

# Ricerca Operativa e Pianificazione delle Risorse

UniShare

Davide Cozzi  
@dlcgold

Gabriele De Rosa  
@derogab

Federica Di Lauro  
@f\_dila

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Introduzione alla Ricerca Operativa</b>	<b>4</b>
2.1	Modelli nella R.O. . . . . .	5
2.2	Soluzione Grafica . . . . .	16
2.2.1	esercizi . . . . .	23
2.3	Simplexso . . . . .	24
2.4	Metodo del Simplexso . . . . .	26
2.4.1	Test di Ottimalità . . . . .	30
2.4.2	Forma Tabellare . . . . .	37
2.4.3	Forma Matriciale . . . . .	39
<b>3</b>	<b>Dualità</b>	<b>44</b>
3.0.1	condizioni di complementarietà . . . . .	47
3.0.2	Analisi di Sensività . . . . .	48
3.0.3	Regola del 100% . . . . .	49
<b>4</b>	<b>Programmazione Lineare Intera</b>	<b>51</b>
4.0.1	Metodo Branch ad Bound . . . . .	54
<b>5</b>	<b>Programmazione Non Lineare</b>	<b>58</b>
5.0.1	Il Caso N-Dimensionale . . . . .	63
5.0.2	PNL non Vincolata . . . . .	66
5.0.3	Metodo di Newton . . . . .	69
5.1	Ottimizzazione Non Lineare Vincolata . . . . .	70
5.1.1	Dimensionality Reduction . . . . .	71
5.1.2	Lagrangiana . . . . .	71
5.1.3	KKT . . . . .	74
<b>6</b>	<b>Metaeuristiche</b>	<b>76</b>
6.1	Tabù Search . . . . .	77

---

6.2	Simulated Annealing . . . . .	78
6.3	Algoritmi Genetici . . . . .	81
6.3.1	Funzione di Fitness e Spazio di Ricerca . . . . .	84
6.3.2	Popolazione . . . . .	84
6.3.3	Selezione e Operatori Genetici . . . . .	85

# Capitolo 1

## Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: <https://github.com/dlcgold/Appunti>.

Grazie mille e buono studio!

Immagini tratte dalle slide del corso, docente V. Messina

## Capitolo 2

# Introduzione alla Ricerca Operativa

La **Ricerca Operativa** è essenziale nel *problem solving* e nell'ambito del *decision making*. Sostanzialmente quindi si studia l'ottimizzazione, massimizzando le performances, l'accuratezza dei costi etc. . . per raggiungere un obiettivo.

*Sulle slides ci sono vari esempi introduttivi di vita reale*

Un altro problema studiato dalla ricerca operativa sono le previsioni, mediante algoritmi predittivi che studiano i *pesi* delle osservazioni (cosa utile nel **Machine Learning** in quanto sono un uso di base delle **Reti Neurali**, *vari esempi introduttivi sulle slides*).

**La ricerca operativa si occupa di formalizzare un problema in un modello matematico e calcolare una soluzione ottimo o approssimata.** Essa costituisce un approccio scientifico alla risoluzione di problemi complessi da ricondurre alla matematica applicata. È utile in ambiti economici, logistici, di progettazione di servizi e di sistemi di trasporto e, ovviamente, nelle tecnologie. *È la branca della matematica più applicata.*

Il *primo passo* consiste nel costruire un modello traducendo il problema reale in linguaggio anturale in un linguaggio matematico, che non è ambiguo. Il *secondo passo* consiste nella costruzione delle soluzioni del modello tramite algoritmi e programmi di calcolo. Il *terzo passo*, ovvero l'ultimo, è l'interpretazione e la valutazione delle soluzioni del modello rispetto a quelle del problema reale.

La ricerca operativa ha origini nel 1800 in un ambiente puramente matematico. È stata resa “*algoritmica*” con la Macchina di Turing. **La ricerca operativa usa anche tecniche numeriche e non solo analitiche.**

Negli ultimi anni si sono sviluppati, mediante il concetto di **gradiente**, nuovi algoritmi per il **deep network**.

## 2.1 Modelli nella R.O.

**Definizione 1.** Data una funzione  $f : \mathbb{R} \rightarrow \mathbb{R}$  e  $X \subseteq \mathbb{R}^n$  un **problema di ottimizzazione** può essere formulato come:

$$\text{opt } f(x) \text{ s.t. } x \in X$$

dove con  $\text{opt} = \min, \max$  intendiamo che  $\text{opt}$  può essere o  $\min$  o  $\max$ , portando ad un problema di minimizzazione con  $\min f(x)$  o di massimizzazione  $\max f(x)$ .

$f(x)$  è detta **funzione obiettivo** e vale che:

$$\max[f(x) : x \in X] = -\min[-f(x) : x \in X]$$

Inoltre  $x \in \mathbb{R}^n$  è **l'insieme delle soluzioni ottenibili** o anche **regione ammissibile**.

Infine  $x \in X$  rappresenta il **vettore delle variabili decisionali** e si tratta di variabili numeriche i cui valori rappresentano la soluzione del problema.

**Si capisce che essendo in  $\mathbb{R}$  si hanno infinite soluzioni.**

**Quindi, un problema di ottimizzazione consiste nel determinare, se esiste, un punto di minimo/massimo della funzione  $f$  tra i punti dell'insieme  $X$ .** Se  $X = \mathbb{R}^n$  si ha un'ottimizzazione **non vincolata**, altrimenti,  $x \in \mathbb{R}^n$  si ha un'ottimizzazione **vincolata**, dove la ricerca dei punti di ottimo della funzione obiettivo è fatta su un sottoinsieme proprio dello spazio di definizione tenendo però conto dei vincoli. Se ho una funzione obiettivo lineare non si può avere un'ottimizzazione non vincolata (non saprei cercare massimi e minimi senza vincoli).

Abbiamo poi l'**ottimizzazione intera o a numeri interi** se  $x \in \mathbb{Z}^n$  e si possono avere ottimizzazioni miste se si hanno interi e reali. Si ha anche l'**ottimizzazione binaria** quando si hanno due vie decisionali.

**Se non specificato si intende  $X \subseteq \mathbb{R}^n$ .**

**Definizione 2.** Quando l'insieme  $X$  delle soluzioni ammissibili di un problema di ottimizzazione è espresso in un sistema di equazioni o disequazioni si parla di **problema di programmazione matematica (PM)**.

Come vincolo si ha un'espressione  $g_i(x) \{\leq, =, \geq\} 0$  ( $g_i \geq 0$  etc ...) e con  $g_i : X \rightarrow \mathbb{R}$  che è una generica funzione che lega due variabili.

**Si possono avere più vincoli ma si ha sempre l'uguale in ogni vincolo per permettere il funzionamento degli algoritmi.**

La regione ammissibile è  $X \subseteq \mathbb{R}^n$  che è l'intersezione di tutti i vincoli del problema

$$X = \{x \in \mathbb{R}^n \mid g_i(x) \{ \leq, 0, \geq \} 0, i = 1, \dots, m\}$$

Si hanno quindi  $m$  vincoli e  $n$  variabili. **Se  $x \in X$  allora  $x$  è soluzione ammissibile, se  $x \notin X$  allora  $x$  è non ammissibile**

**Esempio 1.** abbiamo la funzione obiettivo

$$\min_{x,y} (x^2 + y^2)$$

con i 3 vincoli:

$$x + y \leq 3$$

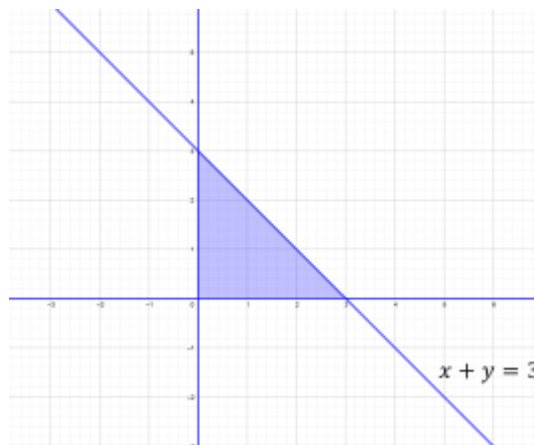
$$x \geq 0$$

$$y \geq 0$$

la regione ammissibile è:

$$\{x \in \mathbb{R}^2 \mid x + y \leq 3, x \geq 0, y \geq 0\}$$

ovvero l'area sottesa alla retta e compresa negli assi cartesiani:



Si possono avere problemi con regione non ammissibile, ovvero con  $X = \emptyset$ , che implica che il problema è mal posto oppure bisogna abbassare qualche vincolo. Si può avere un problema illimitato con:

$$\forall c \in \mathbb{R} \exists x_c \in X : f(x_c) \leq c \text{ se } opt = \min$$

$$\forall c \in \mathbb{R} \exists x_c \in X : f(x_c) \geq c \text{ se } opt = \max$$

Infine si può avere una sola soluzione ottima o più (anche infinite) soluzioni ottime tutte con lo stesso valore della funzione obiettivo.

**Esempio 2.** abbiamo la funzione obiettivo

$$\min_{x,y}(x^2 + y^2)$$

con i 3 vincoli:

$$x + y \leq -1$$

$$x \geq 0$$

$$y \geq 0$$

Non ha soluzione (è matematicamente impossibile) e il problema non è ammissibile

**Esempio 3.** abbiamo la funzione obiettivo

$$\max_{x,y}(x^2 + y^2)$$

con i 2 vincoli:

$$x \geq 0$$

$$y \geq 0$$

Ha come soluzione infinito

**Esempio 4.** abbiamo la funzione obiettivo

$$\max_{x,y,z}(z)$$

con i 4 vincoli:

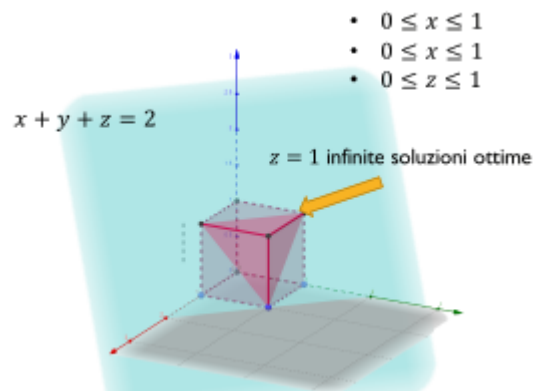
$$x + y + z = 2$$

$$0 \leq x \leq 1$$

$$0 \leq y \leq 1$$

$$0 \leq z \leq 1$$

Ha infinite soluzioni (tutte le soluzioni con  $z = 1$  e  $x + y = 1$ , in quanto cerco il max di  $z$  e come ultimo vincolo ho che al massimo è 1)





La risoluzione di un problema di Programmazione matematica consiste nel trovare una soluzione ammissibile che sia un **ottimo globale** ovvero un vettore  $x^* \in X$  tale che:

$$f(x^*) \leq f(x) \quad \forall x \in X \quad \text{se } opt = \min$$

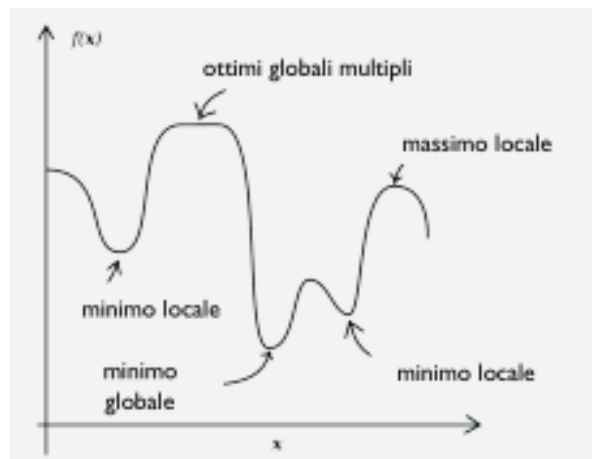
$$f(x^*) \geq f(x) \quad \forall x \in X \quad \text{se } opt = \max$$

Quando il problema è molto difficile da risolvere possiamo accontentarci di un ottimo locale, vale a dire un  $\hat{x} \in X$  tale che, fissato un  $\varepsilon > 0$  opportuno si ha che (per problemi di minimo e massimo):

$$f(\hat{x}) \leq f(x) \quad \forall x \in X : ||x - \hat{x}|| \leq \varepsilon \quad \text{se } opt = \min$$

$$f(\hat{x}) \geq f(x) \quad \forall x \in X : ||x - \hat{x}|| \leq \varepsilon \quad \text{se } opt = \max$$

Un problema di ottimizzazione può avere più ottimi locali e globali e i punti di ottimo globale sono anche di ottimo locale.



**Esempio 5.** abbiamo la funzione obiettivo:

$$\min_{x,y} ((x - 0.2)^2 + y^2)$$

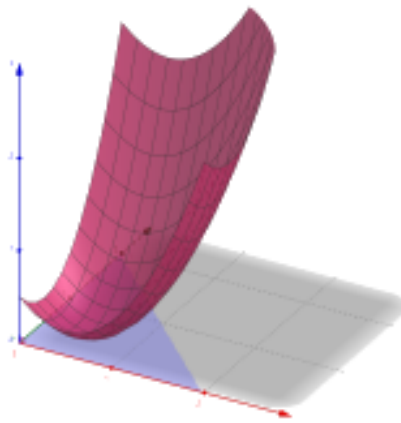
con i 3 vincoli:

$$x + y \leq 1$$

$$x \geq 0$$

$$y \geq 0$$

In viola si ha la funzione obiettivo tridimensionale e convessa e in azzurro la regione ammissibile



si ha solo un minimo globale. Si possono usare le **curve di livello** che sono le proiezioni ortogonali sul piano cartesiano ottenute intersecando il piano  $z$  con il grafico della funzione.

In generale si useranno tecniche numeriche anche se si ha che **una funzione convessa ha un solo minimo globale**

**Esempio 6.** abbiamo la funzione obiettivo:

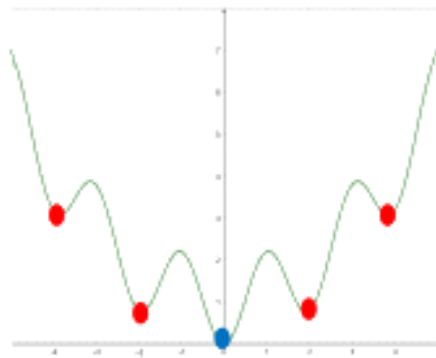
$$\min_x (0.2x^2 + (1 - \cos(\pi x)))$$

con i 2 vincoli:

$$x \geq 5$$

$$y \geq 0$$

e si ha il seguente grafico:



si ha il coseno quindi si ha una funzione nè concava nè convessa si ha quindi un ottimo globale e 4 locali

Si ha:

- **programmazione lineare (PL)**, con obiettivo e vincoli lineari:

$$\text{opt } f(x) = c^T x$$

$$X = \{x \in \mathbb{R}^n : g_i(x) \{\leq, =, \geq\} 0, \ i = 1, \dots, m \ g_i(x) = a_i^T x - b_i\}$$

- **programmazione lineare intera (PLI)**, con obiettivo e vincoli lineari interi:

$$\text{opt } f(x) = c^T x$$

$$X = \{x \in \mathbb{Z}^n : g_i(x) \{\leq, =, \geq\} 0, \ i = 1, \dots, m \ g_i(x) = a_i^T x - b_i\}$$

- **programmazione non lineare (PNL)**, con obiettivo e vincoli  $g_i(x)$  non lineari:

$$\text{opt } f(x)$$

$$X = \{x \in \mathbb{R}^n : g_i(x) \{\leq, =, \geq\} 0, \ i = 1, \dots, m \ g_i(x) = a_i^T x - b_i\}$$

**Esempio 7.** abbiamo la funzione obiettivo:

$$\min_{x,y}(x^2 + y^2)$$

con i 3 vincoli:

$$x + y \leq 1$$

$$x \geq 0$$

$$y \geq 0$$

non è lineare in quanto  $x$  e  $y$  sono al quadrato e quindi non lineare

**Esempio 8.** abbiamo la funzione obiettivo:

$$\min_{x,y}(x + y)$$

con i 2 vincoli:

$$x^2 - 1 \geq 0$$

$$y \geq 0$$

non è lineare in quanto ho un vincolo al quadrato e quindi non lineare

**Esempio 9.** abbiamo la funzione obiettivo:

$$\min_{x,y}(x + 4y)$$

con i 4 vincoli:

$$x + y = 3$$

$$x^2 - 1 \geq 0$$

$$x \geq 0$$

$$y \geq 0$$

non è lineare in quanto ho un vincolo al quadrato e quindi non lineare ma posso renderlo lineare in quanto  $x^2 - 1 = (x - 1)(x + 1)$  ma  $x + 1$  è sempre positivo quindi posso ammorbidire il vincolo rendendo il problema lineare

Ingredienti	Disponibilità massima	Costo x litro
Rum chiaro	6 l	15 €
Cola	15 l	1 €
Limone	3 l	2.5 €

**Esempio 10.**

Le dosi ideali sono: almeno il 25% di rum ( $R$ ) chiaro e il 50% di Cola ( $C$ ) e non più del 10% di limone ( $L$ ) e voglio 10L di cubalibre. Abbiamo quindi, per logica:

$$R \geq 0$$

$$C \geq 0$$

$$L \geq 0$$

$$R \leq 6$$

$$C \leq 15$$

$$L \leq 3$$

quindi:

$$0 \leq R \leq 6$$

$$0 \leq C \leq 15$$

$$0 \leq L \leq 3$$

inoltre:

$$R + C + L \geq 10$$

che è:

$$R + C + L - 10 \geq 0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} R \\ C \\ L \end{bmatrix} - 10 \geq 0$$

quindi:

$$a_1^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, x = \begin{bmatrix} R \\ C \\ L \end{bmatrix}, b_1 = 10$$

Cosa vuol dire almeno il 25% di rum chiaro?

$$R \geq 0.25 \cdot (R + C + L)$$

Cosa vuol dire almeno il 50% di cola?

$$C \geq 0.5 \cdot (R + C + L)$$

Cosa vuol dire almeno il 25% di limone?

$$L \geq 0.1 \cdot (R + C + L)$$

che sono vincoli lineari.

quindi il costo è:

$$\min_{R,C,L} (15R + C + 2.5L)$$

che è una funzione obiettivo lineare. Osserviamo che la funzione obiettivo può essere scritta anche nella seguente forma compatta:

$$\min c^t x, \text{ con } c^t = [15 \quad 1 \quad 2.5] \text{ e } x = \begin{bmatrix} R \\ C \\ L \end{bmatrix}$$

Ora riscriviamo in forma matriciale:

$$\min cx$$

$$Ax \geq b$$

$$x \geq 0$$

con:

$$c = [15 \quad 1 \quad 2.5]$$

$$x = \begin{bmatrix} R \\ C \\ L \end{bmatrix}$$

la prima riga sarà  $R + C + L \geq 10$

la seconda sarà  $R \geq 0.25 \cdot (R + C + L) \geq 0 \rightarrow (1 - 0.25)R - 0.25C - 0.25L$

la terza sarà  $C \geq 0.5 \cdot (R + C + L) \geq 0 \rightarrow -0.5R + (1 - 0.5)C - 0.5L$

la quarta sarà  $L \geq 0.1 \cdot (R + C + L) \geq 0 \rightarrow -0.1R - 0.25C + 0.9(1 - 0.1)L$

la quinta sarà  $0 \leq R \leq 6 \rightarrow -R \geq -6$

la sesta sarà  $0 \leq C \leq 15 \rightarrow -C \geq -15$

la settima sarà  $0 \leq L \leq 3 \rightarrow -L \geq -3$

e quindi:

$$Ax = \begin{bmatrix} 1 & 1 & 1 \\ 0.75 & -0.25 & -0.25 \\ -0.5 & 0.5 & -0.5 \\ 0.1 & 0.1 & -0.9 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} R \\ C \\ L \end{bmatrix} \geq \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ -6 \\ -15 \\ -3 \end{bmatrix}$$

**Esempio 11.** Si ha che:

Vogliamo comprare per il nostro compleanno (plurale maiestatis) un certo numero  $n$  di giochi sulla piattaforma Steam avendo a disposizione:

- 150 euro come budget massimo
- 150 GB di spazio sul pc

Vogliamo i giochi che più ci piacciono (su una scala da 1 a 5)



						
Costo	39,99€	39,99€	59,99€	59,99€	19,99€	29,99€
Spazio	30 GB	40 GB	12 GB	46 GB	12 GB	72 GB
Gradimento	2	5	2	5	3	5

Il comprare o no un videogioco può essere modellizzato per mezzo di variabili decisionali binarie associate ad ogni gioco usando variabili binarie:

$$x_i \in \{0, 1\} \quad i = 1, \dots, n$$

con  $x_i = 1$  si compra con  $x_i = 0$  no.

Non superare il budget massimo di 100 euro può essere espresso dalla seguente relazione:

$$39,99 \cdot x_1 + 39,99 \cdot x_2 + 59,99 \cdot x_3 + 59,99 \cdot x_4 + 19,99 \cdot x_5 + 29,99 \cdot x_6 \leq 150$$

Non superare la memoria massima può essere espresso dalla seguente relazione:

$$30 \cdot x_1 + 40 \cdot x_2 + 12 \cdot x_3 + 46 \cdot x_4 + 12 \cdot x_5 + 72 \cdot x_6 \leq 150$$

e sono vincoli lineari.

Volere i giochi che più ci piacciono si esprime nel seguente modo:

$$\max(2 \cdot x_1 + 5 \cdot x_2 + 2 \cdot x_3 + 5 \cdot x_4 + 3 \cdot x_5 + 5 \cdot x_6)$$

che è una funzione obiettivo lineare.

Volere i giochi che più ci piacciono, avendo 200 euro di budget e 100GB di spazio corrisponde a:

$$\max(2 \cdot x_1 + 5 \cdot x_2 + 2 \cdot x_3 + 5 \cdot x_4 + 3 \cdot x_5 + 5 \cdot x_6)$$

**Si tratta, quindi, di un problema di programmazione lineare a variabili binarie, che sono un caso particolare di variabili intere. Un modello di ottimizzazione di questo tipo prende il nome di Problema dello zaino (Knapsack)**

altri esempi sulle slide

**Esempio 12.** Partendo dai dati di vendita di avatar capire se Endgame supererà gli incassi di avatar, sapendo gli incassi dei primi giorni.

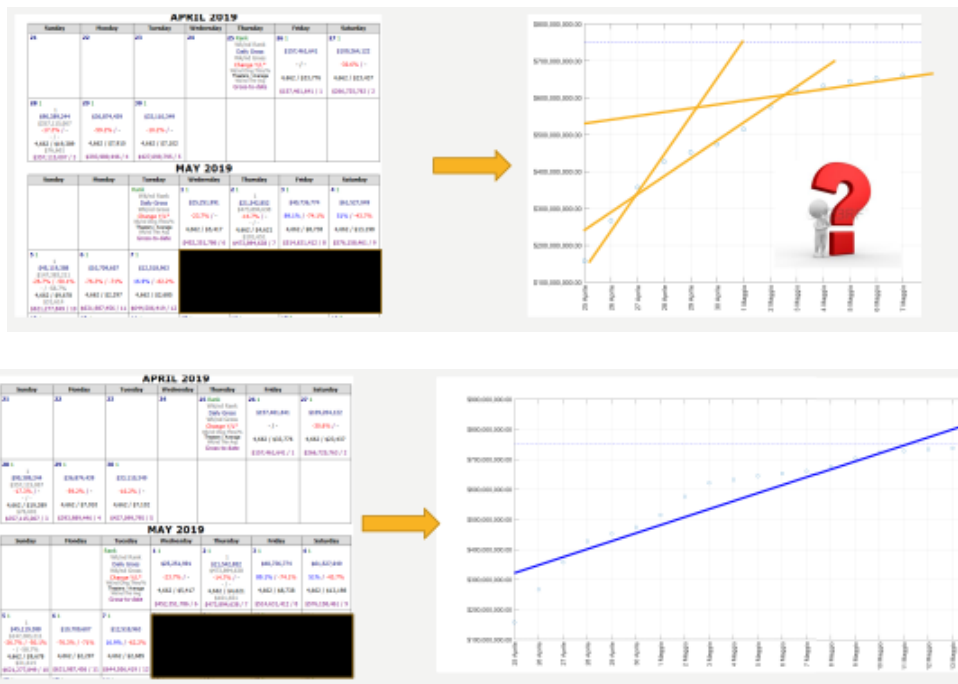
Vogliamo costruire una retta di regressione lineare che interpoli i dati:

$$y = ax + b, \quad a, b \in \mathbb{R}$$

Bisogna calcolare la retta di regressione corrisponde al seguente problema di ottimizzazione non vincolata:

$$\min_{a,b} \left[ \frac{1}{2} \sum_{i=1}^n (y_i - ax_i - b)^2 \right]$$

e quindi si ha funzione obiettivo non-lineare con  $a, b$  variabili continue. Cerco quindi la retta di regressione partendo dai primi dati e ipotizzo una previsione e cerco  $a$  e  $b$  :



Questo è un problema di ottimizzazione non vincolata e non lineare

altri esempi sulle slide



## 2.2 Soluzione Grafica

Consideriamo un problema di programmazione lineare. Abbiamo la seguente funzione obiettivo:

$$\text{opt } f(x) = c^T x$$

con i seguenti vincoli lineari:

$$X = x \in \mathbb{R}^n : g_i(x) \{\leq, =, \geq\} 0, i = 1 \dots m$$

con:

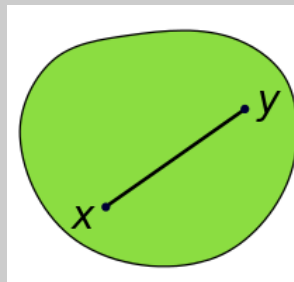
$$g_i(x) = a_i^T x - b_i, a \in \mathbb{R}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, \text{opt} = \{\min, \max\}$$

I problemi di ottimizzazione reali si presentano in forma PL se sono verificate le seguenti ipotesi:

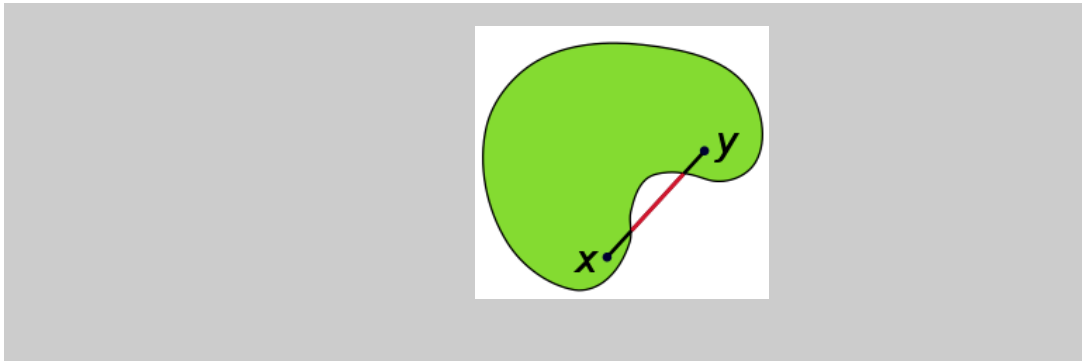
- **proporzionalità:** il contributo di ogni variabile decisionale al valore della funzione obiettivo è proporzionale rispetto al valore assunto dalla variabile stessa
- **additività:** ogni funzione è la somma dei contributi delle variabili
- **continuità:** qualunque valore delle variabili in  $\mathbb{R}^n$  è accettabile

Diamo due definizioni:

- in uno spazio euclideo un **insieme convesso** è un insieme nel quale, per ogni coppia di punti, il segmento che li congiunge è interamente contenuto nell'insieme



- in uno spazio euclideo un **insieme non convesso** è un insieme nel quale, per ogni coppia di punti, il segmento che li congiunge non è interamente contenuto nell'insieme

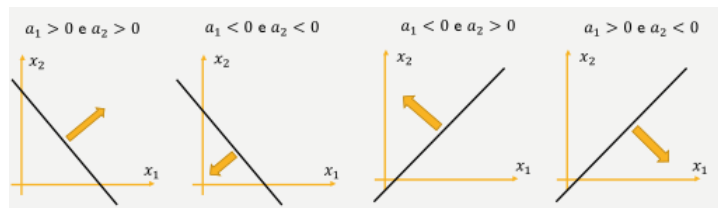


Tornando al problema iniziale iniziamo a studiare il caso **2D**. In questo caso si ha che:

- i vincoli  $g_i(x)$  possono essere rette (se  $g_i(x) = 0$ ) o semipiani (se  $g_i(x) \neq 0$ )
- la regione ammissibile  $X$  risulta essere un sottoinsieme convesso del piano cartesiano
- la funzione obiettivo  $z = c_1x_1 + c_2x_2$  è un piano nello spazio  $\mathbb{R}^3$

assumiamo inoltre, senza perdere generalità, un vincolo di non negatività delle variabili, ovvero  $x_1, x_2 \geq 0$ .

Un vincolo del tipo  $a_1x_1 + a_2x_2 = b_1$  è una retta nel piano, con l'inclinazione che è perpendicolare al vettore  $v = (a_1, a_2)$ :



Questo viene detto **vincolo retta**.

Studiamo ora il **vincolo semipiano**, che è un vincolo del tipo  $a_1x_1 + a_2x_2 \leq b_1$  (che è appunto un semipiano).

Per disegnare il semipiano disegniamo prima la retta associata  $a_1x_1 + a_2x_2 = b_1$ . Scegliamo poi un punto non appartenente a tale retta e:

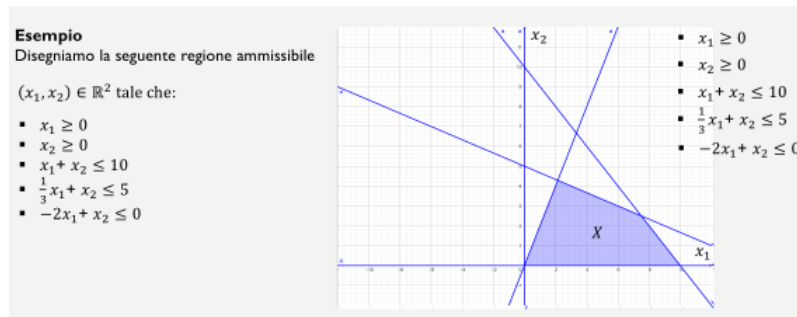
- se il punto verifica la disuguaglianza allora scegliamo il semipiano che lo contiene

- altrimenti scegliamo un altro semipiano

**Esempio 13.** Si ha  $x_1 + x_2 \leq 2$ . Disegniamo quindi  $x_1 + x_2 = 2$ , scegliamo il punto  $(0,0)$  e, essendo  $0+0 \leq 2$  abbiamo che il punto soddisfa la disequazione e quindi il semipiano contiene  $(0,0)$

Un vincolo del tipo  $a_1x_1 + a_2x_2 \geq b_1$  è uguale a  $-a_1x_1 - a_2x_2 \leq -b_1$

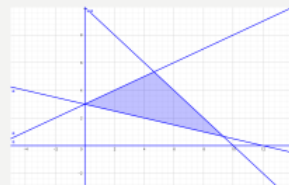
La regione ammissibile  $X$  è data dall'intersezione dei vari vincoli (rette e semipiani) e quindi, dal punto di vista geometrico corrisponde ad un **poliedro convesso** in  $\mathbb{R}^2$ , e può essere limitata (**politopo**) o illimitata



La seguente regione ammissibile...

- $x_1 \geq 0$
- $x_2 \geq 0$
- $x_1 + x_2 \leq 10$
- $\frac{1}{2}x_1 - x_2 \geq 3$
- $\frac{1}{4}x_1 + x_2 \geq 3$

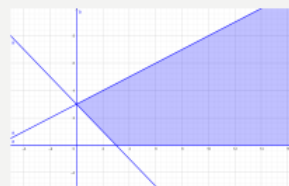
...è un triangolo (politopo con 3 vertici)



La seguente regione ammissibile...

- $x_1 \geq 0$
- $x_2 \geq 0$
- $\frac{1}{2}x_1 - x_2 \geq -3$
- $x_1 + x_2 \geq 3$

...è un poliedro illimitato con 2 vertici



**Esempio 14.** Consideriamo il seguente problema di ottimizzazione:

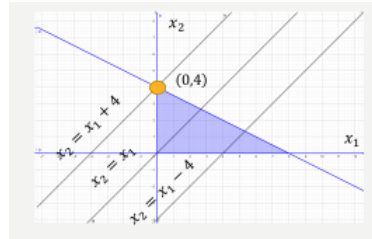
$$z = -x_1 + x_2$$

con i vincoli:

$$x_1 + x_2 \leq 4, x_1 \geq 0, x_2 \geq 0$$

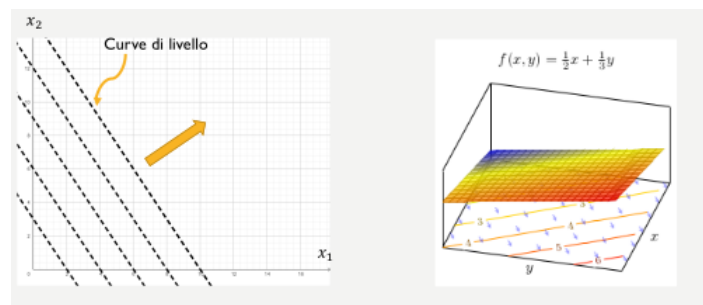
Riscriviamo la funzione obiettivo come  $x_2 = x_1 + z$ , che rappresenta un fascio di rette parallele al variare di  $z$ , che all'aumentare si spostano verso il punto

$(0,4)$ , oltre il quale si esce dalla regione ammissibile. Quindi la soluzione ottima è il punto  $(0,4)$  che rappresenta un vertice della regione ammissibile (poligono convesso):



Vediamo il caso 3D.

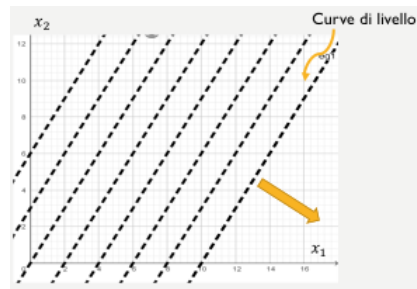
la funzione obiettivo  $z = c_1x_1 + c_2x_2$  rappresenta un piano nello **spazio 3D** passante per l'origine e, in particolare, se  $c_1, c_2 > 0$  allora il piano cresce da sinistra verso destra dal basso verso l'alto:



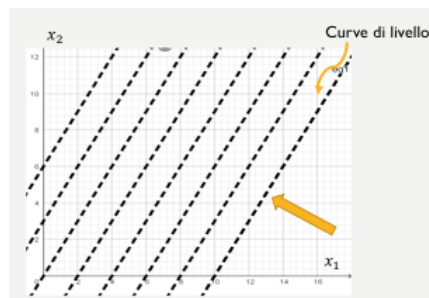
La funzione obiettivo  $z = c_1x_1 + c_2x_2$  rappresenta un piano nello spazio 3D passante per l'origine, in particolare, se  $c_1, c_2 < 0$ , allora il piano cresce da destra verso sinistra dall'alto verso il basso:



Invece se  $c_1 > 0$  e  $c_2 < 0$  allora il piano cresce da sinistra verso destra, dall'alto verso il basso:



Invece se  $c_1 < 0$  e  $c_2 > 0$  allora il piano cresce da destra verso sinistra, dal basso verso l'alto:



È possibile individuare il punto di massimo (minimo) del problema di programmazione lineare (PL) seguendo la direzione di crescita (decrecita) della funzione obiettivo all'interno della regione ammissibile.

Si possono avere 3 situazioni:

1. il problema PL ammette **un'unica soluzione ottima** in un vertice del poligono convesso che delimita la regione ammissibile
2. il problema PL ammette **infinite soluzioni ottime** in un lato del poligono convesso che delimita la regione ammissibile se la direzione di decrescita è perpendicolare ad un lato del poligono
3. il problema PL **non ammette soluzione** perché la regione ammissibile è illimitata e la funzione obiettivo è illimitata superiormente (se è di massimizzazione) o illimitata inferiormente (se è di minimizzazione)



**Esempio 15.** La società *WYNDOR GLASS Co.* realizza prodotti in vetro di alta qualità tra cui finestre e porte in vetro. Essa possiede tre stabilimenti. Lo Stabilimento 1 produce telai in alluminio, lo Stabilimento 2 produce telai in legno, mentre lo Stabilimento 3 produce il vetro e assembla i prodotti. I dirigenti della società hanno deciso di rinnovare la linea di produzione con il lancio di due nuovi prodotti:

- *Prodotto 1: una porta in vetro con intelaiatura in alluminio*
- *Prodotto 2: una finestra con intelaiatura in legno*

Il guadagno stimato per i due nuovi prodotti è di 3000\$ e 5000\$ rispettivamente per lotto. Nella seguente tabella è rappresentato il tempo di produzione richiesto ed il tempo massimo disponibile per lotto nei 3 stabilimenti per i due nuovi prodotti:

Stabilimento	Tempo di produzione richiesto per il Prodotto 1	Tempo di produzione richiesto per il Prodotto 2	Tempo di produzione disponibile
1	1	0	4
2	0	2	12
3	3	2	18

Risolviemo usando il metodo matematico.

Indichiamo con  $x_1$  e  $x_2$  il numero di lotti per il primo ed il secondo prodotto rispettivamente (**variabili decisionali**). Queste due variabili chiaramente non negative e quindi  $x_1, x_2 \in \mathbb{R}^+$  (anche se i lotti sono quantità intere in questo caso si può pensare le due quantità come tassi di produzione nel tempo e quindi come delle quantità continue) Quindi si ha:

- il tempo richiesto dallo Stabilimento 1 per produrre il prodotto 1 è dato da  $1 \cdot x_1$
- il tempo richiesto dallo Stabilimento 2 per produrre il prodotto 1 è dato da  $1 \cdot x_2$
- il tempo richiesto dallo Stabilimento 3 per produrre il prodotto 1 e 2 è dato da  $3 \cdot x_1 + 2 \cdot x_2$

aggiungiamo i vincoli sulle disponibilità massime di produzione:

$$1 \cdot x_1 \leq 4$$

$$2 \cdot x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

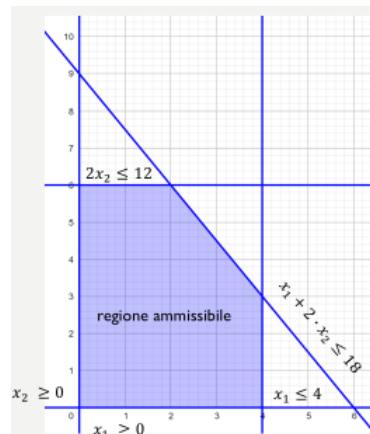
che si aggiungono a:

$$x_1, x_2 \geq 0$$

con la funzione obiettivo, per massimizzare i profitti, che è:

$$\max_{x_1, x_2} z = (3000 \cdot x_1 + 5000 \cdot x_2) = (3 \cdot x_1 + 5 \cdot x_2)$$

ho ottenuto quindi il mio modello matematico. **Si tratta, quindi, di un problema di programmazione lineare a variabili continue con 3 vincoli (più 2 vincoli di non negatività) con la regione ammissibile che è un poligono convesso limitato**



Si ha che la funzione obiettivo crescente lungo la direzione positiva degli assi  $x_1$  e  $x_2$  e si ottiene che il valore che massimizza la funzione obiettivo (con  $z > 0$ ) è  $(2, 6)$  dove  $z = 36$ .

**Essendo la regione ammissibile finita e non vuota, per qualsiasi valore del profitto dei due prodotti si ottiene sempre almeno una soluzione ottima del problema (potrebbero essere infinite):**



### 2.2.1 esercizi

#### Esempio 16. test:

*Nell'intorno di un atomo l'energia di interazione tra l'atomo stesso e un altro atomo sonda che gli viene avvicinato, è data dalla formula*

$$E = \frac{A}{r^{12}} - \frac{B}{r^6}$$

*dove  $A$  e  $B$  sono parametri caratteristici dell'atomo mentre  $r$  rappresenta la distanza Euclidea tra l'atomo e la sonda. È data una configurazione tridimensionale di alcuni atomi, supposti puntiformi e si vuole trovare il punto di minima energia a cui la sonda (anch'essa considerata puntiforme) tende a stabilizzarsi per effetto delle interazioni con gli atomi stessi. Sono dati 5 atomi, posizionati come in Tabella e con i relativi valori di  $A$  e  $B$ . Si vuole determinare la posizione ottimale dell'atomo sonda al fine di minimizzare l'energia complessiva*

Atomo	x	y	z	A	B
1	3.2	2.5	4.8	1.0	200



2	6.6	1.2	4.5	3.0	250
3	8.5	7.8	1.5	1.5	100
4	0.8	5.1	5.6	0.5	400
5	4.1	9.3	0.9	1.7	500

*Si descriva il modello matematico per esprimere la ricerca della soluzione di tale problema **soluzione:***



## 2.3 Simplexso

Consideriamo nuovamente un generico problema di programmazione lineare (PL) con la seguente funzione obiettivo lineare:

$$\text{opt } f(x) = C^T x$$

con i seguenti vincoli lineari:

$$X = x \in \mathbb{R}^n : g_i(x) \{\leq, =, \geq\} 0, i = 1 \dots m$$

che può essere riscritto nella seguente **forma standard** (che è un problema di massimo):

$$\max c^T x$$

tale che  $Ax = b, x \geq 0$ , con:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}, A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, c = [c_1 \quad \cdots \quad c_n]$$

rispettivamente *vettore colonna*  $n \times 1$ , *vettore colonna*  $m \times 1$ , *matrice*  $m \times n$  e *vettore riga*  $m \times 1$ .

Per convenzione  $b \geq 0$  quindi le  $b_i$  sono positive.

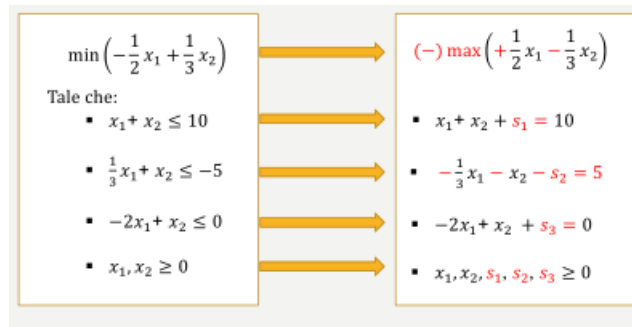
**Ogni problema PL può essere riscritto in forma standard**, infatti:

- $\min c^T x \rightarrow -\max(-c^T x)$  quindi esprimo un problema di minimo come un problema di massimo
- $a_i^T \leq b_i \rightarrow a_i^T + s_i = b_i$ , con  $s_i \geq 0$  che è una nuova variabile che prende il nome di **variabile di slack**. Quindi esprimo un vincolo di disuguaglianza come uno di uguaglianza, introducendo una nuova variabile
- $a_i^T \geq b_i \rightarrow a_i^T - s_i = b_i$ , con  $s_i \geq 0$  che è una nuova variabile che prende il nome di **variabile di surplus**. Quindi esprimo un vincolo di disuguaglianza come uno di uguaglianza, introducendo una nuova variabile
- se  $x_i \in \mathbb{R}$  (variabile intera) allora si possono definire due nuove variabili  $u_i$  e  $v_i$  tali che:

$$\begin{aligned} - u_i &\geq 0 \\ - v_i &\geq 0 \end{aligned}$$

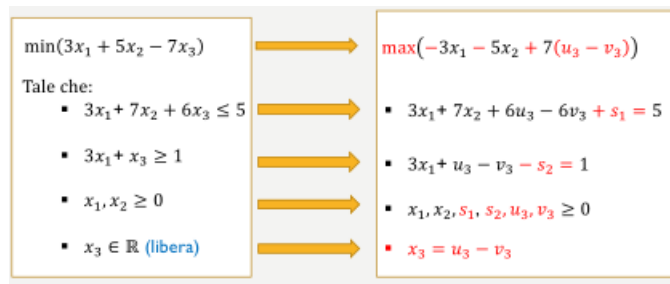
$$- x_i = u_i + v_i$$

Quindi esprimo una variabile libera attraverso una combinazione di variabili non negative, inoltre un'altra possibilità consiste nel ricavare l'espressione di tale variabile da uno dei vincoli del problema, eliminare tale vincolo del modello e sostituire tale espressione laddove compaia la variabile libera.



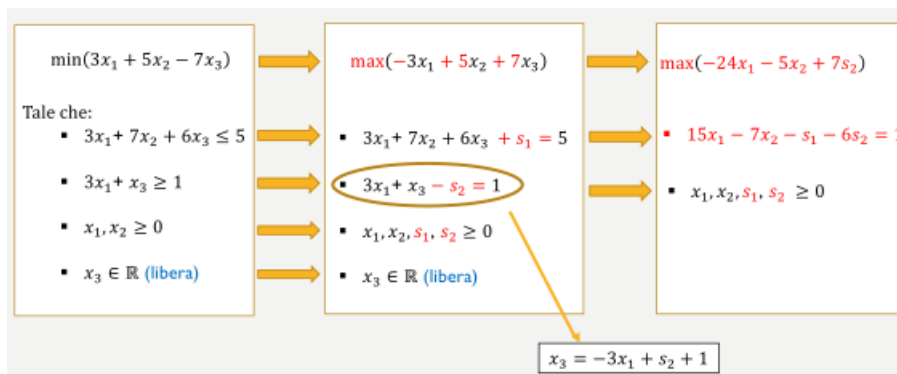
### Esempio 17.

Notiamo che passiamo da un problema con 2 variabili ad uno con 5 variabili

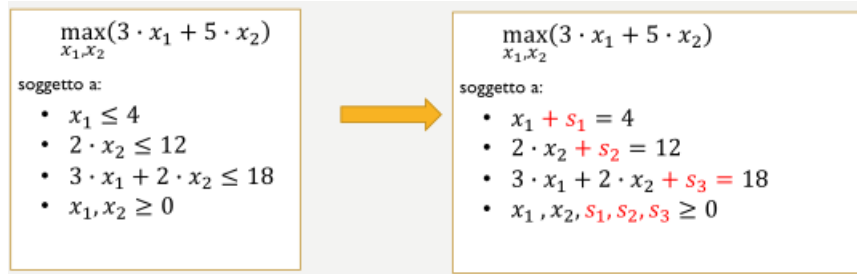


### Esempio 18.

Notiamo che passiamo da un problema con 3 variabili ad uno con 6 variabili. Inoltre, essendop  $x_3$  libera la sostituiamo con  $x_3 = u_3 - v_3$ . Possiamo anche ricavare  $x_3$  dal secondo vincolo una volta introdotta la variabile di surplus, ottenendo  $x_3 = -3x_1 + s_2 + 1$ :



**Esempio 19.** Riprendiamo il modello matematico relativo alla società Wyndor Glass Co.:



ogni sistema di vincoli della forma  $Ax \leq b$  viene trasformato nel sistema:

$$[A \quad I] [xx_s] = b$$

con  $I$  che è una matrice identità di dimensione  $m \times m$  e  $x_s$  sono  $m$  variabili di slack.

Se il numero di vincoli  $m$  di  $A$  è strettamente maggiore del numero di variabili  $n$ , allora:

- se il rango di  $A$  è maggiore di  $n$  il sistema  $Ax = b$  non ha soluzione, come conseguenza di Rouchè-Capelli
- se il rango  $A$  è  $n$  allora ho  $m - n$  vincoli ridondanti
- se il numero di righe  $A$  è uguale al numero di colonne (matrice quadrata) allora:
  - se  $\det(A) \neq 0$  esiste un'unica soluzione del problema  $Ax = b$ . Se tale soluzione ha tutte le componenti non-negative allora è anche la soluzione ottimale del problema. Altrimenti il problema non ha soluzione
  - se  $\det(A) = 0$  il sistema non ammette soluzioni o uno dei vincoli è ridondante

## 2.4 Metodo del Simplexso

Questo metodo è stato ideato da G. Dantzig nel 1947 ed è considerato uno dei dieci migliori algoritmi del 900.

Questo algoritmo ha:

- nel caso medio tempo computazionale lineare rispetto al numero delle variabili

- nel caso peggiore ha tempo computazionale esponenziale

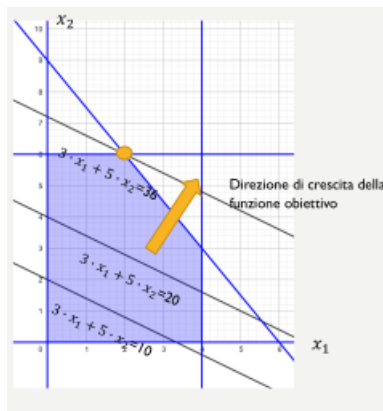
È comunque uno dei più efficienti algoritmi per risolvere un problema PL. Partiamo dal solito modello matematico relativo alla società WYNDOR GLASS & Co:

$$\max x_1, x_2 z = (3x_1 + 5x_2)$$

con:

$$x_1 \leq 4, 2x_2 \leq 12, 3x_1 + 2x_2 \leq 18, x_1 \geq 0, x_2 \geq 0$$

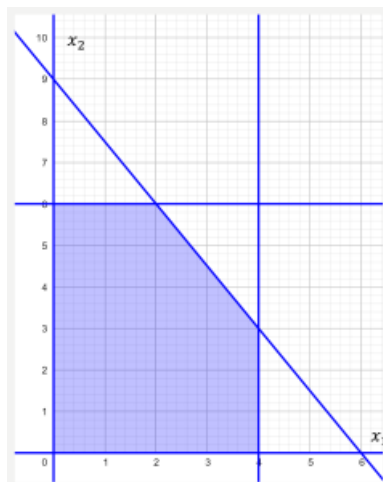
che è un PL a variabili continue con 5 vincoli, 3 funzionali e 2 di non negatività. Il valore che massimizza è (2,6)



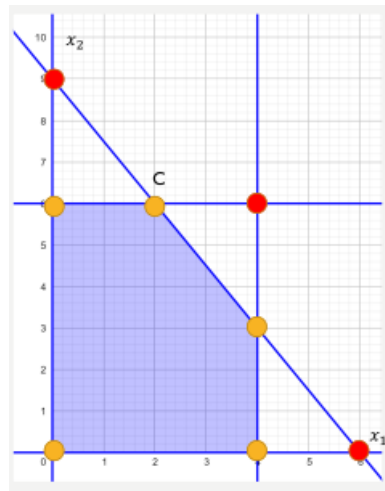
Partiamo ad analizzare i concetti geometrici. Per un vincolo, l'equazione della frontiera è ottenuta sostituendo i  $\geq$  e  $\leq$  con  $=$ , otteniamo quindi:

$$x_1 = 4, 2x_2 = 12, 3x_1 + 2x_2 = 18, x_1 = 0, x_2 = 0$$

Ricordiamo che in due dimensioni una frontiera corrisponde ad una retta mentre nello spazio n-dimensionale la frontiera corrisponde ad un iperpiano

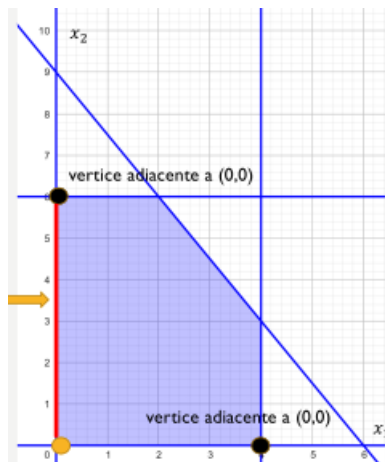


Un **vertice ammissibile** è una soluzione ammissibile che non è presente su nessun segmento che congiunge altre due soluzioni ammissibili. *i vertici di un poligono convesso non possono essere mai ottenuti da una combinazione convessa di altri 2 punti del poligono.* In 2 dimensioni ogni vertice ammissibile è l'intersezione delle frontiere di 2 vincoli mentre in  $n$  dimensioni ogni vertice ammissibile è l'intersezione delle frontiere di  $n$  vincoli. Nell'esempio sopra si hanno 8 vertici di cui 5 ammissibili e 3 no:



e il vertice C è ammissibile ed è dato dall'intersezione della frontiera  $2x_1 = 12$  con  $3x_1 + 2x_2 = 18$ .

In un problema PL 2D, due vertici si dicono **adiacenti** se sono vertici di un medesimo segmento (la frontiera di uno dei vincoli) della regione ammissibile. *Per un problema di PL con  $n$  variabili, due vertici si dicono adiacenti se condividono le frontiere di  $n - 1$  vincoli. Essi sono collegati attraverso un segmento, chiamato **spigolo** della regione ammissibile*



Dove la linea rossa è la frontiera di  $x \geq 0$ .

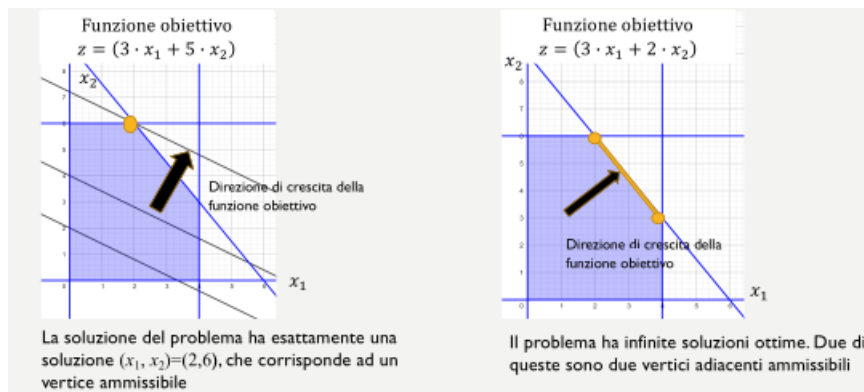
Si hanno i seguenti vertici adiacenti:

Vertici	Vertici Adiacenti
(0,0)	(0,6) ; (4,0)
(0,6)	(2,6) ; (0,0)
(2,6)	(4,3) ; (0,6)
(4,3)	(4,0) ; (2,6)
(4,0)	(0,0) ; (4,3)

Vediamo ora il **teorema fondamentale della PL**:

**Teorema 1.** *Dato un generico problema PL:*

1. *se esiste una sola soluzione ottima (per esempio se  $X$  è non vuoto e limitato), allora deve essere un vertice ammissibile*
2. *se esistono soluzioni ottime multiple e la regione ammissibile è limitata, allora almeno due soluzioni ottime sono vertici adiacenti ammissibili*



**Si ha che il numero di vertici ammissibili è finito.**

Nel caso in esame, questo è una conseguenza del fatto che ogni vertice è l'intersezione di 2 delle 5 frontiere del problema e, quindi, il numero massimo di combinazioni differenti di 5 equazioni considerate 2 alla volta è dato da:

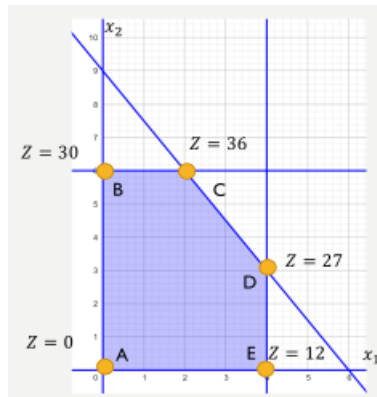
$$\binom{5}{2} = 10$$

e nel nostro caso ne abbiamo 8 di cui 5 ammissibili.

### 2.4.1 Test di Ottimalità

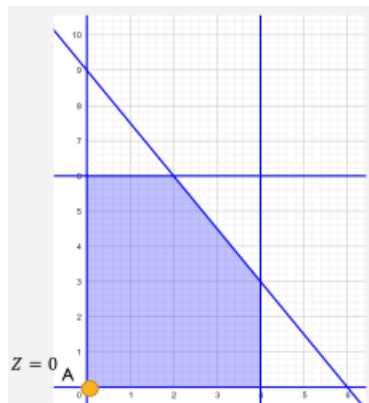
Si consideri un qualunque problema di PL che possieda almeno una soluzione ottima (ad esempio se  $X$  è non vuoto e limitato). Se un vertice ammissibile non ha nessun vertice adiacente migliore (in termini di funzione obiettivo), allora tale vertice è la soluzione ottima.

Riprendendo l'esempio solito il vertice  $A$  ha i due vertici adiacenti  $B$  e  $E$  che sono migliori e quindi  $A$  non è migliore.  $C$  ha gli adiacenti  $B$  e  $D$  peggiori e quindi è la soluzione ottima del problema:

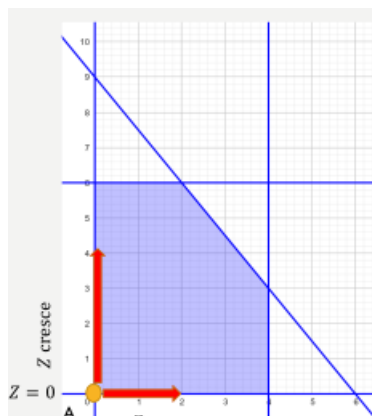


Vediamo ora una simulazione del metodo del simplexso dal punto di vista geometrico:

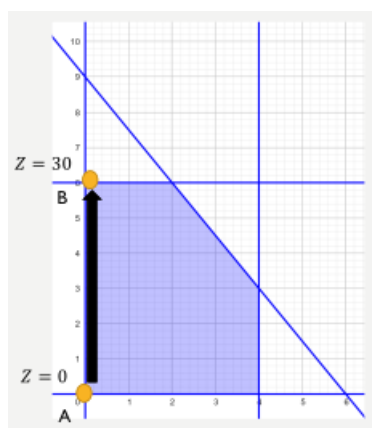
1. **inizializzazione**, dove si sceglie  $(0,0)$  come vertice iniziale:



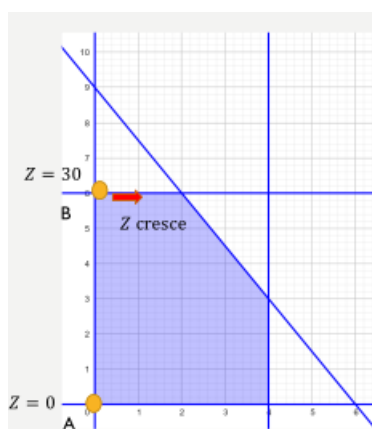
2. **test di ottimalità**:  $(0,0)$  non è una soluzione ottimale ma esistono soluzioni adiacenti migliori



3. **prima iterazione**, mi sposto su  $(0,6)$

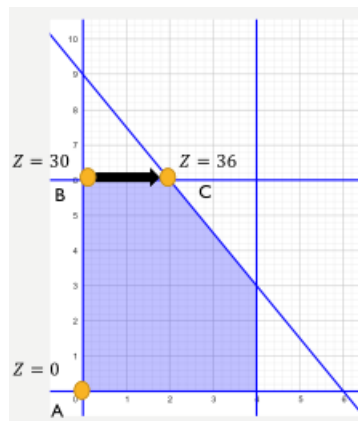


4. **test di ottimalità**:  $(0,6)$  non è soluzione ottimale. Esistono soluzioni adiacenti migliori

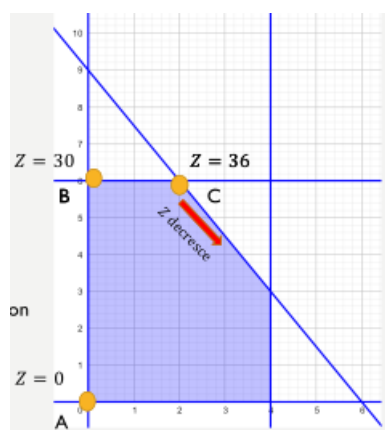




5. **seconda iterazione**, mi sposto su (2,6)

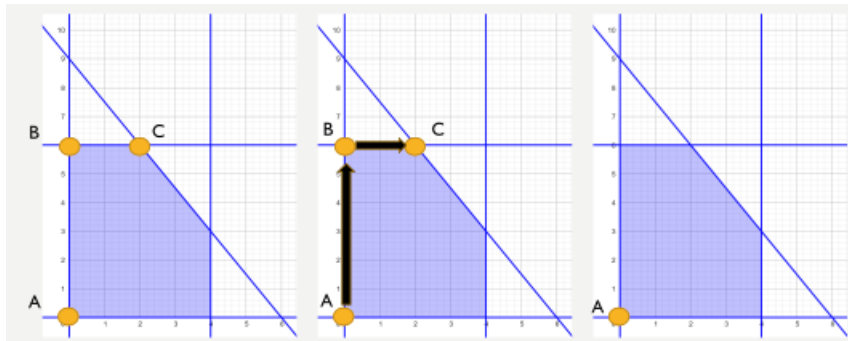


6. **test di ottimalità**, (2,6) è la soluzione ottimale in quanto non esistono vertici migliori adiacenti

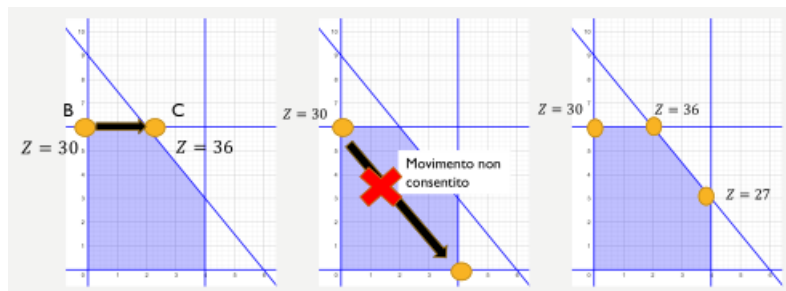


7. **fine procedura**, termina dopo solo 2 iterazioni visitando 3 vertici su 5

Questo metodo è iterativo e si concentra solo sui vertici. Se possibile, si sceglie come vertice iniziale l'origine:



Un'iterazione del metodo del simplexso corrisponde a muoversi dal vertice corrente ad uno adiacente migliore. Il metodo del simplexso si ferma nel momento in cui non vi sono più vertici adiacenti migliori.



Vediamo ora i **concetti algebrici del simplexso**.

a procedure algebrica per il metodo del simplexso, prevede di scrivere il problema PL in forma standard. Considerando il solito problema si avrebbe:

$\max_{x_1, x_2} (3 \cdot x_1 + 5 \cdot x_2)$ <p>soggetto a:</p> <ul style="list-style-type: none"> <li>• <math>2 \cdot x_2 \leq 12</math></li> <li>• <math>3 \cdot x_1 + 2 \cdot x_2 \leq 18</math></li> <li>• <math>x_1 \leq 4</math></li> <li>• <math>x_1, x_2 \geq 0</math></li> </ul>		$\max_{x_1, x_2} (3 \cdot x_1 + 5 \cdot x_2)$ <p>soggetto a:</p> <ul style="list-style-type: none"> <li>• <math>2 \cdot x_2 + s_1 = 12</math></li> <li>• <math>3 \cdot x_1 + 2 \cdot x_2 + s_2 = 18</math></li> <li>• <math>x_1 + s_3 = 4</math></li> <li>• <math>x_1, x_2, s_1, s_2, s_3 \geq 0</math></li> </ul>
--	--	--

Questa nuova forma si chiama **forma aumentata**, in quanto sin introducono le 3 variabili  $s_1, s_2, s_3$ .

Abbiamo quindi ottenuto una forma del tipo:

$$Ax = b$$

dove  $A$  è una matrice  $3 \times 5$  e si passa da 3 a 5 variabili (2 originali più 3 di *slack*).

$$A = \begin{bmatrix} 0 & 2 & 1 & 0 & 0 \\ 3 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 12 \\ 18 \\ 4 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

**Definizione 3.** Una **soluzione aumentata** è una soluzione per la quale alle variabili originali sono aggiunte le variabili di *slack* e corrisponde ad una soluzione del sistema  $Ax = b$ , che è sempre risolubile fissando due valori di variabili.

**Esempio 20.** Se consideriamo la soluzione 1,1 nella formulazione originale del problema, la soluzione aumentata è data da

$$\begin{array}{ll} \bullet & 2 \cdot 1 + s_1 = 12 \\ \bullet & 3 \cdot 1 + 2 \cdot 1 + s_2 = 18 \\ \bullet & 1 + s_3 = 4 \end{array} \quad \longrightarrow \quad \begin{array}{ll} \bullet & s_1 = 10 \\ \bullet & s_2 = 13 \\ \bullet & s_3 = 3 \end{array}$$

Ovvero la soluzione aumentata corrisponde a  $(1, 1, 10, 13, 3)$

Se considero  $(0, 8)$ :

$$\begin{array}{ll} 2 \cdot 8 + s_1 = 12 \\ 3 \cdot 0 + 2 \cdot 8 + s_2 = 18 \\ 0 + s_3 = 4 \end{array} \quad \longrightarrow \quad \begin{array}{ll} \bullet & s_1 = -6 \\ \bullet & s_2 = 0 \\ \bullet & s_3 = 4 \end{array}$$

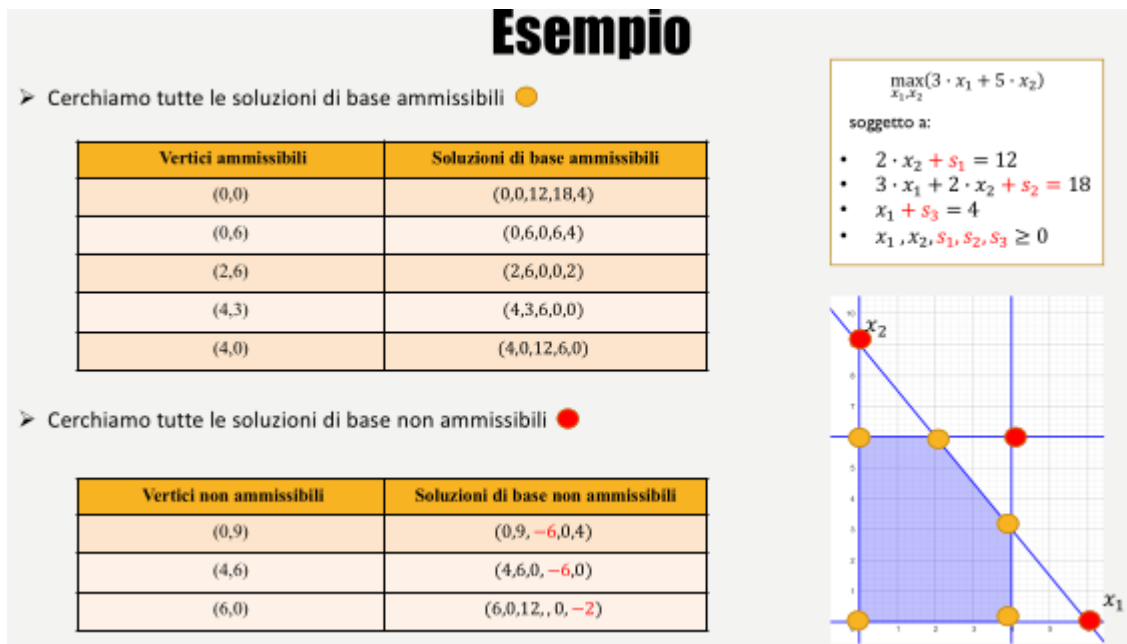
Ovvero la soluzione aumentata corrisponde a  $(0, 8, 6, 0, 4)$  (fuori da  $X$ )

Se considero  $(4, 3)$ :

$$\begin{array}{ll} 2 \cdot 3 + s_1 = 12 \\ 3 \cdot 4 + 2 \cdot 3 + s_2 = 18 \\ 4 + s_3 = 4 \end{array} \quad \longrightarrow \quad \begin{array}{ll} \bullet & s_1 = 6 \\ \bullet & s_2 = 0 \\ \bullet & s_3 = 0 \end{array}$$

**Definizione 4.** Una **soluzione di base** ammissibile è un vertice ammissibile a cui sono aggiunti i corrispondenti valori della variabili di *slack* e corrisponde ad una soluzione di  $Ax = b$  tale che  $x \geq 0$

**Esempio 21.** consideriamo il vertice  $(4, 3)$  to è ammissibile e la soluzione di base ammissibile corrispondente è  $(4, 3, 6, 0, 0)$ . Se considero  $(4, 3)$  to non è ammissibile e la soluzione di base ammissibile corrispondente è  $(0, 9, -6, 0, 0)$  non è ammissibile, infatti viene violato il vincolo di non negatività della variabile di slack  $s_1$ .



Una soluzione di base ha  $m$  (numero dei vincoli del problema) **variabili di base** ( $\leq 0$ ) e le rimanenti  $nm$  ( $n$  numero di variabili della forma aumentata) sono chiamate **variabili non di base** ( $= 0$ ). Si ha che:

$$m \leq n$$

In una soluzione di base le variabili non di base sono nulle e i valori delle variabili di base sono ottenuti risolvendo il sistema di  $m$  equazioni.

**Se una delle variabili di base è nulla, si parla di *soluzione di base degenera***

**Definizione 5.** In un problema PL due soluzioni di base si dicono *adiacenti* se tutte le variabili non di base tranne una sono uguali e questo implica che e variabili di base tranne una sono le stesse, anche se con valori numerici differenti

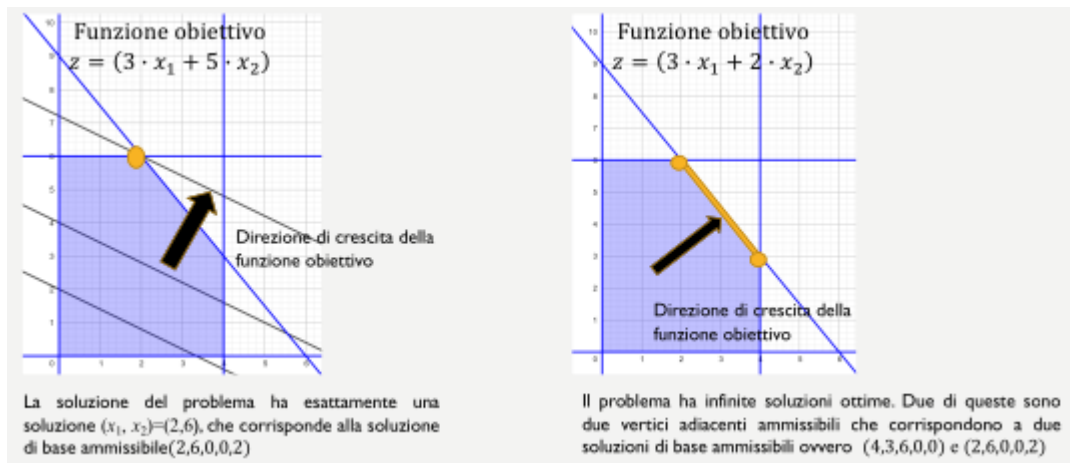
### Esempio

- Consideriamo la soluzione di base associata a (0,0) ovvero (0,0,12,18,4)  
(variabili in base  $s_1, s_2$  e  $s_3$ , non in base  $x_1$  e  $x_2$ )
- Una soluzione di base adiacente è data, ad esempio considerando  $x_2$  come variabile di base al posto di  $s_1$   
(variabili in base  $x_2, s_2$  e  $s_3$ , non in base  $s_1$  e  $x_1$ )
- Annullando  $s_1$  al posto di  $x_2$  si ottiene (quindi  $x_1 = 0$  e  $s_1 = 0$ )
 

• $2 \cdot x_2 + 0 = 12$	• $x_2 = 6$
• $3 \cdot 0 + 2 \cdot x_2 + s_2 = 18$	• $s_2 = 6$
• $0 + s_3 = 4$	• $s_3 = 4$
- La nuova soluzione di base ottenuto è (0,6,0,6,4) che corrisponde al vertice adiacente (0,6)

**Teorema 2.** Dato un generico problema PL:

1. se esiste una sola soluzione ottima (per esempio se  $X$  non è vuoto ed è illimitato) allora questa deve essere una soluzione di base ammissibile
2. se esistono soluzioni ottime multiple e la regione ammissibile è limitata, allora almeno due soluzioni ottime sono soluzioni di base adiacenti ammissibili



Per la generazione di soluzioni di base, avendo  $n$  variabili e  $m$  vincoli, si annullano  $n - m$  variabili corrispondenti al numero di *gradi di libertà*. Inoltre, in generale, il numero di combinazioni differenti considerate  $m$  alla volta è:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

Usiamo un criterio per generare le soluzioni di base, ovvero ci spostiamo lungo i confini della regione ammissibile. I coefficienti della variabili non di base indicano il **tasso di miglioramento** prodotto da un eventuale aumento del valore di tali variabili dai valori correnti. Si hanno quindi  $n - m$  tassi di miglioramento e scelgo sempre il tasso di miglioramento maggiore (se cerco il massimo).

**Se i tassi di miglioramento relativi alle variabili non in base sono tutti negativi allora la soluzione di base considerata è ottima.**

*Sulle slide esempio completo di uso dell'algoritmo.*

Si hanno due casi particolari:

1. **regola di Bland** per evitare il cycling. Per determinare la variabile uscente, nel caso in cui ci siano più variabili candidate ad uscire, viene selezionata quella con l'indice più piccolo
2. **soluzioni ottime multiple.** Ogni qual volta un problema ha più di una base ottima, almeno una delle variabili non di base ha un coefficiente nullo nella riga della funzione obiettivo

La procedura del simplexso si applica a tutti i modelli di programmazione lineare in cui la regione ammissibile è esprimibile come  $Ax \leq b$  con  $b \geq 0$  ponendo al solito:

$$\begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} x \\ x_s \end{bmatrix} = b$$

con  $I$  matrice identità  $m \times m$  e  $x_s$  che sono  $m$  variabili di stack. Scegliendo come soluzione di base iniziale

$$\begin{bmatrix} 0 \\ b \end{bmatrix}$$

ovvero ponendo a zero le variabili del problema originale, e  $x_s = b$

## 2.4.2 Forma Tabellare

È la forma più semplice per rappresentare il simplexso. Si hanno .

- i coefficienti delle variabili nella funzione obiettivo (c) cambiati di segno
- i termini noti delle equazioni (b)
- i coefficienti delle variabili nei vincoli (A)




Per esempio:

I coefficienti delle variabili nei vincoli (A)		Coefficienti delle variabili nella funzione obiettivo				
		x1	x2	x3	x4	x5
	0	-3	-5	0	0	0
	4	1	0	1	0	0
Termini noti	12	0	2	0	1	0
	18	3	2	0	0	1

Matrice A

Nella prima iterazione del metodo del simplexso individuo la variabile da far entrare in base ( $x_2$ )

	x1	x2	x3	x4	x5
0	-3	-5	0	0	0
4	1	0	1	0	0
$2^* / 2$	12	2	0	1	0
18	3	2	0	0	1

	x1	x2	x3	x4	x5
$0^* + 5 * 2^*$					
 0	-3	-5	0	0	0
4	1	0	1	0	0
 6	0	1	0	1/2	0
$3^* - 2 * 2^*$					
 18	3	2	0	0	1

Dato che il coefficiente di  $x_1$  è negativo si ha che la soluzione non è ottimale:

	x1	x2	x3	x4	x5
$0^* + 5 * 2^*$ →	3	-3	0	5/2	0
	0				
	4	1	0	1	0
	6	0	1	0	1/2
$3^* - 2 * 2^*$ →	6	3	0	-1	1

e proseguo così fino ad avere la soluzione ottimale, arrivando ad avere i coefficienti di  $x_4$  e  $x_5$  positivi:

	x1	x2	x3	x4	x5
36	0	0	0	3/2	1
2	0	0	1	1/3	-1/3
6	0	1	0	1/2	0
2	1	0	0	-1/3	1/3

### 2.4.3 Forma Matriciale

Per poter utilizzare questo metodo si parte da 2 presupposti:

1. il numero di righe della matrice  $A$  sia strettamente minore del numero di colonne ( $n < m$ )
2. il rango di  $A$  sia  $m$

È così sempre possibile scegliere tra le  $n$  colonne di  $A$  un sottoinsieme di colonne linearmente indipendenti, ovvero una sottomatrice quadrata invertibile:

$$B \in \mathbb{R}^{m \times m}$$

mentre le restanti  $m - n$  colonne formano una sottomatrice:

$$D \in \mathbb{R}^{m \times (n-m)}$$



$B$  è detta **matrice di base** mentre  $D$  **matrice non di base**.

Da una matrice  $A$  possiamo estrarre al massimo:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

ma potrebbero esserci delle  $B_i$  non invertibili.

vediamo un esempio:

$$\max \left( +\frac{1}{2}x_1 - \frac{1}{3}x_2 \right)$$

- $x_1 + x_2 + s_1 = 10$
- $-\frac{1}{3}x_1 - x_2 - s_2 = 5$
- $-2x_1 + x_2 + s_3 = 0$
- $x_1, x_2, s_1, s_2, s_3 \geq 0$

→

$$\max \left( [1/2 \quad -1/3] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right)$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1/2 & -1 & 0 & -1 & 0 \\ -2 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \geq 0$$

Otteniamo quindi che:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1/2 & -1 & 0 & -1 & 0 \\ -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 10 \\ 5 \\ 0 \end{bmatrix}$$

$$c^T = [1/2 \quad -1/3]$$

Ricaviamo una matrice di base per la matrice  $A$

$$\max \left( +\frac{1}{2}x_1 - \frac{1}{3}x_2 \right)$$

- $x_1 + x_2 + s_1 = 10$
- $-\frac{1}{3}x_1 - x_2 - s_2 = 5$
- $-2x_1 + x_2 + s_3 = 0$
- $x_1, x_2, s_1, s_2, s_3 \geq 0$

→

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1/2 & -1 & 0 & -1 & 0 \\ -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

La prima, la seconda e la terza colonna sono linearmente indipendenti

**B**

$$A = \begin{bmatrix} \boxed{1} & \boxed{1} & \boxed{1} & 0 & 0 \\ -1/2 & -1 & 0 & -1 & 0 \\ -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

la prima, la quarta e la quinta colonna sono linearmente indipendenti

$$A = \begin{bmatrix} \boxed{1} & 1 & 1 & \boxed{0} & \boxed{0} \\ -1/2 & -1 & 0 & -1 & 0 \\ -2 & 1 & 0 & 0 & \boxed{1} \end{bmatrix}$$

Quindi  $B = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & -1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$

Il numero massimo di matrici di base è  $\binom{5}{3} = 10$

Data una matrice di base  $B$  possiamo dividere il vettore  $x$  in due sotto-vettori:

1.  $x_B$  di cardinalità  $m$  con variabili dette **variabili di base**, dove l' $i$ -esimo elemento è associato all' $i$ -esima colonna di  $B$
2.  $X_D$  di cardinalità  $n - m$  con variabili dette **variabili non di base** dove l' $i$ -esimo elemento è associato all' $i$ -esima colonna di  $D$

Possiamo risolvere il sistema  $Ax = b$  come:

$$\begin{bmatrix} B & D \end{bmatrix} \begin{bmatrix} x_B \\ x_D \end{bmatrix} = b \longrightarrow Bx_B + Dx_D = b$$

Essendo  $B$  invertibile allora possiamo moltiplicare primo e secondo membro dell'equazione per  $B^{-1}$  ottenendo:

$$x_B + B^{-1}Dx_D = B^{-1}b$$

e ponendo  $x_D = 0$  si ha:

$$x_b = B^{-1}b$$

**Definizione 6.** Dato un insieme di  $m$  equazioni lineari indipendenti in  $n$  incognite che formano un sistema del tipo  $Ax = b$  sia  $B$  una qualsiasi sottomatrice  $n \times m$  non singolare di  $A$ . Ponendo tutte le  $n - m$  componenti di  $x$  non associate a  $B$  uguali a 0, la soluzione del restante insieme di equazioni è detta **soluzione di base** per il sistema  $Ax = b$  rispetto alla base  $B$ . Inoltre tutte le componenti di  $x$  associate alle colonne di  $B$  sono chiamate **variabili di base**. Le soluzioni di base corrispondono a tutti

Consideriamo un problema PL scritto in forma standard:

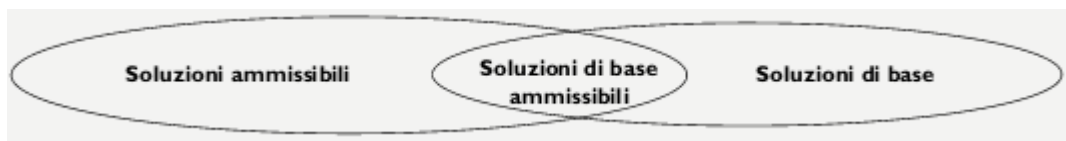
$$\max c^T x$$

con:

$$Ax = b, \quad x \geq 0$$

**Definizione 7.** Un vettore  $x$  risulta essere ammissibile se soddisfa i vincoli del problema PL, ovvero se  $Ax = b$  e  $x \geq 0$ .

Una soluzione di base  $x$  per  $Ax = b$  è detta **soluzione di base ammissibile (SBA)** se soddisfa i vincoli di non negatività. In particolare  $B^{-1}b \geq 0$



Una soluzione di base  $x$  per  $Ax = b$  è detta **soluzione di base degenera** se parte delle componenti della soluzione  $x_B$  è nulla.

Sia:

$$\begin{bmatrix} x_B \\ x_D \end{bmatrix}$$

una soluzione del problema PL in forma standard ( $\min c^T x$ ,  $Ax = b$  e  $x \geq 0$ ). Allora possiamo dividere  $c$  in due sottovettori:

1.  $c_B$  sotto-vettore dei coefficienti di costo associati alle variabili di base
2.  $c_D$  sotto-vettore dei coefficienti di costo associati alle variabili non di base

Inoltre il valore della funzione obiettivo è dato da:

$$c_B^T B^{-1} b$$

in quanto:

- se  $x$  è soluzione di base ammissibile allora  $x_D = 0$  e  $x_B = B^{-1}b$
- quindi:

$$c^T b = [c_B | c_D]^T \begin{bmatrix} x_B \\ x_D \end{bmatrix} = c_D^T x_B + c_D^T x_D = c_D^T B^{-1} b + c_D^T \cdot 0 = c_D^T B^{-1} b$$

**Teorema 3.** Dato un problema PL in forma standard ( $\min c^T x$ ,  $Ax = b$  e  $x \geq 0$ ) dove  $A$  è  $m \times n$  di rango  $m$  allora:

1. se esiste una soluzione ammissibile allora è soluzione ammissibile di base
2. se esiste una soluzione ammissibile ottima allora esiste una soluzione di base ammissibile ottima

Questo teorema comporta che:

- la risoluzione di un problema PL continuo si riduce a cercare le soluzioni di base ammissibili del problema
- il numero di soluzioni di base possibili su  $n$  variabili e  $m$  vincoli è:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

- la soluzione ottima, se esiste, corrisponderà alla soluzione di base ammissibile che minimizza il problema (soddisfacendo il test di affidabilità)

**Questo teorema è di scarsa applicabilità a causa della velocità di crescita del numero delle soluzioni di base al variare di  $n$  e  $m$**

# Capitolo 3

## Dualità

Con *dualità* ci si riferisce ad una diversa visione dei problemi. Questa tecnica limita il numero di vincoli permettendo una più facile risoluzione del problema. Il problema duale fornisce *bounds* (misure(?)) utili per risolvere problemi a variabili intere. Questa tecnica viene detta **branch and bound**. Si hanno diversi esempi di algoritmi che usano la dualità, quali il simplesso duale.

Il meccanismo consiste nell'aggiungere un secondo problema di massimo o minimo con ulteriori vincoli.

Vediamo le regole per costruire il problema duale. Innanzitutto vediamo questa tabella riassuntiva:

	<i>Problema primale</i>		<i>Problem duale</i>
<i>Obiettivo</i>	MIN	↔	MAX
	MAX	↔	MIN
<i>Vincoli/ Variabili</i>	Variabili	↔	Vincoli
	Vincoli	↔	Variabili
<i>Coefficienti</i>	Coeff. FO	↔	RHS
	RHS	↔	Coeff. FO
	"a <sub>ij</sub> "	↔	"a <sub>ji</sub> "

ricordando che si ha un problema di massimo se i vincoli sono espressi con un  $\leq$  e le variabili con un  $\geq 0$ , mentre per un min i vincoli sono  $\geq$ , con le variabili  $\geq 0$ .

A variabili canoniche, anticanoniche o libere del problema primale corrispondono vincoli canonici, anticanonici o all'uguaglianza del duale (e viceversa invertendo tipi di vincoli e tipi variabili):

<i>Problema primale (MAX)</i>	<i>Problema duale (MIN)</i>
Variabile $i$ : $x_i \geq 0$ $x_i \leq 0$ $x_i$ free	Vincolo $i$ : $i$ -esimo vincolo $\geq$ $i$ -esimo vincolo $\leq$ $i$ -esimo vincolo $=$
Vincolo $j$ : $j$ -esimo vincolo $\geq$ $j$ -esimo vincolo $\leq$ $j$ -esimo vincolo $=$	Variabile $j$ : $x_j \leq 0$ $x_j \geq 0$ $x_j$ libera

<i>Problema primale (MIN)</i>	<i>Problema duale (MAX)</i>
Variabile $i$ : $x_i \geq 0$ $x_i \leq 0$ $x_i$ free	Vincolo $i$ : $i$ -esimo vincolo $\leq$ $i$ -esimo vincolo $\geq$ $i$ -esimo vincolo $=$
Vincolo $j$ : $j$ -esimo vincolo $\geq$ $j$ -esimo vincolo $\leq$ $j$ -esimo vincolo $=$	Variabile $j$ : $x_j \geq 0$ $x_j \leq 0$ $x_j$ libera

Il duale di un problema duale è il problema primale di quel duale. Uno dei due problemi ha soluzione ottima sse lo ha anche l'altro, inoltre:

- se un problema è possibile ma illimitato, l'altro problema è inammissibile

- se un problema è inammissibile, l'altro è o inammissibile o illimitato

**Teorema 4.** *Il valore della funzione obiettivo per una qualsiasi soluzione ammissibile del problema primale (cercando il max) non può eccedere quello del problema duale (che cerca il min).*

*Il valore del problema duale è un  $\lim\_sup$  del problema primale.*

Questo è il **teorema della dualità debole**

*Dimostrazione.* avendo:

$$\max cx \mid Ax \leq b, x \geq 0$$

$$\min b^T u \mid u^T A \geq c, u \geq 0$$

si ha che:

$$cx \leq (u^T A)x = u^T (Ax) \leq u^T b$$

↓

$$cx \leq by$$

□

**Teorema 5.** *Se esiste una soluzione ottima finita il valore ottimo della funzione obiettivo del problema primale è uguale a quella del duale:*

$$cx = b^T u$$

questo è il **teorema della dualità forte**

*Dimostrazione.* Vogliamo dimostrare che se  $cx^* = b^T u^*$  allora la coppia è ottima. Partiamo con la dimostrazione per assurdo. Suppongo che  $(x^*, u^*)$  non sia ottima. Esiste quindi una soluzione  $x^\circ$  ottima per cui

$$cx^\circ \leq cx^* = b^T u^*$$

ma questo andrebbe in contrasto alla dualità debole e quindi è un assurdo □

vediamo i corollari, indicando con  $P$  il problema primale e con  $D$  il duale:

- se  $P(D)$  è illimitato allora  $D(P)$  non è ammissibile
- se  $P$  ha soluzione ottima finita la ha anche  $D$  e viceversa
- se  $D(P)$  non è ammissibile allora  $P(D)$  è illimitato o non ammissibile

### 3.0.1 condizioni di complementarità

Considero il solito PL:

$$\max cx \mid Ax \leq b, x \geq 0$$

Se questo vincolo non è attivo per una soluzione ottima  $x^*$  la corrispondente variabile duale sarà nulla in ogni soluzione ottima del problema duale:

$$y_i^*(b_i - \sum_{j=1}^n a_{i,j}x_j^*) = 0, \quad i = 1 \dots m$$

inoltre si ha che:

- se  $y_i^* > 0$  allora  $\sum_{j=1}^n a_{i,j}x_j^* = b_i$
- se  $\sum_{j=1}^n a_{i,j}x_j^* > b_i$  allora  $y_i^* = 0$

quindi per:

$$\begin{aligned} \max cx \mid Ax \leq b, x \geq 0 \\ \min b^T u \mid u^T A \geq c, u \geq 0 \end{aligned}$$

si ha:

- $u_j^{*T}(b_j - A_j x^*) = 0, \quad \forall j$
- $(u^{*T} A_i - c_i)x_i^* = 0, \quad \forall i$

Nel complesso si ha che: **il vettore delle variabili duali ottime per un problema di PL è  $c_b b^{-1}$  ed è visualizzabile se si usa il tableau:**

		$\mathbf{u}$		$c_B B^{-1}I - c = c_B B^{-1}$		
		x1	x2	x3	x4	x5
36		0	0	0	3/2	1
$B^{-1}b$	2	0	0	1	1/3	-1/3
	6	0	1	0	1/2	0
	2	1	0	0	-1/3	1/3



### 3.0.2 Analisi di Sensività

Ci si propone di studiare gli effetti sulla soluzione ottima in seguito alle modifiche di parametri del modello (dei vettori  $b$  o dei coefficienti  $c$ ). Prendendo un qualsiasi problema di PL si arriva, ipotizziamo di variare il coefficiente di uno dei vincoli di un certo  $\Delta c$ . Arriveremo ad un tableau finale della forma:

	x1	x2	x3	x4	x5
$c_b B^{-1} b$	$c_b B^{-1} A - c$		$c_b B^{-1}$		
$B^{-1} b$	$B^{-1} A$		$B^{-1}$		

**Ammissibilità se  $\geq 0$**

12

e il valore  $c_b B^{-1}$  viene definito **prezzo ombra**, ovvero tasso di variazione della funzione obiettivo al variare (limitato) del termine noto e corrispondono alla soluzione ottima del problema duale.

Si definisce l'**intervallo ammissibile per la base ottima** l'intervallo di valori per cui la soluzione di base ottima corrente rimane ammissibile. L'effetto su  $z$  del cambiamento di  $b_i$  dato dal prezzo ombra continua ad essere valido a condizione che i valori di  $b_i$  rimangano all'interno di questo intervallo.

Il cambiamento nel vettore risorse si rappresenta con:

$$b_i \rightarrow b_i + \delta$$

e il prezzo ombra di un vincolo può essere visto come il tasso di miglioramento della funzione obiettivo all'aumentare del termine noto di quel vincolo. Inoltre le condizioni di ottimalità non sono affette dalla variazione indicata ma basta verificare l'**ammissibilità**:

$$B^{-1}(b + \delta e_i) \geq 0 \implies B^{-1}b + b^{-1}e_i \geq 0 \implies x_b + g \geq 0$$

con  $g$  è l' $i$ -sima colonna di  $B^{-1}$ . Quindi per avere una soluzione ammissibile mi serve un  $\delta$  tale che:

$$\min_{g_i < 0} \frac{-x_i}{g_i} \geq \delta \geq \max_{g_i < 0} \frac{-x_i}{g_i}$$

E si ottiene la seguente variazione della funzione obiettivo:

$$c_B B^{-1}(b + \delta e_i) = C_B B^{-1}b + C_B B^{-1}\delta e_i$$

### 3.0.3 Regola del 100%

**Definizione 8.** *I prezzi ombra rimangono validi per la predizione dell'effetto di una modifica simultanea di più termini noti dei vincoli se questi cambiamenti non sono troppo grandi. Se la somma delle percentuali dei cambiamenti rispetto ai propri intervalli con ottimalità supera il 100% allora i prezzi ombra resteranno validi*

**Esempio 22.** Sia  $\max_{x_1, x_2} z = 3x_1 + 5x_2$  con:

$$\begin{cases} x_1 \leq 4 \\ 2x_2 \leq 12 \\ 3x_1 + 2x_2 \leq 18 \\ x_1, x_2 \geq 0 \end{cases}$$

e cambiamo  $b_2$  da 12 a 24. Si arriv al seguente tableau finale:

	x1	x2	x3	x4	x5	
36	0	0	0	3/2	1	
2	0	0	1	1/3	-1/3	
6	0	1	0	1/2	0	<b>B<sup>-1</sup></b>
2	1	0	0	-1/3	1/3	

quindi:

$$\max \quad 3x_1 + 5x_2$$

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

$$B^{-1} = \begin{bmatrix} 1 & 1/3 & -1/3 \\ 0 & 1/2 & 0 \\ 0 & -1/3 & 1/3 \end{bmatrix}$$

$$2 + 1/3\delta_2 \geq 0 \quad 1/3\delta_2 \geq -2 \quad \delta_2 \geq -6$$

$$6 + 1/2\delta_2 \geq 0 \quad 1/2\delta_2 \geq -6 \quad \delta_2 \geq -12$$

$$2 - 1/3\delta_2 \geq 0 \quad 2 \geq 1/3\delta_2 \quad \delta_2 \leq +6$$



$$-6 \leq \delta_2 \leq 6$$

$$b_2 \text{ da } 12 \text{ a } 24 \rightarrow \delta_2 = 12$$

**La Base diventa NON ammissibile !!!!**

$$b = (4, 12, 18) \quad x_B^T = (2, 6, 2)$$

Non è possibile dire nulla sul cambiamento della funzione obiettivo, occorre risolvere il problema con il nuovo dato

e  $b_3$  avrà il seguente intervallo di variazione:

$$\max \quad 3x_1 + 5x_2$$

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

$$B^{-1} = \begin{bmatrix} 1 & 1/3 & -1/3 \\ 0 & 1/2 & 0 \\ 0 & -1/3 & 1/3 \end{bmatrix}$$

$$2 - 1/3\delta_3 \geq 0 \quad 2 \geq 1/3\delta_3 \quad \delta_3 \leq 6$$

$$6 + 0\delta_3 \geq 0 \quad -\infty \leq \delta_3 \leq +\infty$$

$$2 + 1/3\delta_3 \geq 0 \quad 1/3\delta_3 \geq -2 \quad \delta_3 \geq -6$$



$$-6 \leq \delta_3 \leq 6$$

$$b = (4, 12, 18) \quad x_B^T = (2, 6, 2)$$

Supponiamo ora di voler aumentare  $b_2$  a 15 e diminuire  $b_3$  a 15. Si ha che:

$$\delta_2: 12 \rightarrow 15 \quad \text{Incremento: } 100 \left( \frac{15-12}{6} \right) = 50\%$$

$$\delta_3: 18 \rightarrow 15 \quad \text{Decremento: } 100 \left( \frac{18-15}{6} \right) = 50\%$$

proseguire

## Capitolo 4

# Programmazione Lineare Intera

Sia:

$$\text{opt } f(x) = c^T x$$

con i seguenti vincoli lineari:

$$X = x \in \mathbb{R}^n : g_i(x) \{ \leq, =, \geq \} 0, i = 1 \dots m$$

con:

$$g_i(x) = a_i^T x - b_i, a \in \mathbb{R}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, \text{opt} = \{\min, \max\}$$

Si hanno tre divisioni:

1. se  $x \in \mathbb{Z}^n$  si ha un problema di **programmazione lineare intera, *PLI***
2. se  $x \in \{0, 1\}^n$  si ha un problema di **programmazione lineare binaria, *PB***
3. se  $x \in \mathbb{R}^p \mathbb{Z}^q$ , con  $p + q = n$ ,  $p > 0$ ,  $q > 0$ , si ha un problema di **programmazione lineare mista, *PLM***

Un PLI ha un numero di soluzioni inferiori a quelle di un PL e, nel momento in cui si ha una regione ammissibile limitata, si ha per forza un numero di soluzioni ammissibili finito. Purtroppo questo non implica che sia semplice trovare le soluzioni di un PLI.

Per esempio, avendo un PLI con  $n = 20$  variabili binarie ho  $2^{20}$  soluzioni possibili.

Si cerca quindi un metodo per formulare il corrispettivo PL di un PLI, ovvero lo stesso problema ma senza i problemi di interezza. Si ha il cosiddetto **rilassamento lineare**. Vediamo due esempi.

**Esempio 23.** *Il PLI:*

$$\max z = 2x_1 + 4x_2 + 3x_3$$

*coi vincoli*

$$\begin{cases} 3x_1 + 4x_2 + 2x_3 \leq 60 \\ 2x_1 + x_2 + 2x_3 \leq 40 \\ x_1 + 3x_2 + 2x_3 \leq 80 \\ x_1, x_2, x_3 \in \mathbb{N} \end{cases}$$

*diventa il PL:*

$$\max z = 2x_1 + 4x_2 + 3x_3$$

*coi vincoli*

$$\begin{cases} 3x_1 + 4x_2 + 2x_3 \leq 60 \\ 2x_1 + x_2 + 2x_3 \leq 40 \\ x_1 + 3x_2 + 2x_3 \leq 80 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

**Esempio 24.** *Il PLI:*

$$\max z = 2x_1 + 4x_2 + 3x_3$$

*coi vincoli*

$$\begin{cases} 3x_1 + 4x_2 + 2x_3 \leq 60 \\ 2x_1 + x_2 + 2x_3 \leq 40 \\ x_1 + 3x_2 + 2x_3 \leq 80 \\ x_1, x_2, x_3 \in \{0, 1\} \end{cases}$$

*diventa il PL:*

$$\max z = 2x_1 + 4x_2 + 3x_3$$

*coi vincoli*

$$\begin{cases} 3x_1 + 4x_2 + 2x_3 \leq 60 \\ 2x_1 + x_2 + 2x_3 \leq 40 \\ x_1 + 3x_2 + 2x_3 \leq 80 \\ x_1 \leq 1, x_2 \leq 1, x_3 \leq 1 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Formalizziamo questo metodo. In generale dato un PLI si ha che:

- una variabile intera può essere “rilassata” sostituendo  $x \in \mathbb{N}$  con  $x$  variabile continua tale che  $x \geq 0$

- una variabile binaria può essere “rilassata” sostituendo  $x \in \{0, 1\}$  con  $x$  variabile continua tale che:

$$\begin{cases} x \geq 0 & \text{vincolo di non negatività} \\ x \leq 1 & \text{nuovo vincolo funzionale} \end{cases}$$

- una variabile intera non negativa con un vincolo del tipo  $x \leq M$  può essere rilassata sostituendo  $x \in \{0, M\}$  con  $x$  variabile continua tale che:

$$\begin{cases} x \geq 0 & \text{vincolo di non negatività} \\ x \leq M & \text{nuovo vincolo funzionale} \end{cases}$$

Si cerca quindi il PL di un PLI, lo si risolve e si verifica se la soluzione sia intera. In caso contrario si ottiene un **upper bound** o un **lower bound** rispettivamente per problemi di massimo o minimo.

Vediamo un esempio:

**Esempio 25.** Si ha la seguente funzione obiettivo:

$$\max z = x_2$$

con i seguenti vincoli:

$$\begin{cases} x_1 + \frac{5}{2}x_2 \leq 11 \\ x_1, x_2 \in \mathbb{N} \end{cases}$$

Si ha l'ottimo del PL (ovvero del rilassamento lineare) pari a:

$$\left(0, \frac{22}{5}\right)$$

e si ha che la soluzione approssimata del PLI è pari a:

$$(0, 4)$$

ovvero  $\frac{22}{5} = 4,4$  viene approssimato all'intero più vicino, 4

**Questo è un caso fortuito. Approssimare all'intero più vicino può portare ad una soluzione non ammissibile.**

Vediamo qui come ottenere la soluzione del PLI.

### 4.0.1 Metodo Branch ad Bound

Il metodo **Branch and Bound (B&B)** è una tecnica generale per la risoluzione di problemi di ottimizzazione combinatoria (ovvero con spazio di soluzioni finito). Il B&B è una tecnica di **enumerazione implicita**, ovvero si provano tutte le soluzioni fino a trovare l'ottima, scartandone quindi alcune a priori dimostrandone la non ottimalità. Si basa sul ***divide et impera***.

Sia:

$$\max f(x) = c^T x$$

con i seguenti vincoli lineari:

$$X = x \in \mathbb{Z}^n : g_i(x) \{ \leq, =, \geq \} 0, i = 1 \dots m$$

con:

$$g_i(x) = a_i^T x - b_i, a \in \mathbb{R}^n, b \in \mathbb{R}^m, c \in \mathbb{R}^n$$

questo viene definito come **problema completo**. Un PLI è un **sotto-problema** del PL se *presenta la medesima funzione obiettivo ma ha un sottoinsieme proprio di  $X$  come regione ammissibile*.

Formalmente sia  $z^* = f(x^*)$  la soluzione del problema completo e  $\tilde{z} = f(\tilde{x})$  la soluzione ottima di un sotto-problema. Si ha che:

$$\tilde{z} \leq z^*, e f(\tilde{x}) \leq f(x^*)$$

visto che  $\tilde{x}$  è una soluzione del problema completo.

*Per un problema di minimo si ribaltano i  $\leq$  in  $\geq$ .*

Il B&B usa tre tecniche per risolvere un generico PLI:

1. la partizione, ovvero il **branching**. Questa tecnica si usa per dividere un PLI  $P_0$  in  $k$  sotto-problemi  $P_1, \dots, P_k$  di modo che l'unione delle regioni ammissibili  $X_1 \dots, X_k$  dia la regione ammissibile del problema originario  $X$ , quindi:

$$\bigcup_{i=1}^k P_k = P_0$$

Si genera quindi un **albero delle soluzioni** che ha come radice  $P_0$  e che può diramarsi iterativamente. Si ha che l'unione delle foglie è anche il problema  $p_0$ , in base ad un ragionamento simile si possono avere altre unioni per ottenere  $P_0$ . Le diramazioni si hanno in base a dei vincoli (una variabile maggiore o minore di qualcosa etc ...).

Nel caso di PB si ramifica in base ad ogni booleano di ogni variabile, ottenendo un albero binario.

La variabile usata per il branching è detta **variabile di branching** che vengono utilizzate secondo l'ordine degli indici.

Si hanno due modi per dividere l'insieme delle soluzioni ammissibili, fissando una variabile, per esempio  $x_1$ :

- fissare il valore di  $x_1$  a tutti i suoi possibili valori
- specificare un insieme di valori per ogni nuovo sotto-problema che si vuole generare

Nel caso di PML si procede così:

- si calcola la soluzione ottima del rilassamento lineare di  $P_0$
- si sceglie come variabile di branching una  $x_j$  che non abbia valore intero nella soluzione ottima del rilassamento lineare
- si divide  $P_0$  in  $P_1$  e  $P_2$  specificando due intervalli per  $x_j$ :

$$\begin{aligned} - & x_j \leq [x_j^*] \\ - & x_j \geq [x_j^*] + 1 \end{aligned}$$

2. la determinazione di un limite superiore, ovvero il **bounding**. Ad ognuno dei sotto-problemi definiti nel branching si può associare un bound, ovvero un limite,  $z_1, \dots, z_k$  su quanto buona possa essere la soluzione migliore di ogni sotto-problema. Questi bound si ottengono con il **rilassamento lineare associato**. Risolvo quindi i rilassamenti lineari dei sotto problemi e riporto nell'albero del branching sia la soluzione ottima che il bound (il valore di  $z$ ) di ogni sottoproblema.

Si definisce **soluzione incombente**  $x^*$  per  $P_0$  la miglior (la più grande se ho max, la più piccola se ho min) soluzione ammissibile (quindi intera) trovata finora, insieme al suo valore di funzione obiettivo. All'inizio si ha che  $z^* = -\infty$

3. l'eliminazione, ovvero il **fathoming**. Si ha che un sotto-problema  $P_i$  può essere "tagliato via" (fathoming) e quindi escluso dal ragionamento se:



- il suo upper bound è  $\leq z^*$ . Infatti non è sicuramente la soluzione migliore e un suo branch non darà una soluzione migliore
- il suo rilassamento non dà soluzioni ammissibili intere
- la soluzione ottima del rilassamento lineare è intera, in quanto se è migliore della soluzione incombente attuale essa diventa la nuova soluzione incombente (nel caso di PML basta che le componenti relative alla variabili intere della soluzione ottima siano intere)

Quindi, ricapitolando, per un PB si ha il seguente algoritmo  $B\&B$ :

1. **inizializzazione:** si pone  $z^* = -\infty$ . Si fa poi il passo di bounding e quello di fathoming. Si esegue il test di ottimalità e si decide se tagliare via il problema. Se non accade si classifica il problema come sotto problema e si itera con  $B\&B$

2. **iterazione del B&B:**

- **branching:** tra i sottoproblemi rimanenti, quelli *unfathomed* se ne sceglie uno (per esempio quello generato più di recente) e si generano due sotto-problemi a partire da esso fissando la successiva variabile di branching a 0 o a 1
- **bounding:** per ogni sotto-problema ottendo il bound mediante il semplice al suo rilassamento lineare
- **fathoming:** per ogni nuovo sotto-problema seguo i 3 criteri per scartare quelli necessari

**test di ottimalità:** iterare fino ad esaurire i sotto-problemi. Quando smetto di iterare ho la soluzione incombente ottima altrimenti continuo ad iterare

Per un problema PLM:

1. **inizializzazione:** si pone  $z^* = -\infty$ . Si fa poi il passo di bounding e quello di fathoming. Si esegue il test di ottimalità e si decide

se tagliare via il problema. Se non accade si classifica il problema come sotto problema e si itera con  $B\mathcal{E}B$

2. iterazione del B&B:

- **branching:** tra i sottoproblemi rimanenti, quelli *unfathomed* se ne sceglie uno (per esempio quello generato più di recente). Tra le variabili inette che non hanno valore intero nella soluzione ottima del rilassamento lineare associato ne scelgo una come variabile di branching (per esempio la prima in ordine di indice)  $x_j$ . Sia inoltre  $x_j^*$  il relativo valore nella soluzione ottima. Genero quindi due sotto-problemi aggiungendo rispettivamente i vincoli  $x_j \leq \lfloor x_j^* \rfloor$  e  $x_j \geq \lfloor x_j^* \rfloor + 1$
- **bounding:** per ogni sotto-problema ottendo il bound mediante il simplesso al suo rilassamento lineare
- **fathoming:** per ogni nuovo sotto-problema seguo i 3 criteri per scartare quelli necessari

**test di ottimalità:** iterare fino ad esaurire i sotto-problemi. Quando smetto di iterare ho la soluzione incumbente ottima altrimenti continuo ad iterare

## Capitolo 5

# Programmazione Non Lineare

Un **problema di programmazione non lineare PNL** può essere formulato da una funzione obiettivo:

$$\text{opt } f(x)$$

con  $\text{opt} = \{\min, \max\}$  e soggetta ai seguenti vincoli:

$$g_j(x) \leq b_j, \quad j = 1, \dots, m$$

$$x_i \geq 0 \quad i = 1, \dots, n$$

sapendo che  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  e  $g_j : \mathbb{R}^n \rightarrow \mathbb{R}$  sono funzioni note di  $x \in \mathbb{R}^n$  (e non lineari come in un problema PL).

*È più frequente avere problemi non lineari che lineari.*

**Un problema non lineare presenta un'area ammissibile non delimitata da rette ma da curve e quindi l'ottimo non è più da cercare solo sulla frontiera.**

Vediamo ora l'**ottimizzazione non vincolata in una variabile**.

Considero un PNL con la seguente funzione obiettivo da minimizzare

$$\min_{x \in \mathbb{R}} f(x)$$

la condizione sufficiente affinché  $x^*$  sia il punto di minimo è che la derivata in quel punto sia nulla (infatti in quel punto della funzione si avrà sicuramente un punto di ottimo):

$$\frac{df(x^*)}{dx} = 0$$

Il problema è che potremmo avere a che fare con punti di ottimo locali e non globali.

Studiamo quindi l'**ottimizzazione in una variabile**.

Considero un PNL con la seguente funzione obiettivo **convessa** da minimizzare:

$$\min_{x \in \mathbb{R}} f(x)$$

la condizione sufficiente affinché  $x^*$  sia il punto di minimo è che:

$$\frac{df(x^*)}{dx} = 0$$

si ha quindi che: *se tale equazione può essere risolta analiticamente allora il procedimento per trovare l'ottimo termina.*

**Capire se inserire il ripasso di analisi**

Si hanno quindi diverse tipologie di PNL:

- **ottimizzazione non vincolata:** ovvero quei problemi che non hanno vincoli sulla regione ammissibile e hanno la funzione obiettivo che è semplicemente:

$$\max_{z \in \mathbb{R}^n} f(x) \text{ o } \min_{z \in \mathbb{R}^n} f(x)$$

- **ottimizzazione con vincoli lineari:** ovvero i PNL che hanno le funzioni  $g_j(x)$  lineari, pur avendo la funzione obiettivo non lineare. Un caso particolare è la **PNL quadratica**, che presenta una funzione obiettivo, appunto, quadratica
- **programmazione convessa:** ovvero quei PNL in cui  $f(x)$  è concava o convessa mentre ogni funzione  $g_j(x)$  è convessa
- **programmazione non convessa:** ovvero tutti i PNL che non soddisfano le ipotesi di convessità

Si quindi la funzione obiettivo una funzione concava da massimizzare:

$$\max_{x \in \mathbb{R}} f(x)$$

la condizione sufficiente per avere un punto di massimo è sempre:

$$\frac{df(x^*)}{dx} = 0$$

se posso risolvere analiticamente l'equazione ho trovato il massimo (se fosse stata convessa il minimo). In mancanza mi affido ad **algoritmi di risoluzione numerica**.

- costruisco una sequenza di punti  $\{x_k\}$  tali che:

$$\lim_{k \rightarrow +\infty} x_k = x^*$$

*ad ogni iterazione  $k$ , partendo da  $k_k$  si esegue una ricerca sistematica per identificare un punto migliore  $x_{k+1}$*

- devo garantire la convergenza della sequenza al mio punto di ottimo. Se la funzione è continua e concava in un intervallo  $[a, b]$  considero il generico  $x_k$ :

- se  $\frac{df(x^*)}{dx} < 0$  allora l'ottimo è a sinistra di  $x_k$
- se  $\frac{df(x^*)}{dx} > 0$  allora l'ottimo è a destra di  $x_k$
- se  $\frac{df(x^*)}{dx} \approx 0$  allora l'ottimo  $x_k$ ,  $x_k \approx x^*$

Non abbiamo però la certezza che un numero finito di iterazioni ci possa portare ad un risultato. Abbiamo dei criteri per capire quando fermarci:

- come detto sopra quando la soluzione è abbastanza accurata, ovvero  $\frac{df(x^*)}{dx} \approx 0$
- quando si è raggiunto un numero massimo di iterazioni  $N$  stabilito a priori
- quando si hanno progressi lenti, ovvero  $|x_{k+1} - x_k| < \varepsilon_k$  o  $|f(x_{k+1}) - f(x_k)| < \varepsilon_f$
- quando la soluzione diverge, ovvero  $|x_k| \rightarrow +\infty$
- quando si hanno dei cicli, ovvero da  $x_k$  si arriva a  $x_{k+1}$  e da  $x_{k+1}$  si arriva a  $x_k$

Si hanno due tipi di algoritmi possibili:

1. **algoritmi dicotomici**, che risolvono l'equazione della derivata uguale a zero e ad ogni iterazione riducono l'intervallo di ricerca all'interno del quale la funzione derivata si annulla
2. **algoritmi di approssimazione**, che usano approssimazioni locali della funzione

Partendo dalla *formula di Taylor*, centrata in  $x_k$ , si può ottenere un'approssimazione quadratica di tale funzione:

$$f(x_{k+1}) = f(x_k) + f'(x_{k+1} - x_k) + \frac{1}{2}f''(x_k)(x_{k+1} - x_k)^2 + O((x_{k+1} - x_k)^2)$$

$\Downarrow$

$$f(x_{k+1}) \approx f(x_k) + f'(x_{k+1} - x_k) + \frac{1}{2}f''(x_k)(x_{k+1} - x_k)^2$$

Si ha che l'approssimazione quadratica è funzione solo di  $x_{k+1}$ . Calcolo quindi la derivata dell'approssimazione quadratica:

$$f'(x_k) + f''(x_k)(x_{k+1} - x_k) = 0 \implies x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Per l'ottimizzazione in una variabile si hanno 2 metodi:

1. **metodo di bisezione**, che si applica quando  $f(x)$  è concava, continua su un intervallo  $[a, b]$  e derivabile su  $(a, b)$ .

Si basa sul guardare la pendenza della funzione (ovvero la derivata) per un punto di  $x$  per capire dove cercare il massimo (ribaltando nel caso di funzioni concave e ricerca del minimo):

- se  $f'(x) > 0$  allora si può ottenere un nuovo punto spostandosi verso destra
- se  $f'(x) < 0$  allora si può ottenere un nuovo punto spostandosi verso sinistra

inoltre, considerando un generico  $x_k$ :

- se  $f'(x_k) > 0$  allora  $x_k$  è un estremo inferiore  $\underline{x}$  per  $x_{k+1}$
- se  $f'(x_k) < 0$  allora  $x_k$  è un estremo superiore  $\bar{x}$  per  $x_{k+1}$

per ogni passo  $k$  si identifica un **intervallo di ricerca**:

$$[a_k, b_k] \subset [a_{k-1}, b_{k-1}] \subset [a, b]$$

per ridurre lo spazio di ricerca in modo da identificare un'iterazione  $k$  che presenta:

$$|b_k - a_k| < 2\varepsilon \text{ con } |a_k - x^*| < \varepsilon \text{ e } |x^* - b_k| < \varepsilon$$

Vediamo ora l'algoritmo nel dettaglio, nel caso di problema di max:

- **inizializzazione:** si fissano un  $\varepsilon$  e  $k = 0$ . Si calcolano  $\bar{x}$  e  $\underline{x}$  cercando valori di  $x$  per cui la derivata sia positiva e negativa. Si sceglie il punto iniziale:

$$x_0 = \frac{\bar{x} + \underline{x}}{2}$$

- **iterazione del metodo della bisezione:**
  - calcolo  $f'(x_k)$
  - se  $f'(x_k) = 0$  allora  $x_k = x^*$
  - se  $f'(x_k) < 0$  allora  $\bar{x} = x_k$
  - se  $f'(x_k) > 0$  allora  $\underline{x} = x_k$
  - infine pongo  $x_{k+1} = \frac{\bar{x} + \underline{x}}{2}$  e  $k = k + 1$
- **criterio d'arresto:** se  $\bar{x} - \underline{x} \leq 2\varepsilon$  allora  $x^*$  avrà una distanza minore di  $\varepsilon$  dall'estremo superiore (o inferiore). In caso contrario procedo con un'altra iterazione

**Il metodo di bisezione converge molto lentamente in quanto considera unicamente le informazioni della derivata prima**

2. **metodo di Newton**, che si basa sull'usare l'ottimo dell'approssimazione quadratica di  $f(x)$ :

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

che è la formula che viene usata ad ogni generica iterazione  $k$  per calcolare  $x_{k+1}$ .

Si ha che se la funzione è concava allora  $x_k$  converge verso un punto di massimo in quanto:

- se  $f(x)$  è concava allora se  $x_k$  è a sinistra del punto di massimo si ha che  $f'(x) > 0$ , quindi  $-\frac{f'(x_k)}{f''(x_k)} > 0$  e, di conseguenza,  $x_{k+1} > x_k$
- se  $f(x)$  è concava allora se  $x_k$  è a destra del punto di massimo si ha che  $f'(x) < 0$ , quindi  $-\frac{f'(x_k)}{f''(x_k)} < 0$  e, di conseguenza,  $x_{k+1} < x_k$

*Se le funzione fosse stata convessa si avrebbe avuto la convergenza di  $x_k$  verso il minimo.*

Vediamo quindi nel dettaglio l'algoritmo di Newton:

- **inizializzazione:** si fissa  $\varepsilon$  e  $k = 0$
- **iterazione del metodo di Newton:**
  - si calcolano  $f'(x)$  e  $f''(x)$
  - si pone  $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$
- **criterio di arresto:** se  $|x_{k+1} - x_k| \leq \varepsilon$  si ha che  $x_{k+1} = x^*$  e quindi è la soluzione ottima, altrimenti itero nuovamente con  $k = k + 1$

**Può succedere che l'algoritmo di Newton diverga (mentre il metodo della bisezione ci assicura la convergenza) ma gode di una velocità di convergenza quadratica.**

*L'algoritmo di Newton fallisce se il punto iniziale è lontano dall'ottimo (si ha una convergenza globale scarsa)*

### 5.0.1 Il Caso N-Dimensionale

Nel caso unidimensionale la derivata prima della funzione obiettivo ci dà informazioni sull'ottimalità di un punto mentre la derivata seconda fornisce indicazioni sulla convessità della funzione.

Spostandoci al caso n-dimensionale si ha che la derivata prima viene sostituita dal **gradiente** e la derivata seconda dall'**Hessiano**.

**Vediamo un piccolo ripasso di alcuni concetti fondamentali:**

- **derivata direzionale:**

sia  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  una funzione continua e  $v \in \mathbb{R}^n$  un vettore direzionale, quindi  $\|v\| = 1$ , allora si ha che:

$$\lim_{h \rightarrow 0} \frac{f(x + hv) - f(x)}{h} = \frac{\partial f(x)}{\partial v}$$

che viene chiamata **derivata direzionale di  $f$  in  $x$  nella direzione  $v$** .

Si ha il caso particolare delle direzioni della base canonica  $\frac{\partial f(x)}{\partial e_i}$  che vengono chiamate **derivate parziali**



- **gradiente:**

il gradiente rappresenta l'estensione del concetto di derivabilità per una funzione in più dimensioni e corrisponde al vettore delle derivate parziali, data  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  una funzione reale:

$$\nabla f(x) = \left[ \frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right]$$

**Esempio 26.**

$$f = 15x_1 + 2(x_2)^3 - ex_1(x_3)^2$$

$\Downarrow$

$$\nabla f = [15 - 3(x_3)^2, 6(x_2)^2, -6x_1x_3]$$

- **Matrice Hessiana:**

la matrice Hessiana rappresenta l'estensione del concetto di derivata seconda per una funzione in più dimensioni ed è una matrice di dimensioni  $n \times n$ :

$$\nabla^2 f = H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Inoltre si ha che se le derivate seconde sono continue allora si ha che:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$$

e quindi l'Hessiana di  $f$  è simmetrica

**Esempio 27.**

$$f = 15x_1 + 2(x_2)^3 - ex_1(x_3)^2$$

$\Downarrow$

$$\nabla^2 f = \begin{bmatrix} 0 & 0 & -6x_2 \\ 0 & 12x_2 & 0 \\ -6x_3 & 0 & -6x_1 \end{bmatrix}$$

Sia quindi  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  una funzione reale due volte differenziabile. Per ogni punto  $x_0$  è possibile calcolare l'Hessiana in quel punto:

$$H_f(x_0) = \begin{bmatrix} \frac{\partial^2 f(x_0)}{\partial x_1^2} & \frac{\partial^2 f(x_0)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x_0)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x_0)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x_0)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x_0)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x_0)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x_0)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x_0)}{\partial x_n^2} \end{bmatrix}$$

- **matrice positiva o negativa:**

Si hanno i seguenti casi:

- una matrice quadrata è **definita positiva** se tutti gli autovalori della matrice sono positivi
- una matrice quadrata è **definita negativa** se tutti gli autovalori della matrice sono negativi
- una matrice quadrata è **semi-definita positiva** se tutti gli autovalori della matrice sono non negativi
- una matrice quadrata è **semi-definita negativa** se tutti gli autovalori della matrice sono non positivi

definita quindi l'Hessiana in un punto  $x_0$  si ha che:

- se  $H_f(x_0)$  è definita positiva la funzione  $f(x)$  è convessa in quel punto
- se  $H_f(x_0)$  è definita negativa la funzione  $f(x)$  è concava in quel punto
- se la funzione  $f(x)$  è convessa in quel punto allora  $H_f(x_0)$  è semi-definita positiva
- se la funzione  $f(x)$  è concava in quel punto allora  $H_f(x_0)$  è semi-definita negativa

### 5.0.2 PNL non Vincolata

Spostiamoci nel caso  $n$ -dimensionale. Per cercare i punti di ottimo si richiede che:

$$\nabla f = 0$$

ovvero si richiede che derivando secondo le  $n$  variabili si abbia:

$$\frac{\partial f}{\partial x_i} = 0, \quad \forall i = 0, \dots, n$$

una volta trovati i punti di ottimo (mettendo a sistema le varie derivate parziali uguali a 0) valuto la matrice Hessiana, se definita positiva ho un punto di minimo, se definita negativa ho un massimo. Per calcolarlo calcolo gli  $n$  autovalori.

**Questo metodo funziona in casi particolari, con una funzione di cui sappiamo discutere la convessità e avendo equazioni lineari facili da risolvere. In caso contrario si hanno *algoritmi numerici per l'ottimizzazione non lineare*.**

Si hanno due metodi:

1. **metodo del gradiente (*steepest descent*)**
2. **metodo di Newton**

Entrambi appartengono alla categoria di strategie del tipo **line-search** che generano una successione di punti  $x_k$  nello spazio  $\mathbb{R}^n$  tali che  $x_{k+1}$  è ottenuto da  $x_k$  muovendosi lungo una direzione di salita (nel caso di massimo) o di discesa (nel caso di minimo). Nel dettaglio si ha che, data una generica  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  e un generico punto  $x \in \mathbb{R}^n$ , dove  $f$  è differenziabile (indicando con  $\langle \rangle$  il prodotto scalare tra vettori):

- un vettore  $v \in \mathbb{R}^n$  è una **direzione di discesa** se:

$$\langle v, \nabla f(x) \rangle < 0$$

- un vettore  $v \in \mathbb{R}^n$  è una **direzione di salita** se:

$$\langle v, \nabla f(x) \rangle > 0$$

**dubbi sulla parte a seguire (i due elenchi)**

Quindi se voglio cercare un minimo posso procedere così:

- considero un punto generico  $x^0$  e  $k = 0$
- cerco una funzione di discesa  $d^k$
- cerco un nuovo  $x^{k+1}$  lungo  $d^k$ :

$$x^{k+1} = x^k + \alpha^k d^k$$

con  $k$  che è la generica iterazione e  $\alpha^k > 0$  è una quantità scalare chiamata **step-size**

e se voglio cercare un massimo posso procedere così:

- considero un punto generico  $x^0$  e  $k = 0$
- cerco una funzione di salita  $d^k$
- cerco un nuovo  $x^{k+1}$  lungo  $d^k$ :

$$x^{k+1} = x^k + \alpha^k d^k$$

con  $k$  che è la generica iterazione e  $\alpha^k > 0$  è una quantità scalare chiamata **step-size**

Sapendo il punto di partenza  $x^k$  e la direzione di ricerca  $d^k$  calcoliamo lo step size. Sappiamo che cerchiamo di ottimizzare la funzione:

$$f(x^{k+1}) = x^k + \alpha^k d^k$$

ovvero la *restrizione di  $f(x)$  lungo  $d^k$*  e sappiamo che l'unica incognita risulta essere  $\alpha^k$ , quindi mi basta cercare:

$$\frac{df(x^k + \alpha^k d^k)}{d\alpha^k} = 0$$

e usare un metodo di ottimizzazione per trovare  $\alpha^k$  ottima.

### Metodo del Gradiente

Con questo metodo si impone la *direzione di crescita* per problemi di massimo:

$$d^k = \nabla f(x^k)$$

e la *direzione di decrescita* per i problemi di minimo:

$$d^k = -\nabla f(x^k)$$

ottenendo quindi:

$$x^{k+1} = x^k \pm \alpha^k d^k$$

**Il gradiente fornisce una direzione certa della salita della funzione**  
Vediamone quindi l'algoritmo:

1. si considera un generico punto  $x^0$  e si pone  $k = 0$
2. si calcola:

$$\nabla f(x^k)$$

con  $k$  indicante il numero di iterazione

3. si pone per problemi di minimo:

$$d^k = -\nabla f(x^k)$$

o per problemi di massimo:

$$d^k = \nabla f(x^k)$$

4. si pone (**secondo me è**  $\alpha^k$ ):

$$x^{k+1} = x^k + \alpha^k d^k$$

5. per problemi di minimo si calcola:

$$\alpha^k = \arg \max f(x^k + \alpha d^k)$$

con  $\arg \max$  che indica l'**argomento del massimo**, ovvero *insieme dei punti di un dato argomento per i quali una data funzione raggiunge il suo massimo*.

per problemi di minimo si ha:

$$\alpha^k = \arg \min f(x^k + \alpha d^k)$$

con  $\arg \min$  che indica l'**argomento del minimo**, ovvero *insieme dei punti di un dato argomento per i quali una data funzione raggiunge il suo minimo*.

In entrambi i casi **utilizzando un metodo di ottimizzazione per funzioni in una variabile**

6. come **criterio di arresto** si ha che ci si ferma se:

$$|f(x^{k+1}) - f(x^k)| < \varepsilon_1$$

o

$$\|x^{k+1} - x^k\| < \varepsilon_2$$

con  $\varepsilon_i$  piccolo a piacere (solitamente nell'ordine dei  $10^{-1}$ ).

Se non ci si arresta si pone  $k = k + 1$  e si ritorna al calcolo del gradiente

### 5.0.3 Metodo di Newton

Con questo metodo si approssima nell'intorno del punto corrente la funzione  $f(x)$  con una funzione quadratica. Questa nuova funzione viene ottimizzata per ottenere un nuovo punto  $x^k + \Delta x$ .

Partiamo dallo **sviluppo di Taylor per funzioni a più variabili**:

*Dubbi su quanto scritto*

$$f(x^k + \Delta x) = f(x^k) + \nabla f(x^k)\Delta x + \frac{1}{2}\Delta x H_f(x^k)\Delta x$$

cerco quindi una direzione di miglioramento che ottimizzi la funzione nel nuovo punto. Ho quindi bisogno di:

$$\frac{\partial f(x^k + \Delta x)}{\partial \Delta x} = 0$$

e quindi di:

$$H_f(x^k)\Delta x = -\nabla f(x^k)$$

Si ha quindi il cosiddetto **Newton step** che muove ad un punto stazionario del secondo ordine dello sviluppo di Taylor di  $f(x^k)$ .

Vediamo quindi i passi dell'algoritmo di Newton:

1. si seleziona un punto iniziale  $x^0$
2. si calcolano  $\nabla f(x^k)$  e  $H_f(x^k)^{-1}$
3. si calcola il nuovo punto con la seguente equazione:

$$x^{k+1} = x^k - H_f(x^k)^{-1}\nabla f(x^k)$$

4. come **criterio di arresto** si ha che ci si ferma se:

$$|f(x^{k+1}) - f(x^k)| < \varepsilon_1$$

o

$$\|x^{k+1} - x^k\| < \varepsilon_2$$

con  $\varepsilon_i$  piccolo a piacere (solitamente nell'ordine dei  $10^{-1}$ ).

Se non ci si arresta si ritorna al calcolo del del gradiente e dell'Hessiana

Si può notare quindi che, avendo ad ogni iterazione sia il calcolo del gradiente che dell'Hessiana e dovendo invertire una matrice  $n \times n$  con  $n$  numero di variabili, si ha un grande sforzo computazionale, ripagato parzialmente dalla velocità di convergenza.

Si ha che permette una sola iterazione nel caso di funzioni quadratiche.

## 5.1 Ottimizzazione Non Lineare Vincolata

Consideriamo un generico problema di programmazione non lineare vincolata:

$$\text{opt } f(x)$$

con  $\text{opt} = \{\min, \max\}$ , soggetta ai *vincoli di uguaglianza*:

$$g_i(x) = 0 \quad i = 1, \dots, m$$

e ai *vincoli di disuguaglianza*:

$$h_j(x) \leq 0 \quad j = 1, \dots, p$$

Si hanno 3 metodi:

1. riduzione del numero di variabili libere (dimensionality reduction)
2. metodo dei moltiplicatori di Lagrange
3. condizioni di Karash-Kuhn Tucker

### 5.1.1 Dimensionality Reduction

Supponiamo di avere un problema di ottimizzazione soggetto ad un certo numero  $l$  di vincoli di uguaglianza.

Nel caso in cui sia possibile esplicitare  $l$  variabili in funzione delle restanti  $n-l$  variabili utilizzando i vincoli di uguaglianza del problema allora possiamo trasformare tale problema in un problema di ottimizzazione non vincolata con  $n-l$  variabili. Quindi, per esempio, si passerebbe da:

$$\begin{aligned} \max f(x_1, \dots, x_n) \\ g_j(x_1, \dots, x_n) = b_j \quad j = 1, \dots, l \end{aligned}$$

a:

$$\begin{aligned} \max f(x_1, \dots, x_{n-l}) \\ x_{n-l+1} = \tilde{g}_1(x_1, \dots, x_{n-l}) \\ \dots \\ x_n = \tilde{g}_l(x_1, \dots, x_{n-l}) \end{aligned}$$

**Questo metodo non è sempre applicabile.**

Per esempio, fissato  $x_1$ ,  $x_2$  può assumere 2 valori e viceversa. Quindi non posso esplicitare univocamente una variabile in funzione dell'altra.

### 5.1.2 Lagrangiana

Consideriamo un generico problema di programmazione non lineare vincolata con solo vincoli di uguaglianza:

$$\text{opt } f(x_1, \dots, x_n)$$

con  $\text{opt} = \{\min, \max\}$ , soggetto ai vincoli:

$$g_i(x_1, \dots, x_n) = 0, \quad i = 1, \dots, m$$

Si consideri la funzione in  $n+m$  variabili:

$$L(x_1, \dots, x_n, \lambda_1, \dots, \lambda_m) = f(x_1, \dots, x_n) + \sum_{i=1}^m \lambda_i \cdot g_i(x_1, \dots, x_n)$$

Questa funzione è detta **Funzione Lagrangiana** e le variabili  $\lambda_i$  (di numerosità pari al numero di variabili) sono detti **Moltiplicatori di Lagrange**. Si ha che:

*i punti stazionari della Lagrangiana sono fortemente legati agli ottimi della*



funzione.

Sia quindi  $x^* = (x_1^*, \dots, x_n^*)$  il punto stazionario di  $f$ . Esistono quindi  $m$  moltiplicatori di Lagrange  $\lambda^* = (\lambda_1^*, \dots, \lambda_m^*)$  tali che il punto  $(x^*, \lambda^*)$  è un punto stazionario della Lagrangiana associata:

$$L(x^*, \lambda^*) = f(x^*) + \sum_{i=1}^m \lambda_i^* \cdot g_i(x^*)$$

Si hanno due osservazioni:

1.  $x^*$  e  $(x^*, \lambda^*)$  possono essere punti stazionari di tipo diverso (massimo, minimo, di sella)
2. tale condizione fornisce una condizione necessaria ma non sufficiente per l'ottimizzazione del problema vincolato

Si ha quindi che nel punto  $(x^*, \lambda^*)$  il gradiente della funzione Lagrangiana è nullo:

$$\frac{\partial L(x^*, \lambda^*)}{\partial x_i} = 0 \quad i = 1, \dots, n$$

$$\frac{\partial L(x^*, \lambda^*)}{\partial \lambda_i} = 0 \quad i = 1, \dots, m$$

I punti che annullano il gradiente rappresentano i punti candidati ad essere punto di ottimo della funzione:

1. se la funzione  $f$  è convessa allora i punti stazionari sono punti di minimo
2. se la funzione  $f$  è concava allora i punti stazionari sono punti di massimo

Si nota inoltre che:

- $\frac{\partial L(x^*, \lambda^*)}{\partial \lambda_i} = 0 \quad i = 1, \dots, m$  è un modo differente per riscrivere i vincoli  $g_i(x_1, \dots, x_n)$
- $\frac{\partial L(x^*, \lambda^*)}{\partial x_i} = 0 \quad i = 1, \dots, n$  può essere riscritto come:

$$\frac{\partial L(x^*, \lambda^*)}{\partial x_i} = \frac{\partial f(x^*)}{\partial x_i} + \sum_{i=1}^m \lambda_i^* \cdot \frac{\partial g_i(x^*)}{\partial x_i} = 0$$

ovvero:

$$\begin{aligned}\nabla L(x^*, \lambda^*) &= \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \cdot \nabla g_i(x^*) = 0 \\ &\Downarrow \\ \nabla f(x^*) &= - \sum_{i=1}^m \lambda_i^* \cdot \nabla g_i(x^*)\end{aligned}$$

Quest'ultimo risultato comporta:

- l'insieme dei vincoli di fatto restringe lo spazio delle soluzioni ammissibili del problema al sottospazio intersezione dato dall'intersezione dei vari vincoli
- $\nabla f(x^*)$  nel punto di ottimo  $x^*$  deve risultare ortogonale a tale sottospazio
- il vettore ortogonale a tale sottospazio è dato da:

$$\sum_{i=1}^m \lambda_i^* \cdot \nabla g_i(x^*)$$

quindi:

$$\nabla f(x^*) = - \sum_{i=1}^m \lambda_i^* \cdot \nabla g_i(x^*)$$

ovvero nel punto di ottimo il gradiente della funzione  $f$  può essere riscritto come combinazione lineare dei gradienti dei vincoli

Fino ad ora però abbiamo usato la Lagrangiana avendo solo vincoli di uguaglianza. Aggiungiamo anche i vincoli di disuguaglianza:

$$h_j(x_1, \dots, x_n) \geq 0, \quad j = 1, \dots, p$$

Si ha quindi che il punto di ottimo  $x^*$  è tale che:

- $g_i(x^*) = 0, \quad \forall i = 1, \dots, m$
- $h_j(x^*) = 0, \quad \forall j = 1, \dots, p$  se ho un vincolo attivo
- $h_j(x^*) < 0, \quad \forall j = 1, \dots, p$  se ho un vincolo non attivo, in questo caso significa che un piccolo cambiamento non viola il vincolo

Si definisce la **Lagrangiana generalizzata** come una funzione in  $n + m + p$  variabili:

$$\begin{aligned}L(x_1, \dots, x_n, \lambda_1, \dots, \lambda_m, \mu_1, \dots, \mu_p) &= \\ &= f(x_1, \dots, x_n) + \sum_{i=1}^m \lambda_i \cdot g_i(x_1, \dots, x_n) + \sum_{i=1}^p \mu_i \cdot h_i(x_1, \dots, x_n)\end{aligned}$$

### 5.1.3 KKT

Le condizioni di Karush-Kuhn-Tucker rappresentano una generalizzazione del metodo dei moltiplicatori di Lagrange, applicato a problemi in cui siano presenti anche dei vincoli di disuguaglianza.

Consideriamo il solito problema di programmazione non lineare vincolata:

$$\text{opt } f(x)$$

con  $\text{opt} = \{\min, \max\}$ , soggetta ai *vincoli di uguaglianza*:

$$g_i(x) = 0 \quad i = 1, \dots, m$$

e ai *vincoli di disuguaglianza*:

$$h_j(x) \leq 0 \quad j = 1, \dots, p$$

Sia  $x^* = (x_1^*, \dots, x_n^*)$  un ottimo di  $f$ , allora  $h_j(x^*)$  può essere:

- attivo, se  $h_j(x^*) = 0$
- inattivo se  $h_j(x^*) < 0$

L'insieme dei vincoli attivi si indica con  $l(x^*)$ .

Sia  $x^* = (x^* : 1, \dots, x_n^*)$  il punto di ottimo di  $f$ . Esistono, di conseguenza,  $m$  moltiplicatori  $\lambda^* = (\lambda_1^*, \dots, \lambda_m^*)$  e  $p$  moltiplicatori  $\mu^* = (\mu_1^*, \dots, \mu_p^*)$  valgono le seguenti condizioni:

- **condizione di stazionarietà:**

– per problemi di massimo:

$$\nabla f(x^*, \lambda^*, \mu^*) - \sum_{i=1}^m \lambda_i^* \cdot \nabla g_i(x^*) - \sum_{j=1}^p \mu_j^* \cdot \nabla h_j(x^*) = 0$$

– per problemi di minimo:

$$\nabla f(x^*, \lambda^*, \mu^*) + \sum_{i=1}^m \lambda_i^* \cdot \nabla g_i(x^*) + \sum_{j=1}^p \mu_j^* \cdot \nabla h_j(x^*) = 0$$

- **ammissibilità primale:**

$$g_i(x^*) = 0, \quad \forall i = 1, \dots, m$$

$$h_j(x^*) \leq 0, \quad \forall j = 1, \dots, p$$

- ammissibilità duale:

$$\mu_j \geq 0, \quad \forall j = 1, \dots, p$$

- condizione di complementarietà:

$$\mu_j^* \cdot h_j(x^*) = 0, \quad \forall j = 1, \dots, p$$

Le ultime due condizioni inoltre implicano che, nel caso di vincoli di disuguaglianza:

- se ho un vincolo non attivo, quindi  $h_j(x^*) < 0$  allora  $\mu_j^* = 0$
- se ho un vincolo attivo, quindi  $h_j(x^*) = 0$  allora  $\mu_j^* \geq 0$

**Le condizioni di *KKT* rappresentano le condizioni necessarie ma non sufficienti che caratterizzano i punti di ottimo (condizioni di ottimalità).**

I punti di ottimo devono soddisfare le condizioni di KKT che “limitano” la ricerca dei punti di ottimo tra quelli che soddisfano le condizioni.

***Le condizioni KKT non sono un algoritmo***

# Capitolo 6

## Metaeuristiche

Andiamo ad analizzare il mondo nei problemi NP-hard. Si possono avere:

- **metodo B&B**, esatto ma computazionalmente troppo costoso
- **metodi approssimati**, che trova una soluzione di valore corretto entro un certo errore
- **metodi euristici**, che identificano una buona soluzione ammissibile ma non necessariamente esatta. Sono metodi efficienti su larga scala che iterativamente cercano una soluzione migliore ad ogni iterazione. Sono studiati *ad hoc* sul problema. Spesso sono usati in problemi *TSP* o di ottimizzazione non lineare con molti ottimi locali
- **metodi metaeuristici**, che sono di carattere generale, quindi indipendenti dal problema, potendo essere usate come **black box**.

**Definizione 9.** *Una metaeuristica è una metodologia risolutiva generale che gestisce l'iterazione tra le procedure di miglioramento locale e strategie a più alto livello al fine di creare un processo in grado di allontanarsi da un ottimo locale ed eseguire una ricerca approfondita della regione ammissibile. Allontanandosi da un ottimo locale si accerrano anche punti con valore peggiore della funzione obiettivo. Non si garantisce che la soluzione sia migliore ma, d'altro canto, ci si muove velocemente verso le soluzioni migliori.*

A differenza di un metodo euristico si hanno elementi di casualità detti **randomness**, inoltre non si cerca la soluzione migliore ad ogni iterazione. Si hanno tre metaeuristiche interessanti:

1. **tabù Search**

2. Simulated Annealing
3. Algoritmi genetici

## 6.1 Tabù Search

La Tabu Search è un'evoluzione del classico metodo di discesa detto **steepest descent**. L'idea base è partire da una soluzione iniziale ed eseguire una serie di mosse (dettate da una euristica) che portano ad una nuova soluzione nell'intorno di un punto corrente. Si hanno le cosiddette **mosse di non miglioramento** per non rimanere bloccati in un ottimo locale. Si hanno quindi i **tabù**, ovvero si proibiscono le ultime mosse dell'algoritmo evitando di ritornare in un ottimo già analizzato. Si hanno quindi tre step:

1. la **ricerca locale**, con l'accettazione anche di mosse peggiorative
2. la **tabù list**, ovvero la lista di mosse non più consentite
3. la **exploitation vs exploration**, ovvero il bilanciare la parte di ottimizzazione locale con quella di esplorazione

Vediamo quindi l'algoritmo:

- **inializzazione:** scegliere una soluzione ammissibile iniziale
- **iterazione:**
  - usare un metodo di ricerca locale per identificare le mosse ammissibili che non sono nella tabu list. Scegliere una mossa Tabù solo se è la migliore tra tutte le soluzioni trovate fino a quel momento, questo è il **criterio di aspirazione**
  - scegliere la mossa migliore
  - modificare la lista tabù aggiungendo la mossa corrente. Se la lista tabu è piena, rimuovere gli elementi più vecchi
- **criteri di stop:**
  - massimo numero di iterazioni raggiunto
  - massimo tempo computazionale raggiunto
  - massimo numero di iterazioni successive senza alcun miglioramento

– nessun mossa ammissibile permessa

**Si accetta come soluzione ottima la migliore ottenuta.**

A priori non si può sapere che ricerca locale usare, come rappresentare le mosse nella lista tabù, non si sanno definire bene i criteri dis top etc. ...

(Su slide tabù search su *TSP* e *spanning tree*)

Nel complesso quindi la Tabù Search ha diversi **pro**:

- l'uso anche di mosse di peggioramento consente di non rimanere intrappolati in un punto di minimo locale
- l'implementazione della Tabù list consente una maggiore esplorazione dello spazio delle soluzioni
- può essere applicato sia per problemi discreti che continui
- Per problemi complessi (scheduling, quadratic assignment and vehicle routing), il tabù search ottiene soluzioni che spesso sono migliori rispetto ad altre tecniche di ottimizzazione

ma anche diversi **contro**:

- vi sono molto parametri da dover impostare
- il numero di iterazioni necessarie potrebbe essere molto alto per avere una buona soluzione
- non vi è garanzia di raggiungere il punto di ottimo globale

## 6.2 Simulated Annealing

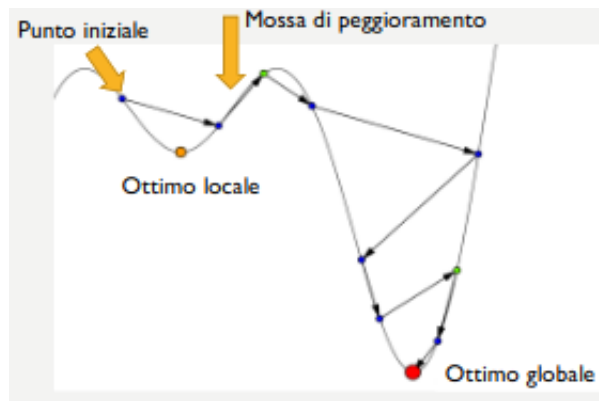
Questo metodo prende spunto dal mondo metallurgico, dove per eliminare difetti causati da ordini dei reticoli cristallini che vengono a mancare si procede con la loro parziale rifusione, l'**annealing**, seguita da una procedura di raffreddamento molto lenta.

Il **Simulating Annealing** cerca di stimare l'ottimo globale evitando quelli locali. Continuando l'analogia della metallurgica si ha:

- l'ottimo globale corrisponde al metallo con struttura cristallina senza difetti
- gli ottimi locali corrispondono al metallo con struttura cristallina con presenza di difetti reticolari

- la procedura algoritmica per trovare il minimo globale, sfuggendo ai minimi locali, corrisponde al processo di lento raffreddamento ed eventuale riscaldamento.

Questo metodo cerca una nuova soluzione in un intorno della soluzione corrente ma sfruttando, a differenza della tabù search, mosse random. Si accettano sia mosse di miglioramento che di peggioramento per scavalcare gli ottimi locali.



Vediamo come procedere.

Si seleziona, per esempio, un generico problema di massimizzazione  $\max f(x)$ . Bisogna capire come scegliere la mossa successiva:

- dalla soluzione corrente  $x_c$  seleziono in modo random una nuova soluzione  $x_n$  in un intorno di  $x_c$
- si definisce  $z_c = f(x_c)$  il valore della funzione obiettivo corrispondente a  $x_c$
- si definisce  $z_n = f(x_n)$  il valore della funzione obiettivo corrispondente a  $x_n$
- se  $z_n \geq z_c$  allora accetto il nuovo punto candidato. Altrimenti posso comunque accettare l'attuale punto candidato con una certa probabilità:

$$P(\text{accept}) = e^{\frac{z_n - z_c}{T}}$$

con  $T$  detto **temperatura**

Si possono fare delle osservazioni:

- se  $z_c \gg z_n$  si ha che  $P \rightarrow 0$ , quindi peggiore è la nuova soluzione e minore probabilità ho di accettarla



- se aumenta allora aumenta la probabilità di accettare anche punti molto peggiori. Viceversa se  $T$  è bassa accetto solo punti di poco peggiori

In caso di minimizzazione si avrebbe:

$$P(\text{accept}) = e^{\frac{z_c - z_n}{T}}$$

Per simulare la probabilità si genera un numero random tra 0 e 1, se è minore di  $e^{\frac{z_n - z_c}{T}}$  accetto  $x_n$  altrimenti rifiuto.

La probabilità di accettazione è direttamente proporzionale a  $T$  e inversamente proporzionale a  $|z_n - z_c|$ . All'inizio la temperatura è molto alta in modo da accettare molte mosse di peggioramento per scappare dai minimi locali per poi calare lentamente fino a 0 (momento nel quale tutte le mosse peggiorative vengono rigettate).

Vediamo quindi l'algoritmo:

- **inizializzazione:** si sceglie un punto iniziale  $x_c$ , un massimo numero di iterazioni  $N$  e un valore minimo di temperatura  $T_{min}$ . Si pone quindi  $n = 0$  e  $z_c = f(x_c)$
- **iterazione:**
  - si sceglie un nuovo candidato  $x_n$  in modo random nell'intorno di  $x_c$  e se ne calcola  $z_n = f(x_n)$
  - si ha:
 

```

          if  $z_n \geq z_c$  then
             $x_{n+1} = x_n$ 
          else
            if  $\text{rand}(0, 1) < e^{\frac{z_n - z_c}{T}}$  then
               $x_{n+1} = x_n$ 
            else
               $x_{n+1} = x_c$ 
          
```
- **controllo di temperatura:** dopo un certo numero di iterazioni con lo stesso valore di  $T$ , diminuire  $T$  e ripetere il procedimento
- **criterio di stop:** numero massimo di iterazioni, massimo tempo computazionale, oppure quando si sono fatte  $N$  iterazioni con la temperatura minima  $T_{min}$

**La soluzione finale è la migliore soluzione ottenuta.**

Come per la Tabù Search anche qui bisogna scegliere varie cose, come la temperatura, il numero di iterazioni etc. ...

Per quanto riguarda la temperatura ci sono dei consigli pratici:

- la temperatura iniziale deve essere tale da poter accettare molte mosse peggiorative (50% delle mosse peggiorative)
- la temperatura va diminuita lentamente ad esempio del 10%
- la temperatura finale deve essere tale da non accettare mosse peggiorative, ovvero  $T \approx 0$

**Teorema 6.** *Simulated Annealing converge asintoticamente al punto di ottimo globale. La velocità di convergenza è influenzata da come viene diminuita la temperatura, a causa del **cooling schedule***

Su slide esempio di uso del metodo per *TSP* e per una *funzione non lineare*

## 6.3 Algoritmi Genetici

Gli algoritmi genetici rappresentano una famiglia di metaeuristiche ispirata ai principi dell'evoluzione genetica di una popolazione. Questi algoritmi simulano l'evoluzione (dove il migliore sopravvive) di una popolazione di individui, che rappresentano le possibili soluzioni del problema di ottimizzazione.

Questi algoritmi appartengono alla classe degli **algoritmi evolutivi**.

Come al solito la soluzione banale consiste nel generare una soluzione in modo casuale e testarne la qualità, fino a trovarne una accettabile. Questa tecnica funziona solo se ci sono poche soluzioni e se si ha abbastanza tempo.

Possiamo migliorare il procedimento testando ogni soluzione. Dopo averle ordinate si eliminano le peggiori e duplicano le migliori (perturbandone leggermente qualcuna), tutto ciò fino ad avere una soluzione migliore accettabile.

Negli algoritmi generici le soluzioni vengono codificate come stringhe di bit a lunghezza fissa dove ogni bit rappresenta un aspetto della soluzione. La nostra funzione obiettivo deve quindi stabilire la bontà delle soluzioni e associarne un valore (tramite una funzione detta **fitness function**). L'insieme delle soluzioni è detto **spazio di ricerca o spazio degli stati**.

Spesso in questi algoritmi i numeri vengono codificati in un numero di bit costante in grado di rappresentare tutti i valori delle varie soluzioni. Queste stringhe vengono chiamate **genotipi o cromosomi** mentre i singoli bit sono detti **geni**. Il valore rappresentato dalla stringa di bit è detto **fenotipo**.

Per migliorare ancora di più il nostro algoritmo aggiungo il meccanismo dei **crossover**, ovvero della riproduzione combinando i geni dei genitori per ottenere nuovi individui. Si selezionano quindi due stringhe di bit dette *genitori*

e si combinano con una certa probabilità, detta **tasso di crossover**. Si ottengono 2 discendenti **offspring**, che possono subire una mutazione casuale. Tutto questo processo prende spunto dalla teoria evolutiva di Darwin, sui concetti di *riproduzione* e *selezione naturale*. Si ha che i figli preservano in parte i caratteri genetici (i cromosomi) dei genitori attraverso il *crossover*. Si ha una maggiore probabilità che le generazioni successive presentino i caratteri degli individui più adatti (con fitness più alta) all'ambiente in cui vivono (evoluzione). Si ha la seguente tabella per la corrispondenza tra concetti biologici e concetti di problemi di calcolo:

Individuo	→	Soluzione del problema
Popolazione	→	Insieme di soluzioni
Fitness	→	Qualità di una soluzione
Cromosoma	→	Rappresentazione di una soluzione
Gene	→	Componente di una rappresentazione
Cross-over/Mutazione	→	Operatori per la ricerca di soluzioni
Selezione naturale	→	Riutilizzo di buone soluzioni
Evoluzione	→	Ricerca di buone soluzioni

e in un algoritmo evoluzionistico si hanno le seguenti componenti:

- strategia di inizializzazione
- popolazione
- rappresentazione (codifica)
- funzione di valutazione (fitness function)
- meccanismo di selezione (dei genitori)
- operatori di cross-over
- condizione di arresto

e lo schema di un generico algoritmo evoluzionistico è il seguente:

1. inizializzazione della popolazione
2. valutazione del fitness della popolazione

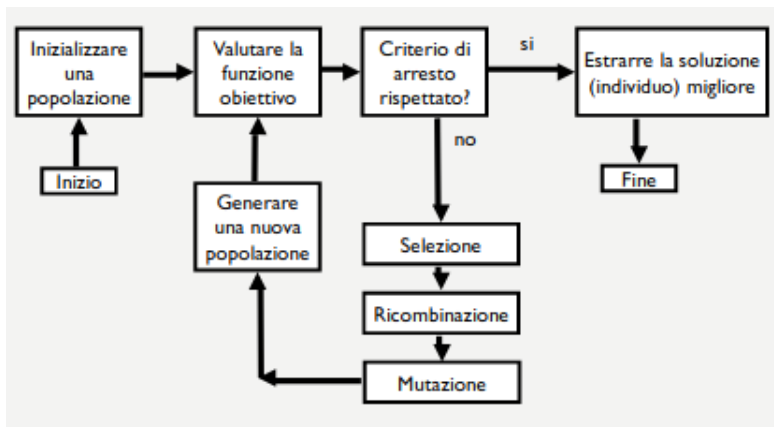
3. ciclare fino a soddisfare il criterio d'arresto con le seguenti operazioni:

- (a) selezionare un sottoinsieme della popolazione
- (b) ricombinare i geni degli individui selezionati, con ricombinazione o crossover
- (c) modificare in modo casuale la popolazione ottenuta mediante mutazione
- (d) valutare la fitness della nuova popolazione

con i seguenti criteri di arresto:

- raggiungimento di soluzioni con alto fitness
- crescita del fitness ridotta (ovvero si stanno ottenendo individui molto simili)
- superamento di un numero  $n$  di iterazioni

graficamente si ha:



Si opera quindi solo sul genotipo e si necessita di una decodifica da genotipo a fenotipo.

Il risultato quindi può essere il migliore dell'ultima generazione o il migliore tra tutti gli individui generati all'inizio del processo (anche se quest'ultima soluzione non è ammissibile biologicamente).

Si ha la **Codifica di Gray** (una codifica binaria in un numero fisso di bit) dove i numeri da 0 a  $2^n - 1$  sono codificati con una stringa binaria di lunghezza  $n$ . Si ha quindi che interi consecutivi differiscono di un solo bit e una qualsiasi inversione di bit produce piccoli cambiamenti. Grandi cambiamenti però risultano maggiori che rispetto alla normale codifica binaria.

### 6.3.1 Funzione di Fitness e Spazio di Ricerca

La funzione di fitness  $f$  valuta la bontà degli individui  $g$  della popolazione  $P$ . Si ha quindi che  $f > 0$ , sempre. Si ha inoltre che  $f$  esiste per ogni individuo e rappresenta una misura della bontà di una soluzione e quindi va massimizzata. Possiamo quindi definirla come:

$$f : P \rightarrow \mathbb{R}$$

per comodità assumiamo questa scrittura:

$$f(g_i) = f_i$$

Analizziamo ora lo spazio di ricerca. Si ha che l'insieme di stringhe binarie di lunghezza  $l$  ha  $2^l$  elementi. Questo insieme è appunto il nostro spazio di ricerca, dove l'algoritmo genetico deve esplorare per risolvere il problema di ricerca. I valori di fitness sui punti dello spazio di ricerca formano il **fitness landscape**.

Si possono codificare anche funzioni multidimensionali e lo spazio di ricerca può essere visualizzato come una superficie (fitness landscape) in cui la fitness determina l'altezza. Quindi ogni genotipo rappresenta un punto nello spazio. Le performances degli algoritmi genetici dipendono dalla natura dello spazio di ricerca e quindi uno spazio completamente casuale non è d'aiuto, in quanto l'algoritmo si può incastrare in ottimi locali, per esempio. Di contro uno spazio di ricerca dove con piccoli miglioramenti ci si avvicina alla soluzione ottima è teoricamente perfetto.

### 6.3.2 Popolazione

Una popolazione è un *multiset* (ovvero un insieme in cui è accettabile la presenza di più copie di uno stesso elemento) di soluzioni. Una popolazione è sempre caratterizzata dalla sua **dimensione** (ovvero dal numero dei suoi individui), che resta quasi sempre costante tra le varie generazioni. Eventualmente può essere caratterizzata anche da una struttura di tipo spaziale in cui ogni individuo ha un insieme di vicini in un suo intorno. Si definisce **diversità di una popolazione** come il numero diverso di individui al suo intorno.

Vediamo un esempio di popolazione:

No.	Cromosoma	Fitness
1	1010011010	1
2	1111100001	2
3	1011001100	3
4	1010000000	1
5	0000010000	3
6	1001011111	5
7	0101010101	1
8	1011100111	2

### 6.3.3 Selezione e Operatori Genetici

Dopo che la funzione di fitness ha determinato la bontà di ogni individuo su genera una nuova popolazione di individui attraverso i cosiddetti **operatori di Holland**:

- *selezione* (ispirato alla selezione naturale)
- *crossover* (ispirato alla genetica)
- *mutazione* (ispirato alla genetica)

gli ultimi due sono detti anche **operatori genetici**.

#### Selezione

Si hanno diverse strategie di selezione dl punto di vista algoritmico (spesso non biologicamente plausibili). In un algoritmico genetico si hanno solitamente due situazioni:

1. un gruppo di soluzioni è selezionato per l'accoppiamento (**matingpool**)
2. coppie di individui sono estratti a caso dal matingpool e vengono accoppiati (**riproduzione**)

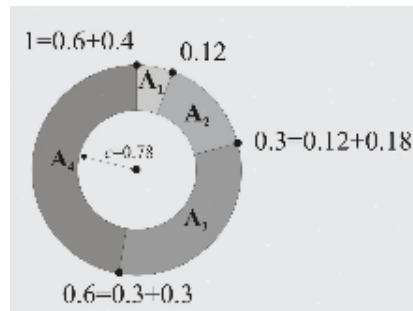
Ogni volta che un individuo della popolazione è selezionato ne viene creata una copia; tale copia è inserita nel così detto matingpool. Quando il matingpool è riempito con esattamente  $n$  (numero di individui della popolazione)

copie di individui della popolazione, nuovi  $n$  discendenti sono creati applicando gli operatori genetici.

Holland invece propose un metodo di soluzione proporzionale al valore di fitness. Sia quindi  $f_i$  il valore di fitness del genotipo  $g_i$ . La probabilità che  $g_i$  sia selezionato per la riproduzione è:

$$p_{s,i} = \frac{f_i}{\sum f_i}$$

con tali probabilità sono usate per costruire una sorta di roulette dove ogni individuo occupa uno spicchio di ampiezza pari alla probabilità appena calcolata. Si genera quindi un valore da 0 a 1 e si seleziona l'individuo che occupa quella parte di roulette:



Un'altra strategia consiste nell'usare un array di lunghezza  $N$  stabilita dalle frazioni che rappresentano le probabilità, per esempio se ho  $p_1 = \frac{1}{6}$ ,  $p_2 = \frac{1}{4}$  e  $p_3 = \frac{5}{12}$  scelgo  $N = 12$ , riempirlo in gli indici dei gli individui in modo proporzionale alla probabilità calcolata sopra, estrarre un numero a caso e selezionare l'individuo corrispondente al valore nella posizione estratta.

Una terza tecnica consiste nell'estrarre un numero casuale  $r$  con  $0 \leq r \leq \sum_j f_j$  tale che:

$$\sum_{j=1}^{i-1} f_j \leq r < \sum_{j=1}^i f_j$$

La tecnica della selezione proporzionale presenta soprattutto due difetti:

1. **convergenza prematura**, ovvero se un individuo  $i$  ha fitness molto maggiore della media della popolazione ma molto minore della massima fitness possibile, tenderà ad essere sempre selezionato e quindi a generare una popolazione mediocre
2. **stagnazione**, ovvero dopo un certo numero di generazioni, tutti gli individui hanno una buona fitness e quindi tendono ad avere la stessa probabilità di essere selezionati

Si hanno anche altri tipi di selezione:

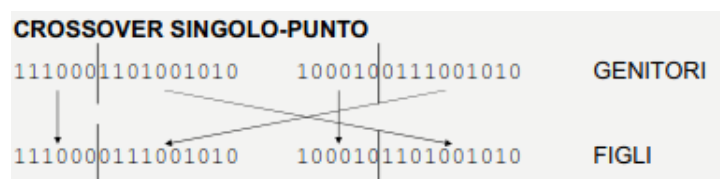
- **selezione per rango**, dove si ordinano gli individui in base al fitness (in ordine decrescente) e si impone una distribuzione di probabilità decrescente con la posizione occupata indipendentemente dai valori di fitness. Non si ha convergenza prematura perché nessun individuo ha probabilità molto maggiore di essere selezionato e non c'è stagnazione in quanto la distribuzione di probabilità non varia. In compenso è computazionalmente molto pesante
- **selezione tramite torneo**, dove per ogni individuo da selezionare si seleziona un gruppo di individui e si clona il migliore. L'operazione consiste quindi nel scegliere  $k$  individui in modo random, con o senza rimpiazzamento, e selezionare i migliori di questi  $k$  confrontando le loro fitness con maggiore probabilità per selezionare i migliori. Ha un vantaggio rispetto alla selezione per rango: non usa ordinamenti
- **selezione elitista**, dove almeno una copia dell'individuo migliore viene mantenuta nella generazione successiva. Questo comporta che non si perdono soluzioni buone nei processi casuali di selezione ma di contro si rimane incastrati in ottimi locali se i caratteri di un individuo diventano dominanti

## Crossover

Come già detto è un'operazione in cui i figli vengono generati ricombinando il materiale genetico degli individui che costituiscono il matingpool e consente di ottenere nuovi individui il cui codice genetico è mutuato in parte da un genitore e in parte dall'altro.

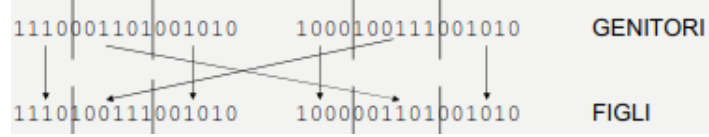
Si hanno due possibili rappresentazioni:

1. **rappresentazione binaria**, che sfrutta la rappresentazione con 0 e 1 e si divide in tre categorie:
  - (a) **crossover singolo punto**, dove si seleziona un punto a caso all'interno del genoma e si scambiano le due sezioni destre o sinistre:

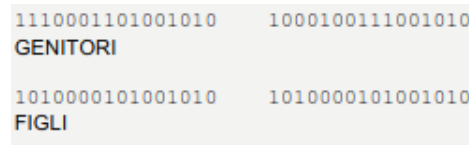




- (b) **crossover doppio-punto**, dove si fanno 2 *tagli* e si scambiano le 2 parti interne o le due parti esterne ad essi:



- (c) **crossover uniforme**, dove ogni bit è selezionato a caso da uno dei genitori:



*Gli stessi operatori possono essere utilizzati nel caso in cui si usi una rappresentazione basata su stringhe di interi*

2. **rappresentazione floating-point**, dove anziché selezionare i diversi elementi copiandoli da uno dei genitori  $x$  e  $y$ , combinano i loro valori facendone una somma pesata. Si hanno tre tipi di ricombinazioni:

- (a) **ricominazione semplice**, dove si sceglie un punto  $k$  nella stringa e un valore  $a < 1$ . I due figli si ottengono così:

- *figlio 1*: si ha:

$$x_1, \dots, x_k, ay_{k+1} + (a-1)x_{k+1}, \dots, ay_N + (1-a)x_N$$

- *figlio 2*: come il figlio 1 invertendo  $x$  e  $y$

- (b) **ricombinazione singola**, dove si sceglie un punto  $k$  nella stringa e un valore  $a < 1$ . I due figli si ottengono così:

- *figlio 1*: si ha:

$$x_1, \dots, ay_k + (1-a)x_k, x_{k+1}, \dots, x_N$$

- *figlio 2*: come il figlio 1 invertendo  $x$  e  $y$

- (c) **ricombinazione completa**, dove si sceglie un punto  $k$  nella stringa e un valore  $a < 1$ . I due figli si ottengono così:

- *figlio 1*: si ha:

$$ay + (1 - a)x$$

- *figlio 2*: si ha:

$$ax + (1 - a)y$$

SI ha poi il cosiddetto **crossover per permutazioni**, che vediamo nell'implementazione dei **partially-mapped crossover**, **PMX**. Si hanno sei steps:

1. si scelgono 2 punti nella stringa  $G_1$  e si copiano in  $F_1$  i valori fra essi compresi.  
Per esempio  $G_1 = 123456789$ ,  $G_2 = 937826514$ :  
 $F_1 = \dots \underline{4567}..$
2. Dal primo punto si guarda quali elementi di  $G_2$  compresi fra i due punti selezionati non sono stati ancora copiati
3. per ciascuno di questi  $i$  si guarda in  $F_1$  quale elemento  $j$  compare nella posizione corrispondente
4. si pone  $i$  nella posizione occupata da  $j$  in  $G_2$ :  
 $F_1 = \dots \underline{4567}.8$
5. e il posto occupato da  $j$  in  $G_2$  è già stato occupato in  $F_1$  da un elemento  $k$ , poni  $i$  nella posizione occupata da  $k$  in  $G_2$ :  
 $F_1 = ..\underline{24567}.8$
6. gli altri elementi di  $F_1$  vengono copiati direttamente da  $G_2$ :  
 $F_1 = 932\underline{4567}18$

**Il figlio  $F_2$  è creato in modo analogo invertendo i G.**

### Mutazione

La mutazione è un operatore che mira a mantenere diversità genetica per cercare di esplorare anche zone dello spazio di ricerca non presenti nella popolazione attuale. Si hanno tre rappresentazioni possibili:

1. **mutazione per rappresentazioni binarie**, dove si sceglie a caso un bit nel genotipo e lo si inverte

2. **mutazione per rappresentazioni intere**, dove è possibile sostituire un gene con un valore random ammissibile oppure aggiungere o rimuovere una certa quantità a quel valore, con probabilità più elevata in un intorno di zero
3. **mutazione per rappresentazioni floating point**, dove dato un vettore  $\langle x_1, \dots, x_n \rangle$ ,  $x_i \in [L_i, U_i]$  si passa a  $\langle x'_1, \dots, x'_n \rangle$ ,  $x'_i \in [L_i, U_i]$  attraverso un processo di sostituzione dei valori con una componente random. **capire che insieme è**

tra i tipi di mutazione elenchiamo:

- **mutazione uniforme**, dove si sostituiscono tutti gli elementi della rappresentazione con un valore random appartenente allo stesso intervallo di definizione
- **mutazione non uniforme con distribuzione fissa**, dove si ottiene il nuovo vettore aggiungendo ad ogni elemento un numero random appartenente ad una distribuzione centrata in zero e decrescente col valore assoluto, per esempio una gaussiana  $G(0, \sigma)$