# Complementi di Ricerca Operativa

Prof. Trubian Marco
5 CFU

**Luca Cappelletti**

Lectures Notes
Year 2017/18

Magistrale Informatica LM-18
Università statale di Milano
Italy
October 5, 2017

# Contents

**1 Chapter 2** **2**
1.1 Modelli quadratici . . . 2
1.1.1 Casi Possibili . . . 2
1.1.2 Esempio . . . 2
1.2 Introduzione agli algoritmi . . . 2
1.2.1 Quanto è buono un algoritmo di ottimizzazione? . . . 2
1.2.2 Come determiniamo un punto di minimo . . . 3
1.2.3 Condizioni di Wolf . . . 3

**2 Ampl** **4**
2.1 Introduzione alla programmazione lineare . . . 4
2.1.1 Primo esempio . . . 4
2.1.2 Secondo esempio con separazione dei dati dal model . . . 6
2.1.3 Primo laboratorio . . . 8
2.1.4 Secondo laboratorio . . . 8

# Chapter 1

# Chapter 2

## 1.1 Modelli quadratici

Algoritmi di ottimizzazione che approssimano localmente $f$ con modelli quadratici:

$$\min f(x) = \frac{1}{2} x^T Q x - b^T x, \text{t.c.} c \in \mathbb{R}^n$$

dove $Q$ è una matrice quadrata di ordine $n$.

### 1.1.1 Casi Possibili

- $Q$ non è semi-definita positiva: $f$ non ha un minimo. - $Q$ è definita positiva: $x^* = Q^{-1}b$ è l'unico minimo locale. Il punto $x^*$ è il **punto di ottimo globale**. - $Q$ è definita semi-positiva: - $Q$ non è singolare: $x^* = Q^{-1}b$ è l'unico minimo globale. - $Q$ è singolare: - non ho soluzioni. - ho infinite soluzioni.

### 1.1.2 Esempio

$$f(x, y) = \frac{1}{2}(ax^2 + by^2) - x$$

Riscrivo nei termini della formula per l'algoritmo:

$$f(x) = \frac{1}{2} \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

## 1.2 Introduzione agli algoritmi

Metodi di ottimizzazione continua:

- Dato un punto di inizio $x_0$, generiam una sequenza $x_{k_{k=0}}^{\infty}$. - Terminato l'algoritmo, quando le condizioni necessarie sono soddisfatte con una certa precisione, per esempio $\|\nabla f(x_k)\| \leq \epsilon$ - Monotone algorithms requires that $f(x_k) < f(x_{k-1}) \forall k$

### 1.2.1 Quanto è buono un algoritmo di ottimizzazione?

Un algoritmo è **decente** se **converge**.

**Definition 1.2.1 (Convergente globalmente)** *Un algoritmo è chiamato convergente globalmente se converge a un punto* $x^*$

// PERSE COSE DA SLIDE QUI

Un algoritmo è **buono** se **converge rapidamente**

Chiamando $x_k$ una sequenza in $\mathbb{R}^n$ che converge a $x^*$. La **convergenza** è chiamata:

- **Q-lineare** se $\exists r \in (0,1) \text{s.t.} \frac{\|x_{k+1-x^*}\|}{\|x_k - x^*\|} \leq r, \text{for} k \geq \overline{k}$

- **Q-superlineaee** se $\lim_{k\to\infty} \frac{\|x_{k+1-x^*}\|}{\|x_k - x^*\|} = 0$

- **Q-quadratica** se $\exists C > 0 \text{s.t.} \frac{\|x_{k+1-x^*}\|}{\|x_k - x^*\|^2} \leq C, \text{for} k \geq \overline{k}$

Q-quadratica implica superlineare che implica lineare.

### 1.2.2 Come determiniamo un punto di minimo

#### Line search

Dato il punto corrente determino la direzione e dopo di che determino di quanto muovermi.

#### Trust Region

Costruisco un modello quadratico in base alle informazioni locali, quindi scelgo un parametro $\Delta k$, un raggio, e scelgo la direzione risolvendo un problema di ottimizzazione vincolato al parametro $\Delta_k$.

### 1.2.3 Condizioni di Wolf

// Iniziare a guardare queste

# Chapter 2

# Ampl

## 2.1 Introduzione alla programmazione lineare

Per risolvere un problema utilizzando ampl è necessario utilizzare 3 tipi diversi di file:

1. Model file (.mod)
2. Data file (.dat)
3. Command file (.run)

Ampl carica questi file e li invia al *solver* (cplex, minos, ...), che quindi legge ed elabora il *Command file*.

***Gli esempi che seguono sono tratti dal canale youtube "Yong Wang"***: https://www.youtube.com/channel/UCXEnJBeaJx3P87A_UfZpd0Q

### 2.1.1 Primo esempio

**Esempio di Model file**

```
# PART 1: DECISION VARIABLES
var x1 >= 0; # first variable
var x2 >= 0; # second variable

# PART 2: OBJECTIVE FUNCTION
maximize z: 300*x1 + 200*x2;

# PART 3: CONSTRAINTS
s.t. M1:    2*x1 +   x2 <= 8; #s.t. significa "subject to"
s.t. M2:      x1 + 2*x2 <= 8;
```

**Esempio di Command file**

```
#RESET THE AMPL ENVIROMENT
reset;

#LOAD THE MODEL
model example1.mod;

#CHANGE THE SOLVER (optional)
option solver cplex;

```

```
#SOLVE
solve;

#SHOW RESULTS
display x1, x2, z;
```

### 2.1.2 Secondo esempio con separazione dei dati dal model

**Data file**

```
param n := 4;
param m := 4;

param    C :=
        1    50
        2    20
        3    30
        4    80;
param    A:    1    2    3    4:=
        1    400    200    150    500
        2    3    2    0    0
        3    2    2    4    4
        4    2    4    1    5;
param    B :=
        1    500
        2    7
        3    10
        4    8;
```

**Model file**

```
param n;
param m;
set J := {1..n};  #set of decision variables
set I := {1..m};  #set of constraints

param C {J} >= 0;  #objective function coefficients
param A {I,J} >= 0;  #constraint coefficients matrix
param B {I} >= 0;  #rhs of the constraints

var X {J} >=0;  #decision variables

minimize z:  sum {j in J} C[j] * X[j];

s.t. Constraint {i in I}:
    sum {j in J} A[i,j] * X[j] >= B[i];
```

**Command file**

```
#RESET THE AMPL ENVIROMENT
reset;

#LOAD THE MODEL
model example2.mod;

#LOAD THE DATA
data example2.dat;

#DISPLAY THE PROBLEM FORMULATION
```

```
expand z, Constraint;

#CHANGE THE SOLVER (optional)
option solver cplex;

#SOLVE
solve;

#SHOW RESULTS
display X, z;
```

### 2.1.3 Primo laboratorio

**Data file**

```
data;

set PROD := bands coils;

param:    rate  profit  market :=
  bands    200    25     6000
  coils    140    30     4000 ;

param avail := 40;
```

**Model file**

```
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
```

### 2.1.4 Secondo laboratorio

**Data file**

```
data;

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 ;
```

**Model file**

```
set ORIG;   # origins
set DEST;   # destinations

param supply {ORIG} >= 0;   # amounts available at origins
param demand {DEST} >= 0;   # amounts required at destinations

   check: sum {i in ORIG} supply[i] = sum {j in DEST} demand[j];

param cost {ORIG,DEST} >= 0;   # shipment costs per unit
var Trans {ORIG,DEST} >= 0;    # units to be shipped

minimize Total_Cost:
   sum {i in ORIG, j in DEST} cost[i,j] * Trans[i,j];

subject to Supply {i in ORIG}:
   sum {j in DEST} Trans[i,j] = supply[i];

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