

Soluzione di problemi di ottimizzazione

- Problema di programmazione lineare:

$$\min z = c'x$$

$$Ax \geq b$$

$$x \geq 0$$

- Possibili ulteriori vincoli (Es.: x INTERO)

- Soluzione:

- Trovare x^* tale che

$$c'x^* \leq c'x \quad \text{per ogni } x / Ax \geq b, x \geq 0$$

Solutori di problemi di PL/PLI

- Input:
 - $n, m, (c), (A), (b)$
 - Eventuali altri vincoli (integrità variabili)
- Output:
 - $z, (x)$
 - Informazioni aggiuntive
(variabili duali, costi ridotti, ...)

Uso di Modelli Matematici

Es: Problema della dieta:

- n possibili cibi
- m sostanze nutritive
- b_i : fabbisogno giornaliero sostanza i
- c_j : costo cibo j
- a_{ij} : quantità di sostanza i per unità di cibo j

Determinare la dieta (mix di alimenti) di costo minimo, assicurando il fabbisogno giornaliero di ogni sostanza

Uso di Modelli Matematici (2)

x_j = quantità di cibo j nella dieta ($j=1, \dots, n$)

Modello matematico:

$$\min \sum_{j=1, n} c_j x_j$$

$$\sum_{j=1, n} a_{ij} x_j \geq b_i \quad i = 1, \dots, m$$

$$x_j \geq 0 \quad j = 1, \dots,$$

Dieta - esempio

$n = 5$ (cibi)

$m = 8$ (nutrienti)

$c_j = \{10, 8, 15, 6, 16\}$

$b_i = \{40, 20, 30, 10, 42, 41, 65, 17\}$

$a_{ji} = \begin{Bmatrix} 0, & 14, & 2, & 36, & 0, & 55, & 33, & 44, \\ 36, & 89, & 1, & 99, & 30, & 17, & 7, & 10, \\ 8, & 42, & 15, & 9, & 26, & 3, & 23, & 11, \\ 20, & 17, & 0, & 6, & 55, & 1, & 0, & 0, \\ 7, & 44, & 16, & 19, & 28, & 0, & 10, & 4 \end{Bmatrix}$

Soluzione 1: scrittura manuale

- Definizione di un file di input (testo) per un risolutore (es. Cplex)

```
/* file CPLEX */
```

```
MINIMIZE
```

```
z: 10 x1 + 8 x2 + 15 x3 + 6 x4 + 16 x5
```

```
SUBJECT TO
```

```
CONSTR1:          36 x2 + 8 x3 + 20 x4 + 7 x5    >= 40
```

```
CONSTR2: 14 x1 + 89 x2 + 42 x3 + 17 x4 + 44 x5  >= 20
```

```
CONSTR3: 2 x1 +      x2 + 15 x3 +              16 x5 >= 30
```

```
CONSTR4: 36 x1 + 99 x2 + 9 x3 + 6 x4 + 19 x5    >= 10
```

```
CONSTR5:          30 x2 + 26 x3 + 55 x4 + 28 x5  >= 42
```

```
CONSTR6: 55 x1 + 17 x2 + 3 x3 +      x4          >= 41
```

```
CONSTR7: 33 x1 + 7 x2 + 23 x3 +              10 x5 >= 65
```

```
CONSTR8: 44 x1 + 10 x2 + 11 x3 +              4 x5  >= 17
```

```
END
```

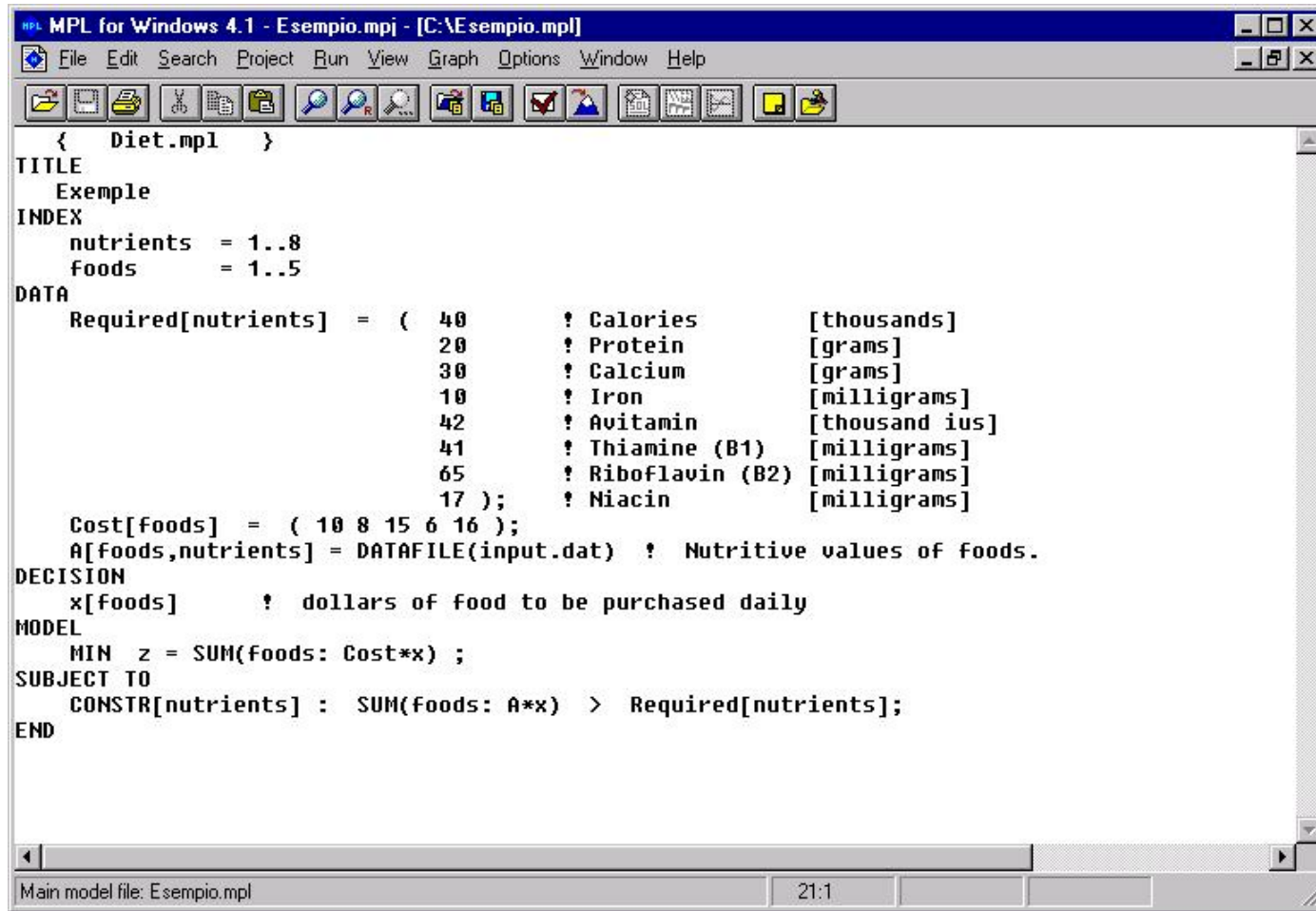
Soluzione 2: scrittura su File

- Scrittura automatica del file di input (testo)

```
void diet( n, m, a, b, c )
{
    int i, j;
    FILE *fou;

    fou = fopen("diet.lp",w);
    fprintf(fou,"MINIMIZE\n Z: ");
    for (j = 1; j <= n; j++) fprintf(fou, "+%d x%d ",c[j], j );
    fprintf(fou, " \n SUBJECT TO \n");
    for (i = 1; i <= m; i++) {
        fprintf(fou, "CONSTR%d : ",i );
        for (j = 1; j <= n; j++)
            fprintf(fou, "+%d x%d ",a[i,j],j);
        fprintf(fou, " >= %d \n",b[i] );
    }
    fprintf(fou, "END\n");
    fclose(fou);
}
```

Soluzione 3: MPL



The screenshot shows the MPL for Windows 4.1 interface. The window title is "MPL for Windows 4.1 - Esempio.mpl - [C:\Esempio.mpl]". The menu bar includes File, Edit, Search, Project, Run, View, Graph, Options, Window, and Help. The toolbar contains various icons for file operations and execution. The main text area displays the following code:

```
{ Diet.mpl }
TITLE
  Exemple
INDEX
  nutrients = 1..8
  foods     = 1..5
DATA
  Required[nutrients] = ( 40      ? Calories      [thousands]
                        20      ? Protein       [grams]
                        30      ? Calcium       [grams]
                        10      ? Iron          [milligrams]
                        42      ? Avitamin      [thousand ius]
                        41      ? Thiamine (B1) [milligrams]
                        65      ? Riboflavin (B2) [milligrams]
                        17 ); ? Niacin        [milligrams]
  Cost[foods] = ( 10 8 15 6 16 );
  A[foods,nutrients] = DATAFILE(input.dat) ? Nutritive values of foods.
DECISION
  x[foods]      ? dollars of food to be purchased daily
MODEL
  MIN z = SUM(foods: Cost*x) ;
SUBJECT TO
  CONSTR[nutrients] : SUM(foods: A*x) > Required[nutrients];
END
```

The status bar at the bottom indicates "Main model file: Esempio.mpl" and "21:1".

Soluzione 3: MPL (2)

•Modello MPL del problema

TITLE

Example

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

Required[nutrients] = (40, 20, 30, 10, 42, 41, 65, 17);

Cost[foods] = (10, 8, 15, 6, 16);

A[foods,nutrients] = (0, 14, 2, 36, 0, 55, 33, 44,
36, 89, 1, 99, 30, 17, 7, 10,
8, 42, 15, 9, 26, 3, 23, 11,
20, 17, 0, 6, 55, 1, 0, 0,
7, 44, 16, 19, 28, 0, 10, 4);

DECISION

x[foods] ! dollars of food to be purchased daily

MODEL

MIN z = SUM(foods: Cost*x) ;

SUBJECT TO

CONSTR[nutrients] : SUM(foods: A*x) > Required[nutrients]

END

Soluzione 4: MPL+DATABASE

- Modello MPL del problema integrato con una base di dati

```
TITLE
  Example
OPTIONS
  DatabaseType = Access
  DatabaseAccess = "diet.mdb"
INDEX
  nutrients := DATABASE("sostanze","sostanzaID");
  foods := DATABASE("cibi","ciboID");
DATA
  Required[nutrients] = DATABASE("sostanze", "min_req");
  Cost[foods] = DATABASE("cibi", "costo_unit");
  A[foods,nutrients] = DATABASE("valori", foods="IDfoods",
  nutrients="IDnutrients");
DECISION
  x[foods] EXPORT ACTIVITY TO DATABASE("sostanze");
MODEL
  MIN z = SUM(foods: Cost*x) ;
SUBJECT TO
  CONSTR[nutrients] : SUM(foods: A*x) > Required[nutrients]
END
```

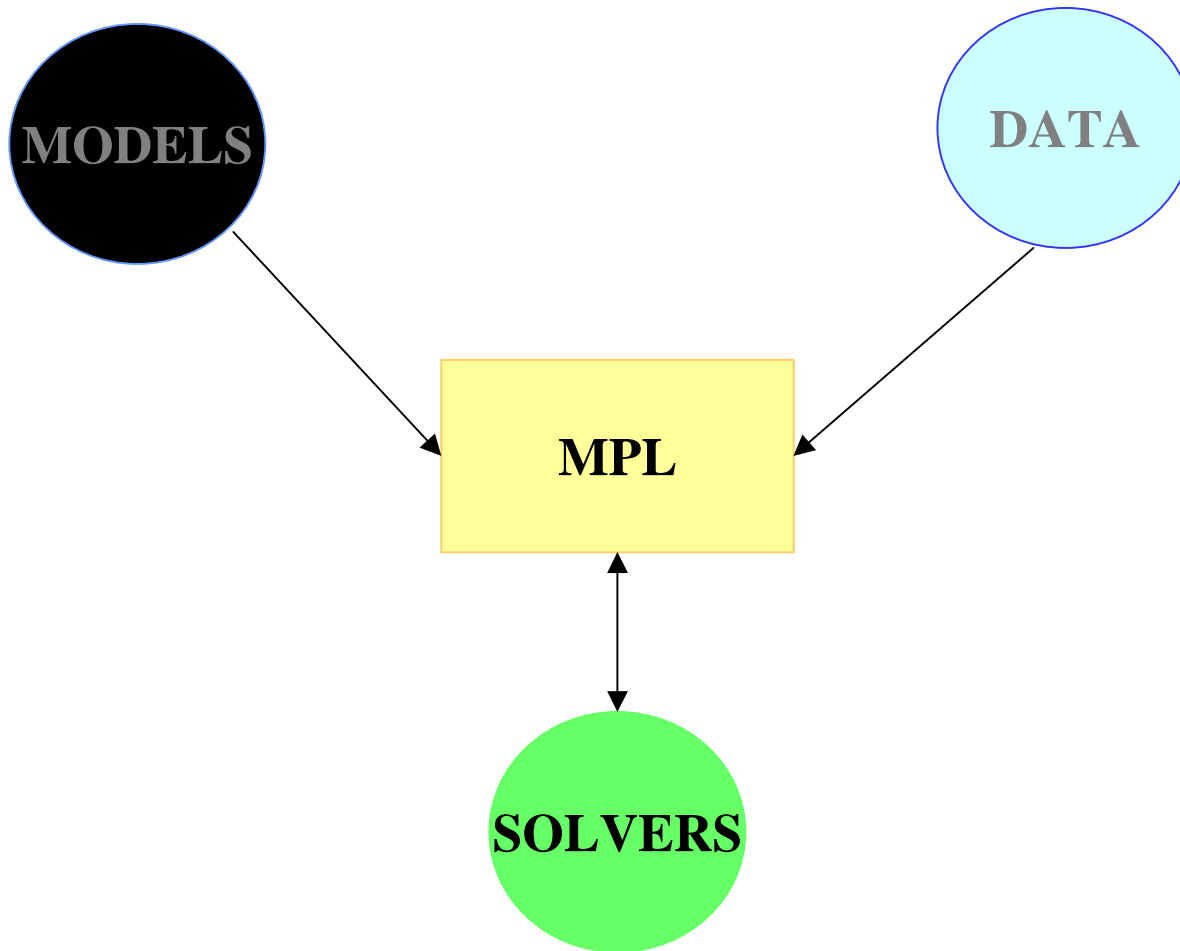
MPL: Introduzione

- MPL è un pacchetto software che permette di implementare problemi di Programmazione Lineare (PL) e Programmazione Lineare Intera (PLI) in modo chiaro, efficiente e conciso.
- E' dotato di un linguaggio ad alto livello che permette di descrivere sistemi anche molto complessi.
- Realizza la separazione tra dati e modello.

MPL: Introduzione (2)

- I modelli sono indipendenti dal risolutore impiegato e dalla piattaforma sulla quale sono eseguiti (Windows, Unix, Macintosh, OSF Motif)
- Permette di importare i dati da diverse sorgenti (file di testo, excel, database, ...)
- Nell'output sono indicate, in modo comprensibile, tutte le operazioni svolte dal risolutore.
- Si può interfacciare con strumenti di grafica

MPL Modeling System (1)



MPL: Solver

MPL si può interfacciare con i seguenti Solvers:

- CPLEX (ILOG)
- XPress-MP (DASH Associates)
- OSL
- XA
- FrontLine
- Lindo
- FortMP
- C-Whiz

MPL: Input Data

MPL consente di acquisire i dati di input:

- Direttamente nel file MPL.
- Da file di testo.
- Da database esterno (ACCESS, EXCEL, ODBC, ORACLE.....).
- Da database interno a MPL.

MPL: Modeling Language

Alcune caratteristiche:

- Utilizzo di nomi lunghi e alias.
- Importazione dati da altri programmi.
- Utilizzo di sommatorie di vettori e matrici.
- Lunghezza delle righe: 255 caratteri (i rimanenti vengono ignorati da MPL).
- Separatori: ogni statement deve terminare con ” ; “
- Commenti: “ { } ” racchiudono un blocco di commenti anche su più righe. “ ! ” commenta fino a fine riga.

Struttura file MPL

•Parte I: Dichiarazioni

TITLE	Nome del Modello (Opzionale)
INDEX	Definizione indici per insiemi
DATA	Vettori di input e costanti
DECISION	Variabili decisionali
MACRO	Definizione di macro
OPTIONS	Opzioni varie

Struttura file MPL (2)

•Parte II: Modello

MAX / MIN	Funzione obiettivo
SUBJECT TO	Vincoli del modello
BOUNDS	Upper e Lower bound
INTEGER	Variabili intere
BINARY	Variabili binarie
FREE	Variabili libere
END	Fine modello

Modello dieta

TITLE

Diet;

INDEX

Gli indici definiscono i domini del problema.

Possono essere:

- Numerici:

```
nutrients := 1..8;
```

- Nominali:

```
foods := (pasta, ham, cheese, egg, bread);
```

- Il numero di caratteri può essere limitato

```
foods := (pasta, ham, cheese, egg, bread) : 3;
```

```
genera (pas, ham, che, egg, bre);
```

INDEX (2)

- Indici circolari:
 - Se i domini rappresentano periodi di tempo è spesso necessario usare operazioni in modulo sul valore dell'indice

```
day := (mo, tu, we, th, fr, sa, su) CIRCULAR;  
month := 1..12 CIRCULAR;
```
- Si possono definire sottoinsiemi di indici, specificando:
 - l'indice di base
 - la lista di valori del sottoinsieme

```
day := (mo, tu, we, th, fr, sa, su);  
holiday[day] := (sa, su);
```

Modello dieta (2)

TITLE

Diet;

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

- In questa sezione si specificano i coefficienti utilizzati dal modello (costanti o vettori)

1. Costanti:

DATA

MaxP = 10;

MeseFerie = 3;

...

DATA (2)

2. Vettori (contenenti i dati di ingresso)

INDEX

```
nutrients := 1..8;
```

DATA

```
Required[nutrients] :=  
  (40, 20, 30, 10, 42, 41, 65, 17);
```

- Per l'output sono generati i nominativi:

```
Required1, Required2, ... Required8
```

con

```
Required1 = 40, Required2 = 20, ...
```


DATA (3)

3. Matrici (definite per righe)

INDEX

$i := 1..4;$

$j := 1..3;$

DATA

```
Cost[i, j] := (40, 20, 30,  
              13, 21, 7,  
              10, 42, 41,  
              65, 17, 13);
```

DATA (4)

Dati da una fonte esterna:

1. File di testo:

```
A[foods,nutrients]:= DATAFILE("input.dat");
```

```
                { input.dat }
```

!	Calories	Protein	Calcium	Iron	VitaminaA	Thiamine	Riboflavin	Niacin
!	(1000)	(grams)	(grams)	(MG)	(1000 IU)	(MG)	(MG)	(MG)
WheatFlour	0	14	2	36	0	55	33	44
CornMeal	36	89	1	99	30	17	7	10
EvapMilk	8	42	15	9	26	3	23	11
Margarine	20	17	0	6	55	1	0	0
Cheese	7	44	16	19	28	0	10	4

2. File Excel:

```
A[foods,nutrients]=EXCELRANGE("input.xls","foods");
```

Modello dieta (3)

TITLE

Diet;

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

Required[nutrients] = (40, 20, 30, 10, 42, 41, 65, 17);

Cost[foods] = (10, 8, 15, 6, 16);

A[foods,nutrients] = (0, 14, 2, 36, 0, 55, 33, 44,
36, 89, 1, 99, 30, 17, 7, 10,
8, 42, 15, 9, 26, 3, 23, 11,
20, 17, 0, 6, 55, 1, 0, 0,
7, 44, 16, 19, 28, 0, 10, 4);

VARIABILI DECISIONALI

DECISION

- In questa sezione sono definite le variabili del problema.
- Per definire un insieme di variabili con lo stesso nome, si indica il nome del vettore e la dimensione del vettore stesso.

```
x[foods];
```

- Si possono definire variabili multidimensionali.

```
y[processor,day];
```

VARIABILI DECISIONALI (2)

- Se alcune delle variabili decisionali sono intere bisogna indicare ciò con la parola chiave `INTEGER`

```
INTEGER VARIABLES
```

```
y[processor,day];
```

- Nel caso in cui siano binarie si usa `BINARY`

```
BINARY VARIABLES
```

```
y[processor,day];
```

- La definizione può essere fatta alla fine del file `MPL`

Modello dieta (4)

TITLE

Diet;

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

Required[nutrients] = (40, 20, 30, 10, 42, 41, 65, 17);

Cost[foods] = (10, 8, 15, 6, 16);

A[foods,nutrients] = (0, 14, 2, 36, 0, 55, 33, 44,
36, 89, 1, 99, 30, 17, 7, 10,
8, 42, 15, 9, 26, 3, 23, 11,
20, 17, 0, 6, 55, 1, 0, 0,
7, 44, 16, 19, 28, 0, 10, 4);

DECISION

x[foods];

FUNZIONE OBIETTIVO

Esprime la funzione lineare da ottimizzare.

Deve essere

- indicata all'inizio del modello MPL (parte II);
- preceduta dalla parola chiave MIN o MAX;

MAX $3x_1 + 5x_2$;

MIN $Z = \text{SUM}(\text{foods}:\text{cost}*x)$;

MINIMIZE $\text{Cost} = \text{SUM}(\text{foods}, \text{nutrients}: Y)$;

- Il nome è opzionale.

Modello dieta (5)

TITLE

Diet;

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

Required[nutrients] = (40, 20, 30, 10, 42, 41, 65, 17);

Cost[foods] = (10, 8, 15, 6, 16);

A[foods,nutrients] = (0, 14, 2, 36, 0, 55, 33, 44,
36, 89, 1, 99, 30, 17, 7, 10,
8, 42, 15, 9, 26, 3, 23, 11,
20, 17, 0, 6, 55, 1, 0, 0,
7, 44, 16, 19, 28, 0, 10, 4);

DECISION

x[foods];

MODEL

MIN z = SUM(foods: Cost*x);

VINCOLI

- I vincoli del modello vengono definiti immediatamente dopo la funzione obiettivo.
- Preceduti dalla parola chiave SUBJECT TO
- I *plain constraints* sono espressi direttamente

```
Name: X1 + X2 + X3 >= 2;
```

- I *vector constraints* sono associati ad indici.

```
CONST[nutrients]:
```

```
SUM(foods:A*x)>required[nutrients];
```

Modello dieta (6)

TITLE

Diet;

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

Required[nutrients] = (40, 20, 30, 10, 42, 41, 65, 17);

Cost[foods] = (10, 8, 15, 6, 16);

A[foods,nutrients] = (0, 14, 2, 36, 0, 55, 33, 44,
36, 89, 1, 99, 30, 17, 7, 10,
8, 42, 15, 9, 26, 3, 23, 11,
20, 17, 0, 6, 55, 1, 0, 0,
7, 44, 16, 19, 28, 0, 10, 4);

DECISION

x[foods];

MODEL

MIN z = SUM(foods: Cost*x);

SUBJECT TO

CONSTR[nutrients]: SUM(foods: A*x) > Required[nutrients];

DOMINIO VARIABILI

- E' possibile indicare nel modello un lower bound e/o un upper bound per ogni variabile.
- Si utilizza la parola chiave BOUNDS

BOUNDS

```
X1 >= MinP;
```

- Default: lower bound =0 per tutte le variabili.
- Il file MPL deve terminare con la parola END

Modello dieta (completo)

TITLE

Diet;

INDEX

nutrients := 1..8;

foods := (pasta, ham, cheese, egg, bread);

DATA

Required[nutrients] = (40, 20, 30, 10, 42, 41, 65, 17);

Cost[foods] = (10, 8, 15, 6, 16);

A[foods,nutrients] = (0, 14, 2, 36, 0, 55, 33, 44,
36, 89, 1, 99, 30, 17, 7, 10,
8, 42, 15, 9, 26, 3, 23, 11,
20, 17, 0, 6, 55, 1, 0, 0,
7, 44, 16, 19, 28, 0, 10, 4);

DECISION

x[foods];

MODEL

MIN z = SUM(foods: Cost*x);

SUBJECT TO

CONSTR[nutrients]: SUM(foods: A*x) > Required[nutrients];

END

MPL su web

- www.maximal-usa.com
- Download software/download
 - Step1: request activation code form
 - Step2: download the software (with http)
student version (300 constraints/variables, 100 integer)
 - Step3: install the software (winzip)
 - Step4: activating the Cplex license
- MPL user manual `/mplman/mpltoc.html`