Ricerca Operativa - Laboratorio

Lezione 2 - Insiemi, indicizzazione e espressioni algebriche in AMPL

Docente: Luigi De Giovanni

Dipartimento di Matematica “Tullio Levi-Civita”
Università degli Studi di Padova

luigi@math.unipd.it
https://www.math.unipd.it/∼luigi/

Corso di Laurea Magistrale in Matematica
Università degli Studi di Padova
a.a. 2019–2020

File di modello e file di dati

In generale, è opportuno che il modello sia indipendente dai dati. Quindi:

- scrivere il *modello del problema* in forma generale nel file `.mod` (dichiarazioni)
- scrivere i dati che definiscono la specifica *istanza del problema* nel file `.dat` (assegnazioni)

File di modello e file di dati

In generale, è opportuno che il modello sia indipendente dai dati. Quindi:

- scrivere il *modello del problema* in forma generale nel file `.mod` (dichiarazioni)
- scrivere i dati che definiscono la specifica *istanza del problema* nel file `.dat` (assegnazioni)

Per caricare il modello:
`model` *<PATH>\nome_file.mod*

File di modello e file di dati

In generale, è opportuno che il modello sia indipendente dai dati. Quindi:

- scrivere il *modello del problema* in forma generale nel file `.mod` (dichiarazioni)
- scrivere i dati che definiscono la specifica *istanza del problema* nel file `.dat` (assegnazioni)

Per caricare il modello:
`model` *<PATH>\nome_file.mod*

Per caricare i dati:
`data` *<PATH>\nome_file.dat*

Modello e dati: dieta

Istanza del problema

$$
\begin{aligned}
\min\ & 4x_1 + 10x_2 + 7x_3 \\
& 5x_1 + 15x_2 + 4x_3 \geq 20 \\
& 6x_1 + 10x_2 + 5x_3 \geq 30 \\
& 5x_1 + 3x_2 + 12x_3 \geq 30 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0
\end{aligned}
$$

Modello e dati: dieta

Modello del problema

- Insiemi: I (risorse) e J (richieste)
- Parametri: costi C_i, $\forall i \in I$
  quantità R_j, $\forall j \in J$
  apporto A_{ij}, $\forall i \in I, j \in J$
- Variabili: acquisti $x_i \, \forall i \in I$
- Funzione obiettivo e Vincoli

$$\min \quad \sum_{i \in I} C_i x_i$$

$$s.t.$$

$$\sum_{i \in I} A_{ij} x_i \geq R_j \qquad \forall \, j \in J$$

$$x_i \in \mathbb{R}_+ \; [\, \mathbb{Z}_+ \mid \{0,1\} \,] \qquad \forall \, i \in I$$

Istanza del problema

$$\begin{aligned} \min \; & 4x_1 + 10x_2 + 7x_3 \\ & 5x_1 + 15x_2 + 4x_3 \geq 20 \\ & 6x_1 + 10x_2 + 5x_3 \geq 30 \\ & 5x_1 + 3x_2 + 12x_3 \geq 30 \\ & x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0 \end{aligned}$$

Modello e dati: dieta

Modello del problema

- Insiemi: I (risorse) e J (richieste)
- Parametri: costi C_i, $\forall i \in I$
  quantità R_j, $\forall j \in J$
  apporto A_{ij}, $\forall i \in I, j \in J$
- Variabili: acquisti $x_i \; \forall i \in I$
- Funzione obiettivo e Vincoli

$$\min \quad \sum_{i \in I} C_i x_i$$

$$s.t.$$

$$\sum_{i \in I} A_{ij} x_i \geq R_j \qquad \forall \, j \in J$$

$$x_i \in \mathbb{R}_+ \; [\, \mathbb{Z}_+ \mid \{0,1\} \,] \qquad \forall \, i \in I$$

Istanza del problema

- $I = \{verdura, carne, frutta\}$
  $J = \{proteine, ferro, vitamine\}$
- $C = \begin{bmatrix} 4 & 10 & 7 \end{bmatrix}$
  $R = \begin{bmatrix} 20 & 30 & 30 \end{bmatrix}$
  $A = \begin{bmatrix} 5 & 6 & 5 \\ 15 & 10 & 3 \\ 4 & 5 & 12 \end{bmatrix}$

$$\begin{aligned} \min \; & 4x_1 + 10x_2 + 7x_3 \\ & 5x_1 + 15x_2 + 4x_3 \geq 20 \\ & 6x_1 + 10x_2 + 5x_3 \geq 30 \\ & 5x_1 + 3x_2 + 12x_3 \geq 30 \\ & x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0 \end{aligned}$$

Gli insiemi generici

- Gli insiemi generici sono dichiarati con la parola chiave `set` seguita da un'etichetta:

```
set NomeInsieme;
```

Gli insiemi generici

- Gli insiemi generici sono dichiarati con la parola chiave `set` seguita da un'etichetta:

```
set NomeInsieme;
```

- Assegnazione di elementi a un insieme generico:

```
set NomeInsieme := e1 e2 e3 e4 e5;
```

Possiamo separare la dichiarazione di un insieme (nel file .mod) dall'assegnazione (file .dat).

Gli insiemi generici

- Gli insiemi generici sono dichiarati con la parola chiave `set` seguita da un'etichetta:

```
set NomeInsieme;
```

- Assegnazione di elementi a un insieme generico:

```
set NomeInsieme := e1 e2 e3 e4 e5;
```

Possiamo separare la dichiarazione di un insieme (nel file .mod) dall'assegnazione (file .dat).

Attenzione

Dopo aver dichiarato un insieme, si possono definire *variabili*, *parametri*, *vincoli* etc. indicizzati sull'insieme

Altri tipi di insieme

- insiemi ordinati (esiste un *primo* elemento, un *successivo* etc.)

```
set NomeInsieme ordered;
```

Altri tipi di insieme

- insiemi ordinati (esiste un *primo* elemento, un *successivo* etc.)

```
set NomeInsieme ordered;
```

- insiemi ordinati e ciclici (il primo elemento è anche successivo al primo)

```
set NomeInsieme circular;
```

Altri tipi di insieme

- insiemi ordinati (esiste un *primo* elemento, un *successivo* etc.)

```
set NomeInsieme ordered;
```

- insiemi ordinati e ciclici (il primo elemento è anche successivo al primo)

```
set NomeInsieme circular;
```

- insiemi numerici

```
set NomeInsieme := 1 .. N;
set NomeInsieme := 1 .. N by p;
```

Altri tipi di insieme

- insiemi ordinati (esiste un *primo* elemento, un *successivo* etc.)

```
set NomeInsieme ordered;
```

- insiemi ordinati e ciclici (il primo elemento è anche successivo al primo)

```
set NomeInsieme circular;
```

- insiemi numerici

```
set NomeInsieme := 1 .. N;
set NomeInsieme := 1 .. N by p;
```

Quando si dichiara un insieme numerico, si sta di fatto facendo anche un'assegnazione. Pertanto l'assegnazione di un insieme numerico non deve essere ripetuta nel file `.dat`.

Altri tipi di insieme

- insiemi ordinati (esiste un *primo* elemento, un *successivo* etc.)

```
set NomeInsieme ordered;
```

- insiemi ordinati e ciclici (il primo elemento è anche successivo al primo)

```
set NomeInsieme circular;
```

- insiemi numerici

```
set NomeInsieme := 1 .. N;
set NomeInsieme := 1 .. N by p;
```

Quando si dichiara un insieme numerico, si sta di fatto facendo anche un'assegnazione. Pertanto l'assegnazione di un insieme numerico non deve essere ripetuta nel file `.dat`.

- insiemi dichiarati come sottoinsieme

```
set A within B;
```

la futura assegnazione di *A* deve assicurare che *A* sia sottoinsieme di *B*

Operatori su insiemi

Operatore/Funzione	Significato
`A union B`	insieme di elementi che stanno in A, o B
`A inter B`	insieme di elementi che stanno sia in A, sia in B
`A diff B`	insieme di elementi che stanno in A, ma non in B
`A symdiff B`	insieme di elementi che stanno in A, o in B, ma non in entrambi
`A cross B`	insieme delle coppie ordinate del prodotto cartesiano tra A e B ($\{(a, b) \mid a \in A, b \in B\}$)
`card(A)`	numero di elementi che stanno in A

Operatori su insiemi

Operatore/Funzione	Significato
`A union B`	insieme di elementi che stanno in A, o B
`A inter B`	insieme di elementi che stanno sia in A, sia in B
`A diff B`	insieme di elementi che stanno in A, ma non in B
`A symdiff B`	insieme di elementi che stanno in A, o in B, ma non in entrambi
`A cross B`	insieme delle coppie ordinate del prodotto cartesiano tra A e B ($\{(a, b) \mid a \in A, b \in B\}$)
`card(A)`	numero di elementi che stanno in A

Valgono le seguenti regole:

- le operazioni tra insiemi all'interno di una espressione complessa vengono effettuate da sinistra verso destra;
- gerarchia: inter seguito da union, diff, symdiff (stessa prioritá);
- A union B inter C diff D equivale a [A union (B inter C)] diff D.

Operatori tra insiemi ordinati

Funzione	Significato
`first(A)`	primo elemento di *A*
`last(A)`	ultimo elemento di *A*
`next(a,A)`	elemento di *A* dopo *a*
`prev(a,A)`	elemento di *A* prima di *a*
`next(a,A,k)`	*k*-esimo elemento di *A* dopo *a*
`prev(a,A,k)`	*k*-esimo elemento di *A* prima di *a*
`ord(a,A)`	posizione di *a* in *A*
`ord0(a,A)`	come `ord(a,A)` ma restituisce 0 se *a* non è in *A*
`member(k,A)`	elemento di *A* in *k*-esima posizione

Gli insiemi multidimensionali (i)

Nel caso di insiemi multidimensionali, la dimensione deve essere definita nella dichiarazione:

```
set TERNE dimension 3;
```

In questo caso, l'insieme è costituito da terne *ordinate*. L'assegnazione di valori nel file `.dat` si può effettuare in due modi:

```
set TERNE := (a,b,c) (d,e,f) (g,h,i) (l,m,n);
```

oppure

```
set TERNE :=
a b c
d e f
g h i
l m n;
```

oppure

```
set TERNE := a b c d e f g h i l m n;
```

Gli insiemi multidimensionali (ii)

Insiemi multidimensionali di dimensione *p* possono essere ottenuti come prodotto cartesiano di *p* insiemi, che si indica in AMPL con `cross`:

```
set A;
set B;
set C;
set TERNE := A cross B cross C;
```

Gli insiemi multidimensionali (ii)

Insiemi multidimensionali di dimensione *p* possono essere ottenuti come prodotto cartesiano di *p* insiemi, che si indica in AMPL con `cross`:

```
set A;
set B;
set C;
set TERNE := A cross B cross C;
```

Viceversa, da un insieme multidimensionale è possibile ottenere gli insiemi componenti con l'istruzione `setof`:

```
set A := setof{(i,j,k) in TERNE} i;
set B := setof{(i,j,k) in TERNE} j;
set C := setof{(i,j,k) in TERNE} k;
```

dietagen: .mod e .dat: insiemi

Modello del problema

- Insiemi: I (risorse) e J (richieste)
- Parametri: costi C_i, $\forall i \in I$
  quantità R_j, $\forall j \in J$
  apporto A_{ij}, $\forall i \in I, j \in J$
- Variabili: acquisti $x_i \; \forall i \in I$
- Funzione obiettivo e Vincoli

$$\min \quad \sum_{i \in I} C_i x_i$$

$$s.t.$$

$$\sum_{i \in I} A_{ij} x_i \geq R_j \qquad \forall \, j \in J$$

$$x_i \in \mathbb{R}_+ \; [\, \mathbb{Z}_+ \mid \{0,1\} \,] \qquad \forall \, i \in I$$

Istanza del problema

- $I = \{verdura, carne, frutta\}$
  $J = \{proteine, ferro, vitamine\}$
- $C = \begin{bmatrix} 4 & 10 & 7 \end{bmatrix}$
  $R = \begin{bmatrix} 20 & 30 & 30 \end{bmatrix}$
  $A = \begin{bmatrix} 5 & 6 & 5 \\ 15 & 10 & 3 \\ 4 & 5 & 12 \end{bmatrix}$

$$\begin{aligned} \min \; & 4x_1 + 10x_2 + 7x_3 \\ & 5x_1 + 15x_2 + 4x_3 \geq 20 \\ & 6x_1 + 10x_2 + 5x_3 \geq 30 \\ & 5x_1 + 3x_2 + 12x_3 \geq 30 \\ & x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0 \end{aligned}$$

Espressioni indicizzanti

Gli insiemi sono utilizzati nell'ambito di espressioni indicizzanti per definire gli indici di variabili, parametri, vincoli, sommatorie etc.
Le espressioni indicizzanti sono racchiuse tra `{...}`.

- Si possono utilizzare insiemi precedentemente dichiarati
  `{A}` `{B}` `{I}` `{J}`

Espressioni indicizzanti

Gli insiemi sono utilizzati nell'ambito di espressioni indicizzanti per definire gli indici di variabili, parametri, vincoli, sommatorie etc.
Le espressioni indicizzanti sono racchiuse tra `{...}`.

- Si possono utilizzare insiemi precedentemente dichiarati
  `{A}` `{B}` `{I}` `{J}`
- insiemi ottenuti a partire da insiemi dichiarati
  `{A cross B}` `{B diff A}` `{I inter J}`

Espressioni indicizzanti

Gli insiemi sono utilizzati nell'ambito di espressioni indicizzanti per definire gli indici di variabili, parametri, vincoli, sommatorie etc.
Le espressioni indicizzanti sono racchiuse tra `{...}`.

- Si possono utilizzare insiemi precedentemente dichiarati
  `{A}` `{B}` `{I}` `{J}`
- insiemi ottenuti a partire da insiemi dichiarati
  `{A cross B}` `{B diff A}` `{I inter J}`
- gli insiemi multidimensionali possono essere definiti in modo implicito come segue
  `{I,J}` `{I,J,A diff B}`
  equivalgono a (rispettivamente)
  `{I cross J}` `{I cross J cross (A diff B)}`

Espressioni indicizzanti

Gli insiemi sono utilizzati nell'ambito di espressioni indicizzanti per definire gli indici di variabili, parametri, vincoli, sommatorie etc.
Le espressioni indicizzanti sono racchiuse tra `{...}`.

- Si possono utilizzare insiemi precedentemente dichiarati
  `{A}` `{B}` `{I}` `{J}`
- insiemi ottenuti a partire da insiemi dichiarati
  `{A cross B}` `{B diff A}` `{I inter J}`
- gli insiemi multidimensionali possono essere definiti in modo implicito come segue
  `{I,J}` `{I,J,A diff B}`
  equivalgono a (rispettivamente)
  `{I cross J}` `{I cross J cross (A diff B)}`
- si possono esplicitare i nomi degli indici...
  `{a in A}` `{i in I, j in J}` `{(u,v) in I cross J}`

Espressioni indicizzanti

Gli insiemi sono utilizzati nell'ambito di espressioni indicizzanti per definire gli indici di variabili, parametri, vincoli, sommatorie etc.
Le espressioni indicizzanti sono racchiuse tra `{...}`.

- Si possono utilizzare insiemi precedentemente dichiarati
  `{A}` `{B}` `{I}` `{J}`
- insiemi ottenuti a partire da insiemi dichiarati
  `{A cross B}` `{B diff A}` `{I inter J}`
- gli insiemi multidimensionali possono essere definiti in modo implicito come segue
  `{I,J}` `{I,J,A diff B}`
  equivalgono a (rispettivamente)
  `{I cross J}` `{I cross J cross (A diff B)}`
- si possono esplicitare i nomi degli indici...
  `{a in A}` `{i in I, j in J}` `{(u,v) in I cross J}`
- ... e indicare restrizioni
  `{i in I, j in J : i != j}`

I parametri

- I parametri rappresentano, in forma astratta (letterale), i dati del problema.

I parametri

- I parametri rappresentano, in forma astratta (letterale), i dati del problema.
- *Dichiarazione* obbligatoria nei file .mod con la parola chiave `param` seguita da un'etichetta, e da eventuali restrizioni e valore di default:

```
param capacita_massima;
param t >= 0;
param c1 integer, <= N, default 3;
```

I parametri

- I parametri rappresentano, in forma astratta (letterale), i dati del problema.
- *Dichiarazione* obbligatoria nei file .mod con la parola chiave `param` seguita da un'etichetta, e da eventuali restrizioni e valore di default:

```
param capacita_massima;
param t >= 0;
param c1 integer, <= N, default 3;
```

- *Assegnazione* di un valore a un parametro (di solito nel file .dat):

```
param t := 2;
```

I parametri

- I parametri rappresentano, in forma astratta (letterale), i dati del problema.
- *Dichiarazione* obbligatoria nei file .mod con la parola chiave `param` seguita da un'etichetta, e da eventuali restrizioni e valore di default:

```
param capacita_massima;
param t >= 0;
param c1 integer, <= N, default 3;
```

- *Assegnazione* di un valore a un parametro (di solito nel file .dat):

```
param t := 2;
```

- L'assegnazione o l'indicazione di un valore di default (usato in assenza di assegnazione) è obbligatoria, e può avvenire nel file .dat (o nel .mod).

I parametri

- I parametri rappresentano, in forma astratta (letterale), i dati del problema.
- *Dichiarazione* obbligatoria nei file .mod con la parola chiave `param` seguita da un'etichetta, e da eventuali restrizioni e valore di default:

```
param capacita_massima;
param t >= 0;
param c1 integer, <= N, default 3;
```

- *Assegnazione* di un valore a un parametro (di solito nel file .dat):

```
param t := 2;
```

- L'assegnazione o l'indicazione di un valore di default (usato in assenza di assegnazione) è obbligatoria, e può avvenire nel file .dat (o nel .mod).
- Una volta che i valori vengono assegnati, i parametri non sono oggetto di modifica d parte del solutore: le eventuali restrizioni sono considerate solo in fase di assegnazione per segnalare violazioni (errore bloccante).

Le variabili (i)

- Le variabili rappresentano le incognite del problema, e il loro valore viene calcolato dal solutore

Le variabili (i)

- Le variabili rappresentano le incognite del problema, e il loro valore viene calcolato dal solutore
- *Dichiarazione* obbligatoria, nel file .mod, con la parola chiave `var`, seguita da un'etichetta e da eventuali restrizioni

Le variabili (i)

- Le variabili rappresentano le incognite del problema, e il loro valore viene calcolato dal solutore
- *Dichiarazione* obbligatoria, nel file .mod, con la parola chiave `var`, seguita da un'etichetta e da eventuali restrizioni
- Di default, ogni variabile è considerata reale. E' però possibile specificare se è intera o binaria

Le variabili (i)

- Le variabili rappresentano le incognite del problema, e il loro valore viene calcolato dal solutore
- *Dichiarazione* obbligatoria, nel file .mod, con la parola chiave `var`, seguita da un'etichetta e da eventuali restrizioni
- Di default, ogni variabile è considerata reale. E' però possibile specificare se è intera o binaria

Esempio:

```
var x;
var n integer;
var d binary;
var y <= 0;
var z integer >= 3 , <= 9;
```

Le variabili (ii)

E' possibile (ma non obbligatorio) utilizzare i seguenti comandi:

Per inizializzare una variabile a un determinato valore (assegnazione di un valore di partenza che il solutore è libero di modificare), si utilizza `let`:

```
let x := 10;
```

Le variabili (ii)

E' possibile (ma non obbligatorio) utilizzare i seguenti comandi:

Per inizializzare una variabile a un determinato valore (assegnazione di un valore di partenza che il solutore è libero di modificare), si utilizza `let`:

```
let x := 10;
```

Per fissare una variabile a un determinato valore, si utilizza `fix`:

```
fix x := 4;
```

Le variabili (ii)

E' possibile (ma non obbligatorio) utilizzare i seguenti comandi:

Per inizializzare una variabile a un determinato valore (assegnazione di un valore di partenza che il solutore è libero di modificare), si utilizza `let`:

```
let x := 10;
```

Per fissare una variabile a un determinato valore, si utilizza `fix`:

```
fix x := 4;
```

Per sbloccare una variabile precedentemente fissata, si utilizza `unfix`:

```
unfix x;
```

Le variabili (ii)

E' possibile (ma non obbligatorio) utilizzare i seguenti comandi:

Per inizializzare una variabile a un determinato valore (assegnazione di un valore di partenza che il solutore è libero di modificare), si utilizza `let`:

```
let x := 10;
```

Per fissare una variabile a un determinato valore, si utilizza `fix`:

```
fix x := 4;
```

Per sbloccare una variabile precedentemente fissata, si utilizza `unfix`:

```
unfix x;
```

I comandi `let`, `fix` e `unfix` possono essere utilizzati nel file `.dat` o nel file `.run`, ma non nel file `.mod`.

Parametri indicizzati su insiemi

I parametri possono essere indicizzati su uno o più insiemi facendo uso delle *espressioni indicizzanti*.

- Dichiarazione di parametri indicizzati su un insieme:

  ```
  set NomeInsieme;
  param p{NomeInsieme} >= 0;
  ```

  `p` è un vettore di parametri p a valori non negativi, con tante componenti quante sono gli elementi dell'insieme.

Parametri indicizzati su insiemi

I parametri possono essere indicizzati su uno o più insiemi facendo uso delle *espressioni indicizzanti*.

- Dichiarazione di parametri indicizzati su un insieme:

  ```
  set NomeInsieme;
  param p{NomeInsieme} >= 0;
  ```

  `p` è un vettore di parametri *p* a valori non negativi, con tante componenti quante sono gli elementi dell'insieme.

- Dichiarazione di parametri indicizzati su più insiemi:

  ```
  set I; set J
  param p1{I,J} >= 0;
  param p2{I cross J};
  ```

  `p1` (come anche `p2`) è vettore a due dimensioni di parametri, con tante componenti quante sono gli elementi del prodotto cartesiano tra `I` e `J`.

Parametri indicizzati su insiemi: esempio (i)

Dichiarazione (.mod)

```
set PROD; set ZONA;
param T;
param costo{PROD};
param limite{PROD};
param prezzo{PROD,ZONA};
param sconto{PROD cross ZONA} >=0 , <1;
param domanda{PROD,ZONA,1..T};
```

Parametri indicizzati su insiemi: esempio (i)

Dichiarazione (.mod)

```
set PROD; set ZONA;
param T;
param costo{PROD};
param limite{PROD};
param prezzo{PROD,ZONA};
param sconto{PROD cross ZONA} >=0 , <1;
param domanda{PROD,ZONA,1..T};
```

Abbiamo dichiarato:

- il parametro `T` (scalare);

Parametri indicizzati su insiemi: esempio (i)

Dichiarazione (.mod)

```
set PROD; set ZONA;
param T;
param costo{PROD};
param limite{PROD};
param prezzo{PROD,ZONA};
param sconto{PROD cross ZONA} >=0 , <1;
param domanda{PROD,ZONA,1..T};
```

Abbiamo dichiarato:

- il parametro `T` (scalare);
- il parametro `costi` (o `limite`) indicizzato dall'insieme `PROD`: `costi` è un vettore che ha tante componenti quanti sono gli elementi di `PROD`;

Parametri indicizzati su insiemi: esempio (i)

Dichiarazione (.mod)

```
set PROD; set ZONA;
param T;
param costo{PROD};
param limite{PROD};
param prezzo{PROD,ZONA};
param sconto{PROD cross ZONA} >=0 , <1;
param domanda{PROD,ZONA,1..T};
```

Abbiamo dichiarato:

- il parametro `T` (scalare);
- il parametro `costi` (o `limite`) indicizzato dall'insieme `PROD`: `costi` è un vettore che ha tante componenti quanti sono gli elementi di `PROD`;
- il parametro a due dimensioni `prezzo` (o `sconto`), indicizzato dagli insiemi `PROD` e `ZONA` (prodotto cartesiano);

Parametri indicizzati su insiemi: esempio (i)

Dichiarazione (.mod)

```
set PROD; set ZONA;
param T;
param costo{PROD};
param limite{PROD};
param prezzo{PROD,ZONA};
param sconto{PROD cross ZONA} >=0 , <1;
param domanda{PROD,ZONA,1..T};
```

Abbiamo dichiarato:

- il parametro `T` (scalare);
- il parametro `costi` (o `limite`) indicizzato dall'insieme `PROD`: `costi` è un vettore che ha tante componenti quanti sono gli elementi di `PROD`;
- il parametro a due dimensioni `prezzo` (o `sconto`), indicizzato dagli insiemi `PROD` e `ZONA` (prodotto cartesiano);
- il parametro a tre dimensioni `domanda`, indicizzato dagli insiemi `PROD`, `ZONA` e dall'insieme dei numeri interi che vanno da 1 a T.

Parametri indicizzati su insiemi: esempio (ii)

Assegnazione (.dat)

```
set PROD := p1 p2;
set ZONA := z1 z2 z3;

param T := 2;

param costi :=
p1 5
p2 4;

param limite :=
p1 1500
p2 2800;

param:   prezzo :=
p1 z1    2
p1 z2    7
p1 z3    8
p2 z1    5
p2 z2    9
p2 z3    4;
```

```
param   sconto :=
p1 z1   0.2
p1 z2   0.1
p1 z3   0.3
p2 z1   0.0
p2 z2   0.4
p2 z3   0.2;

param      domanda :=
p1 z1 1    10
p1 z1 2    32
p1 z2 1    15
p1 z2 2    25
p1 z3 1    12
p1 z3 2    27
p2 z1 1    13
p2 z1 2    18
p2 z2 1    22
p2 z2 2    15
p2 z3 1    20
p2 z3 2    19;
```

`dietagen`: `.mod` e `.dat`: parametri

Modello del problema

- Insiemi: I (risorse) e J (richieste)
- Parametri: costi C_i, $\forall i \in I$
  quantità R_j, $\forall j \in J$
  apporto A_{ij}, $\forall i \in I, j \in J$
- Variabili: acquisti $x_i \; \forall i \in I$
- Funzione obiettivo e Vincoli

$$\min \quad \sum_{i \in I} C_i x_i$$

$$s.t.$$

$$\sum_{i \in I} A_{ij} x_i \geq R_j \qquad \forall \, j \in J$$

$$x_i \in \mathbb{R}_+ \; [\, \mathbb{Z}_+ \mid \{0,1\} \,] \qquad \forall \, i \in I$$

Istanza del problema

- $I = \{verdura, carne, frutta\}$
  $J = \{proteine, ferro, vitamine\}$
- $C = \begin{bmatrix} 4 & 10 & 7 \end{bmatrix}$
  $R = \begin{bmatrix} 20 & 30 & 30 \end{bmatrix}$
  $A = \begin{bmatrix} 5 & 6 & 5 \\ 15 & 10 & 3 \\ 4 & 5 & 12 \end{bmatrix}$

$$\begin{aligned} \min \; & 4x_1 + 10x_2 + 7x_3 \\ & 5x_1 + 15x_2 + 4x_3 \geq 20 \\ & 6x_1 + 10x_2 + 5x_3 \geq 30 \\ & 5x_1 + 3x_2 + 12x_3 \geq 30 \\ & x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0 \end{aligned}$$

Assegnazioni multiple (i)

E' possibile, con uno stesso "`:=`", assegnare i valori a due o più parametri *indicizzati sullo stesso insieme*.
In questo caso bisogna usare "`:`" dopo `param`.

```
param: costi limite:=
p1     5     1500
p2     4     2800;

param:   prezzo sconto :=
p1 z1    2      0.2
p1 z2    7      0.1
p1 z3    8      0.3
p2 z1    5      0.0
p2 z2    9      0.4
p2 z3    4      0.2;
```

Assegnazioni multiple (ii)

In modo analogo, l'uso di ":" consente l'assegnazione di parametri a più dimensioni sotto forma tabellare.

```
param prezzo: z1    z2    z3 :=
p1           2     7     8
p2           5     9     4;

param domanda :=
[*,*,1] :   z1    z2    z3 :=
p1         10    15    12
p2         13    22    20
[*,*,2] :   z1    z2    z3 :=
p1         32    25    27
p2         18    15    19;
```

Controllo e calcolo di parametri

E' possibile dichiarare parametri assegnando direttamente valori "calcolati" tramite espressioni algebriche (vedi dopo):

```
set PROD;
param offerta{PROD};
param offertatot := sum{p in PROD} offerta[p];
param offertamax := max{p in PROD} offerta[p];
```

Controllo e calcolo di parametri

E' possibile dichiarare parametri assegnando direttamente valori "calcolati" tramite espressioni algebriche (vedi dopo):

```
set PROD;
param offerta{PROD};
param offertatot := sum{p in PROD} offerta[p];
param offertamax := max{p in PROD} offerta[p];
```

L'istruzione `check` effettua controlli in base ad espressione logiche:

```
set PROD;
param offertatot >0;
param offerta{PROD} >0;
check: sum{p in PROD} offerta[p]=offertatot;
```

Dichiarazione di variabili

Analogamente ai parametri, le variabili possono essere indicizzate su uno o più insiemi facendo uso delle *espressioni indicizzanti*

```
set PROD; set I; set J;
param limiteBase;
param limite{I,J} default limiteBase;

var x{PROD};
vax y{I,J};
var z{PROD,I,J};
var w{i in I, j in J : limite[i,j] > 10 };
```

dietagen: .mod e .dat: variabili

Modello del problema

- Insiemi: I (risorse) e J (richieste)
- Parametri: costi C_i, $\forall i \in I$
  quantità R_j, $\forall j \in J$
  apporto A_{ij}, $\forall i \in I, j \in J$
- Variabili: acquisti $x_i \; \forall i \in I$
- Funzione obiettivo e Vincoli

$$\min \quad \sum_{i \in I} C_i x_i$$

$$s.t.$$

$$\sum_{i \in I} A_{ij} x_i \geq R_j \qquad \forall \, j \in J$$

$$x_i \in \mathbb{R}_+ \; [\, \mathbb{Z}_+ \mid \{0,1\} \,] \qquad \forall \, i \in I$$

Istanza del problema

- $I = \{verdura, carne, frutta\}$
  $J = \{proteine, ferro, vitamine\}$
- $C = \begin{bmatrix} 4 & 10 & 7 \end{bmatrix}$
  $R = \begin{bmatrix} 20 & 30 & 30 \end{bmatrix}$
  $A = \begin{bmatrix} 5 & 6 & 5 \\ 15 & 10 & 5 \\ 4 & 5 & 12 \end{bmatrix}$

$$\begin{aligned} \min \; & 4x_1 + 10x_2 + 7x_3 \\ & 5x_1 + 15x_2 + 4x_3 \geq 20 \\ & 6x_1 + 10x_2 + 5x_3 \geq 30 \\ & 5x_1 + 3x_2 + 12x_3 \geq 30 \\ & x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0 \end{aligned}$$

Espressioni algebriche

Parametri e variabili indicizzati su insiemi possono essere composti in espressioni algebriche:

- usano indici indicati esplicitamente da espressioni indicizzanti
- accesso a parametri e variabili indicizzate con “[...]” (indici separati da “,”)

Espressioni algebriche

Parametri e variabili indicizzati su insiemi possono essere composti in espressioni algebriche:

- usano indici indicati esplicitamente da espressioni indicizzanti
- accesso a parametri e variabili indicizzate con “[...]” (indici separati da “,”)

Dichiarazione di funzioni obiettivo (.mod)

```
minimize totCosto: sum{i in PROD} costo[i]*x[i];
maximize totRicavo:
  sum{i in PROD, k in ZONA} prezzo[i,k]*sconto[i,k]*x[i];
```

Espressioni algebriche

Parametri e variabili indicizzati su insiemi possono essere composti in espressioni algebriche:

- usano indici indicati esplicitamente da espressioni indicizzanti
- accesso a parametri e variabili indicizzate con “[...]” (indici separati da “,”)

Dichiarazione di funzioni obiettivo (.mod)

```
minimize totCosto: sum{i in PROD} costo[i]*x[i];
maximize totRicavo:
  sum{i in PROD, k in ZONA} prezzo[i,k]*sconto[i,k]*x[i];
```

Dichiarazione di vincoli (.mod)

```
s.t. qualita: sum{i in PROD: i < 5} x[i]
                    <= 0.3 * sum{i in PROD} x[i];
```

Espressioni algebriche

Parametri e variabili indicizzati su insiemi possono essere composti in espressioni algebriche:

- usano indici indicati esplicitamente da espressioni indicizzanti
- accesso a parametri e variabili indicizzate con “[...]” (indici separati da “,”)

Dichiarazione di funzioni obiettivo (.mod)

```
minimize totCosto: sum{i in PROD} costo[i]*x[i];
maximize totRicavo:
  sum{i in PROD, k in ZONA} prezzo[i,k]*sconto[i,k]*x[i];
```

Dichiarazione di vincoli (.mod)

```
s.t. qualita: sum{i in PROD: i < 5} x[i]
                    <= 0.3 * sum{i in PROD} x[i];
```

`dietagen:` `.mod` e `.dat`: vincoli?

Vincoli indicizzati su insiemi (i)

Anche i vincoli possono essere indicizzati. Quindi, in una sola espressione si possono dichiarare più vincoli.

```
set PROD; set REPARTI;

param ore_lavoro{PROD,REPARTI};
param max_ore{REPARTI};
param prezzo{PROD};

var x{PROD};

maximize ricavo : sum{i in PROD} prezzo[i]*x[i];

s.t. v_ore{j in REPARTI} :
    sum{i in PROD} ore_lavoro[i,j]*x[i] <= max_ore[j];
```

Vincoli indicizzati su insiemi (i)

Anche i vincoli possono essere indicizzati. Quindi, in una sola espressione si possono dichiarare più vincoli.

```
set PROD; set REPARTI;

param ore_lavoro{PROD,REPARTI};
param max_ore{REPARTI};
param prezzo{PROD};

var x{PROD};

maximize ricavo : sum{i in PROD} prezzo[i]*x[i];

s.t. v_ore{j in REPARTI} :
    sum{i in PROD} ore_lavoro[i,j]*x[i] <= max_ore[j];
```

In questo modo, è possibile esprimere i vincoli senza conoscere a priori quanti sono gli elementi dell'insieme `REPARTI`.

Vincoli indicizzati su insiemi (i)

Anche i vincoli possono essere indicizzati. Quindi, in una sola espressione si possono dichiarare più vincoli.

```
set PROD; set REPARTI;

param ore_lavoro{PROD,REPARTI};
param max_ore{REPARTI};
param prezzo{PROD};

var x{PROD};

maximize ricavo : sum{i in PROD} prezzo[i]*x[i];

s.t. v_ore{j in REPARTI} :
    sum{i in PROD} ore_lavoro[i,j]*x[i] <= max_ore[j];
```

In questo modo, è possibile esprimere i vincoli senza conoscere a priori quanti sono gli elementi dell'insieme `REPARTI`.

`dietagen`: `.mod` e `.dat`: vincoli!

Vincoli indicizzati su insiemi (ii)

Alternativamente, si dovrebbe prima assegnare gli elementi $\{R_1, R_2, \ldots, R_m\}$ all'insieme *R*EPARTI, ed esprimere i vincoli come:

```
s.t. v_ore_1 :
  sum{i in PROD} ore_lavoro[i,"R1"]*x[i] <= max_ore["R1"];

s.t. v_ore_2 :
  sum{i in PROD} ore_lavoro[i,"R2"]*x[i] <= max_ore["R2"];

   .
   .
   .

s.t. v_ore_m :
  sum{i in PROD} ore_lavoro[i,"Rm"]*x[i] <= max_ore["Rm"];
```

Vincoli indicizzati su insiemi (ii)

Alternativamente, si dovrebbe prima assegnare gli elementi $\{R_1, R_2, \ldots, R_m\}$ all'insieme *R*EPARTI, ed esprimere i vincoli come:

```
s.t. v_ore_1 :
  sum{i in PROD} ore_lavoro[i,"R1"]*x[i] <= max_ore["R1"];

s.t. v_ore_2 :
  sum{i in PROD} ore_lavoro[i,"R2"]*x[i] <= max_ore["R2"];

   .
   .
   .

s.t. v_ore_m :
  sum{i in PROD} ore_lavoro[i,"Rm"]*x[i] <= max_ore["Rm"];
```

...ma il modello non sarebbe indipendente dai dati!

Le espressioni (i)

Funzione	Significato
`abs(x)`	valore assoluto di x
`sin(x)`	$\sin(x)$
`cos(x)`	$\cos(x)$
`tan(x)`	$\tan(x)$
`asin(x)`	$\arcsin(x)$
`acos(x)`	$\arccos(x)$
`atan(x)`	$\arctan(x)$
`exp(x)`	$\exp(x)$
`sqrt(x)`	radice quadrata di x, $\sqrt{x}$
`log(x)`	logaritmo naturale di x, $\ln(x)$
`log10(x)`	logaritmo in base 10 di x, $\log(x)$
`ceil(x)`	parte intera superiore di x, $\lceil x \rceil$
`floor(x)`	parte intera inferiore di x, $\lfloor x \rfloor$
`round(x)`	intero più vicino a x

Le espressioni (ii)

Operatori aritmetici	Significato
^	potenza
+	somma
-	sottrazione
*	prodotto
/	divisione
div	divisione intera
mod	modulo
sum	sommatoria
prod	produttoria
min	minimo
max	massimo
>	maggiore
>=	maggiore o uguale
<	minore
<=	minore o uguale
=	uguale
<>, !=	diverso

Le espressioni (iii)

Operatori logici	Significato
`not`	negazione logica
`or`	“or” logico
`and`	“and” logico
`exists`	quantificatore esistenziale logico
`forall`	quantificatore universale logico
`if then else`	espressione condizionale

Esempio modello di PL (i)

Esempio 2.1

Un'acciaieria acquista rottame di quattro tipi differenti (T1, T2, T3, T4) per ottenere due leghe (L1, L2) con caratteristiche chimiche differenti. I quattro tipi di rottame hanno i seguenti contenuti in percentuale di Piombo, Zinco e Stagno, e il seguente prezzo unitario di acquisto (in migliaia di € a tonnellata).

	T1	**T2**	**T3**	**T4**
Piombo	40%	30%	25%	38%
Zinco	35%	40%	35%	32%
Stagno	25%	30%	40%	30%
prezzo	2.5	1.8	2	2.2

La lega L1 deve avere un contenuto non superiore al 30% di piombo, al 60% di zinco e al 42% di stagno.
La lega L2 deve avere un contenuto non superiore al 46% di piombo, al 38% di zinco e al 56% di stagno.
Definire le quantità di ciascun tipo di rottame da utilizzare in ciascuna delle leghe in modo da minimizzare il costo complessivo e soddisfare esattamente un ordine di 1500 tonnellate di lega L1 e 2000 tonnellate di lega L2.

Esempio modello di PL (ii)

$$\min\ 2.5(x_{11}+x_{12})+1.8(x_{21}+x_{22})+2(x_{31}+x_{32})+2.2(x_{41}+x_{42})$$

$$0.4x_{11}+0.3x_{21}+0.25x_{31}+0.38x_{41} \leq 0.3(x_{11}+x_{21}+x_{31}+x_{41})$$

$$0.35x_{11}+0.4x_{21}+0.35x_{31}+0.32x_{41} \leq 0.6(x_{11}+x_{21}+x_{31}+x_{41})$$

$$0.25x_{11}+0.3x_{21}+0.40x_{31}+0.3x_{41} \leq 0.42(x_{11}+x_{21}+x_{31}+x_{41})$$

$$0.4x_{12}+0.3x_{22}+0.25x_{32}+0.38x_{42} \leq 0.46(x_{12}+x_{22}+x_{32}+x_{42})$$

$$0.35x_{12}+0.4x_{22}+0.35x_{32}+0.32x_{42} \leq 0.38(x_{12}+x_{22}+x_{32}+x_{42})$$

$$0.25x_{12}+0.3x_{22}+0.40x_{32}+0.3x_{42} \leq 0.56(x_{12}+x_{22}+x_{32}+x_{42})$$

$$x_{11}+x_{21}+x_{31}+x_{41}=1500$$

$$x_{12}+x_{22}+x_{32}+x_{42}=2000$$

$$x_{ij} \geq 0 \quad i=1,\ldots,4 \quad j=1,2$$

Esempio modello di PL (modello generale)

- Insiemi: I (`ROTTAMI`); J (`LEGHE`); K (`METALLI`).
- Parametri: C_i (`Prezzo` rottame $i \in I$); R_j (`Ordine` lega $j \in J$); $A_{k,i}$ (`cont`enuto metallo $k \in K$ in rottame $i \in I$); $U_{k,j}$ (`con`enuto `max` di metallo $k \in K$ nella lega $j \in J$).
- Variabili: x_{ij} (acquisti di rottame $i \in I$ usati per la lega $j \in J$)
- Modello PL:

$$
\begin{aligned}
\min \quad & \sum_{i \in I} C_i \sum_{j \in J} x_{ij} \\
s.t. \quad & \\
& \sum_{i \in I} A_{ki} x_{ij} \leq U_{kj} \sum_{i \in I} x_{ij} && \forall\, j \in J, k \in K \\
& \sum_{i \in I} x_{ij} = R_j && \forall\, j \in J, \\
& x_{ij} \in \mathbb{R}_+ && \forall\, i \in I, j \in J
\end{aligned}
$$

Esempio modello di PL (iii)

rottame.mod

```
set ROTTAMI;
set LEGHE;
set METALLI;
param cont{METALLI,ROTTAMI};
param prezzo{ROTTAMI};
param cont_max{METALLI,LEGHE};
param ordine{LEGHE};

var x{ROTTAMI,LEGHE} >= 0;

minimize f:
  sum{i in ROTTAMI} prezzo[i]*sum{j in LEGHE} x[i,j];

s.t. v_c{k in METALLI, j in LEGHE}: sum{i in ROTTAMI}
cont[k,i]*x[i,j] <= cont_max[k,j]*sum{i in ROTTAMI}x[i,j];

s.t. v_o{j in LEGHE}: sum{i in ROTTAMI} x[i,j] = ordine[j];
```

Esempio modello di PL (iv)

rottame.dat

```
set ROTTAMI := T1 T2 T3 T4;
set LEGHE := L1 L2;
set METALLI := Piombo Zinco Stagno;

param cont : T1   T2   T3   T4 :=
Piombo      0.4  0.3  0.25 0.38
Zinco       0.35 0.4  0.35 0.32
Stagno      0.25 0.3  0.4  0.3;

param prezzo :=
T1 2.5
T2 1.8
T3 2
T4 2.2;
```

Esempio modello di PL (v)

```
param cont_max : L1    L2 :=
Piombo           0.3   0.46
Zinco            0.6   0.38
Stagno           0.42  0.56;

param ordine :=
L1 1500
L2 2000;
```

Esempio modello di PL (vi)

rottame.run

```
reset;
model rottame.mod;
data rottame.dat;

option solver cplex;

solve;

display f;
display x;
```