

$T_{i,j}$: unidades transportadas desde el origen i al destino j

Modelo

$$\text{Min } \sum_i \sum_j c_{i,j} T_{i,j}$$

s.a.

$$\sum_j T_{i,j} \leq \text{oferta}_i \quad \text{para todo origen } i=1, \dots, m$$

$$\sum_i T_{i,j} = \text{demanda}_j \quad \text{para todo destino } j=1, \dots, n$$

$$T_{i,j} \geq 0 \quad i=1, \dots, m; j=1, \dots, n$$

Se presenta un modelo correspondiente a un problema de transporte como el del ejemplo asociado a los archivos **TRANSP.MOD** y **TRANSP.DAT**:

The screenshot shows the AMPL IDE interface with three main panes:

- Current Directory:** Lists files in the current directory, including `ej1ampl.mod`, `ej2ampl.dat`, `ej2ampl.mod`, `ej2ampl.run`, `STEEL.DAT`, `STEEL.MOD`, `STEELT.DAT`, `STEELT.MOD`, `TRANSP.DAT`, `TRANSP.MOD`, and `TRANSP.RUN`.
- Console:** Displays the AMPL execution output:

```
AMPL
ampl: include TRANSP.RUN;
CPLEX 12.6.1.0: sensitivity
display=2
No LP presolve or aggregator reductions.

Iteration   Dual Objective   In Variable   Out
  1         37700.000000   x18
  2         61100.000000   x9
  3         63600.000000   x16
  4         75300.000000   x10
  5         94000.000000   x19
  6        101000.000000   x5
  7        110000.000000   x7
  8        191400.000000   x6
  9        195000.000000   x15
 10        195800.000000   x11
 11        195800.000000   x21
 12        196200.000000   x14

CPLEX 12.6.1.0: optimal solution; objective 196200
12 dual simplex iterations (0 in phase I)

suffix up OUT;
suffix down OUT;
suffix current OUT;
Trans [*,*] (tr)
:   CLEV  GARY  PITT  :=
DET 1200   0    0
FRA  0    0   900
FRE  0   1100  0
LAF 400   300  300
LAN 600   0    0
STL  0    0  1700
WIN 400   0    0
;

:   Trans.down Trans.current Trans.up   :=
CLEV_DET -1e+20    9    11
```
- TRANSP.DAT:** Shows the data input for the model:

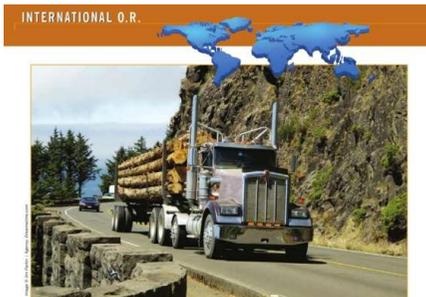
```
param: ORIG: supply := # defines set "ORIG" and param "supply"
        GARY 1400
        CLEV 2600
        PITT 2900 ;

param: DEST: demand := # defines "DEST" and "demand"
        FRA  900
        DET 1200
        LAN  600
        WIN  400
        STL 1700
        FRE 1100
        LAF 1000 ;

param cost:
        FRA  DET  LAN  WIN  STL  FRE  LAF :=
GARY  39  14  11  14  16  82  8
CLEV  27   9  12   9  26  95  17
PITT  24  14  17  13  28  99  20 ;
```

Ejemplo 4. Problema de producción y transporte.

Dado un conjunto de múltiples plantas (orígenes) donde se elaboran múltiples productos, el problema consiste en definir niveles óptimos de producción y despacho para satisfacer la demanda de cada cliente (destino) minimizando el costo total de producción y transporte.



Se presenta a continuación un modelo que representa el problema de producción y transporte en AMPL asociado a los archivos STEELP.MOD y STEELP.DAT.

The screenshot displays the AMPL IDE interface. On the left, a file explorer shows the directory structure, with 'STEELP.MOD' selected. The main window shows the AMPL model code, and the console window shows the solver's output.

```
AMPL
amp1: include STEELP.RUN;
gurobi 8.1.0: optimal solution; objective 1392175
10 simplex iterations
Make :=
CLEV bands      0
CLEV coils     1950
CLEV plate      0
GARY bands     1125
GARY coils     1750
GARY plate      300
PITT bands      775
PITT coils     500
PITT plate      500
;

Trans [CLEV,*,*]
: bands coils plate :=
DET  0  750  0
FRA  0  0  0
FRE  0  0  0
LAF  0  500  0
LAN  0  400  0
STL  0  50  0
WIN  0  250  0

[GARY,*,*]
: bands coils plate :=
DET  0  0  0
FRA  0  0  0
FRE  225  850  100
LAF  250  0  0
LAN  0  0  0
STL  650  900  200
WIN  0  0  0

[PITT,*,*]
: bands coils plate :=
DET  300  0  100
FRA  300  500  100
FRE  0  0  0
LAF  0  0  250
LAN  100  0  0
STL  0  0  0
WIN  75  0  50
;

Time.dual [*] :=
CLEV -1300
GARY -2800
PITT  0
;

total_cost = 1392180

amp1:
```

```
set ORIG; # origins (steel mills)
set DEST; # destinations (factories)
set PROD; # products

param rate {ORIG,PROD} > 0; # tons per hour at origins
param avail {ORIG} >= 0; # hours available at origins
param demand {DEST,PROD} >= 0; # tons required at destinations

param make_cost {ORIG,PROD} >= 0; # manufacturing cost/ton
param trans_cost {ORIG,DEST,PROD} >= 0; # shipping cost/ton

var Make {ORIG,PROD} >= 0; # tons produced at origins
var Trans {ORIG,DEST,PROD} >= 0; # tons shipped

minimize total_cost:
sum (i in ORIG, p in PROD) make_cost[i,p] * Make[i,p] +
sum (i in ORIG, j in DEST, p in PROD)
trans_cost[i,j,p] * Trans[i,j,p];

subject to Time {i in ORIG}:
sum (p in PROD) (1/rate[i,p]) * Make[i,p] <= avail[i];

subject to Supply {i in ORIG, p in PROD}:
sum (j in DEST) Trans[i,j,p] = Make[i,p];

subject to Demand {j in DEST, p in PROD}:
sum (i in ORIG) Trans[i,j,p] = demand[j,p];
```

Ejemplo 5. Problema de localización y transporte.

Asuma que se tiene un conjunto de n clientes, de los cuales el cliente j demanda d_j unidades de un producto determinado. Una compañía desea satisfacer esas demandas desde un cierto conjunto de bodegas elegidas de entre m potenciales lugares donde se instalarán.

Denotamos por c_i los costos fijos asociados a la instalación de la planta i , y t_{ij} el costo de transporte de una unidad desde la bodega i al cliente j .

El problema consiste en decidir cuáles plantas habilitar de modo de satisfacer las demandas (estimadas) al mínimo costo.

Variables de decisión:

y_i = variable binaria que toma el valor 1 si se elabora en la planta i y 0 en caso contrario, con $i=1, \dots, m$.

x_{ij} = el número de unidades elaboradas en la planta i para satisfacer el cliente j , con $i=1, \dots, m$ y $j=1, \dots, n$.

Función objetivo:

$$\textit{Min} \quad \sum_{i=1}^m c_i y_i \quad + \quad \sum_{i=1}^m \sum_{j=1}^n t_{ij} x_{ij}$$

Costo de
Instalación

Costo de
Transporte

Restricciones:

Demanda cliente $j=1, \dots, n$:
$$\sum_{i=1}^m x_{ij} = d_j$$

Relacionar variables transporte con las asociadas a la apertura de cada planta $i=1, \dots, m$:

$$\sum_{j=1}^n x_{ij} \leq M_i y_i$$

donde M_i es una constante suficientemente grande.

Las variables además satisfacen: $x_{ij} \geq 0$ e $y_j \in \{0, 1\}$.

Si discute a continuación un modelo en AMPL correspondiente al problema descrito de localización y transporte detallado en los archivos trnloc1.mod y trnloc1.dat.

```
AMPL
ampl: include trnloc1.run;
CPLEX 12.6.1.0: sensitivity
display=2
CPLEX 12.6.1.0: optimal integer solution within mipgap or absmipgap
108 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 304.508, relmipgap = 5.30944e-05
No basis.
Build [*] :=
 1 0 4 0 7 0 10 0 13 0 16 0 19 0 22 1 25 1
 2 0 5 0 8 0 11 0 14 0 17 1 20 1 23 0
 3 0 6 0 9 0 12 0 15 0 18 1 21 0 24 1
;

Ship [*,*]
: A3 A6 A8 A9 B2 B4 :=
 1 0 0 0 0 0 0
 2 0 0 0 0 0 0
 3 0 0 0 0 0 0
 4 0 0 0 0 0 0
 5 0 0 0 0 0 0
 6 0 0 0 0 0 0
 7 0 0 0 0 0 0
 8 0 0 0 0 0 0
 9 0 0 0 0 0 0
10 0 0 0 0 0 0
11 0 0 0 0 0 0
12 0 0 0 0 0 0
13 0 0 0 0 0 0
14 0 0 0 0 0 0
15 0 0 0 0 0 0
16 0 0 0 0 0 0
17 0 0 8810 0 0 960
18 0 0 5190 13500 0 0
19 0 0 0 0 0 0
20 0 12000 0 0 9220 0
21 0 0 0 0 0 0
```

```
# CARGA DEL MODELO Y DATOS
reset;
model trnloc1.mod;
data trnloc1.dat;

# SELECCION DEL SOLVER
option solver cplex;

solve;

display Build;

display Ship;

display Costo_Total;

# GUARDAR RESULTADOS EN UN ARCHIVO

display Build > trnloc1.sal;
display Ship > trnloc1.sal;
display Costo_Total > trnloc1.sal;
```