

Ricerca Operativa

a.a. 2018-2019



SAPIENZA
UNIVERSITÀ DI ROMA

Lavinia Amorosi
Lezione XXIII

Algoritmi di Ricerca Locale

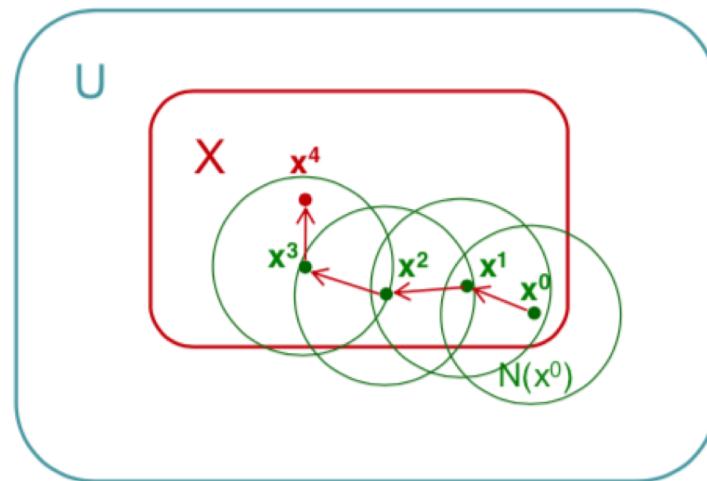
Algoritmo di Ricerca Locale

SCHEMA DI ALGORITMO DI RL:

Metodo **iterativo** ispirato alla filosofia dei miglioramenti successivi, **a breve termine** (algoritmo di discesa), o **a lungo termine** (**euristiche** di RL).

L'algoritmo termina producendo una soluzione finale che è un **ottimo locale** del problema (non necessariamente un ottimo globale).

Caso continuo



ALGORITMO DI RL:

- intorno
- soluzione iniziale
- mossa
- arresto

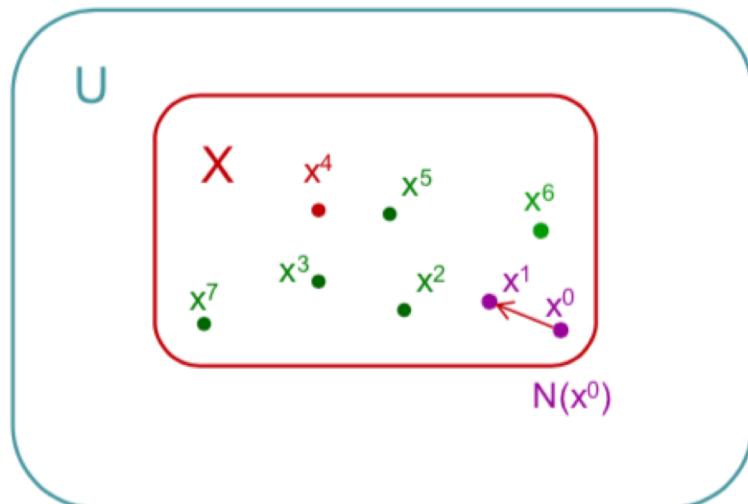
Algoritmo di Ricerca Locale

SCHEMA DI ALGORITMO DI RL:

Metodo **iterativo** ispirato alla filosofia dei miglioramenti successivi, **a breve termine** (algoritmo di discesa), o **a lungo termine** (euristiche di RL).

L'algoritmo termina producendo una soluzione finale che è un **ottimo locale** del problema (non necessariamente un ottimo globale).

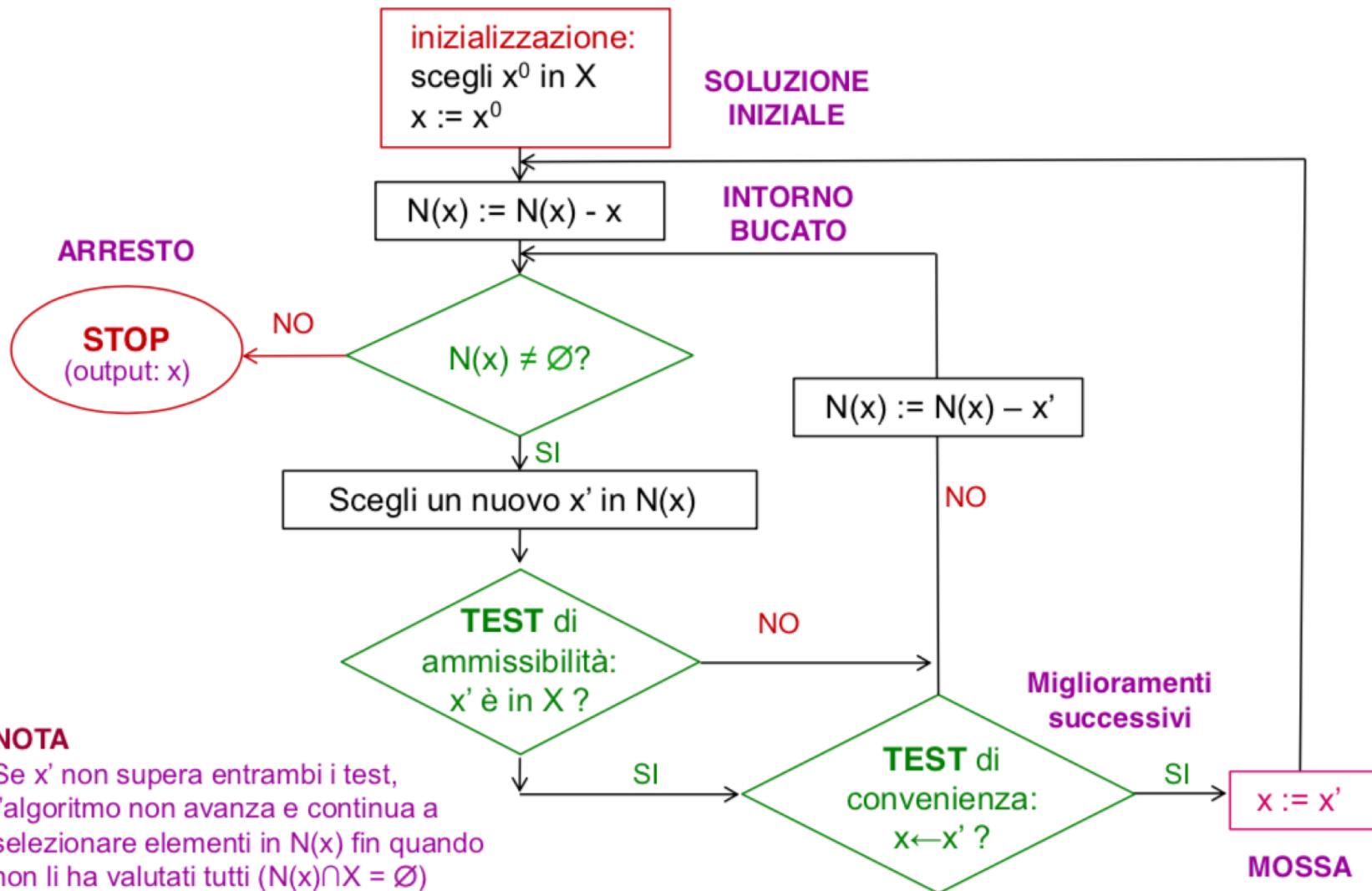
Caso discreto



ALGORITMO DI RL:

- intorno
- soluzione iniziale
- mossa
- arresto

Paradigma generale di algoritmo di RL



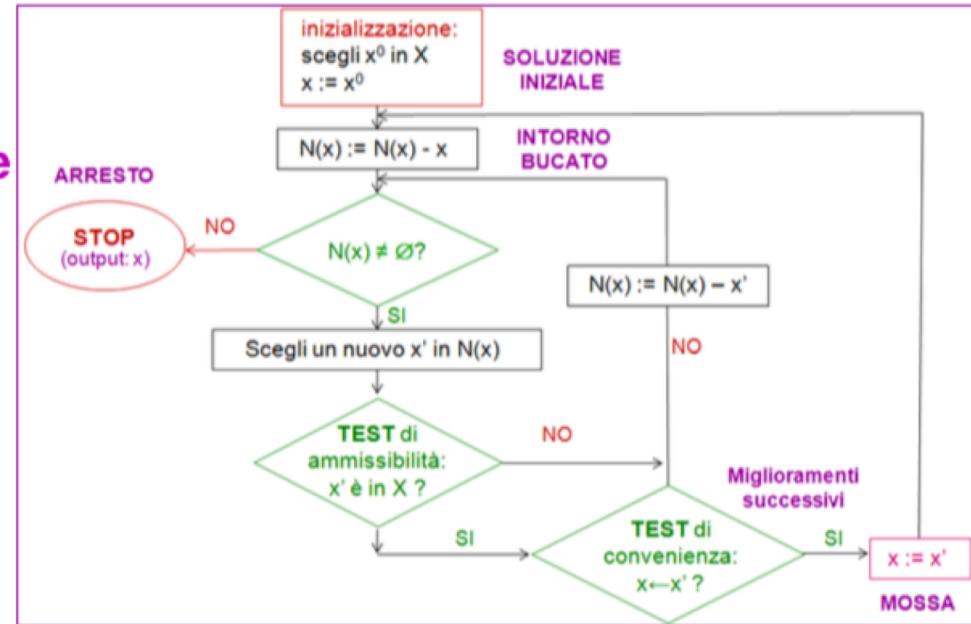
Progettazione di un algoritmo di RL

Nello schema generale di algoritmo di RL ci sono varie operazioni che si possono implementare utilizzando tecniche di tipo diverso:

1. Regola di scelta della soluzione iniziale.
2. Struttura e dimensione degli intorni.
3. Visita dell'intorno.
4. Scelta della mossa (TEST di convenienza).
5. Regola di arresto.

NOTA

- La diversa implementazione di queste operazioni conduce a **diversi algoritmi di RL**.
- Ciascuna di queste scelte **condiziona la performance** dell'algoritmo di RL.



Per semplicità
indichiamo l'intorno
bucato con $N(x)$.

Progettazione di un algoritmo di RL

5. Regola di arresto

Durante l'esecuzione,
l'algoritmo può arrestarsi in
momenti diversi e per motivi
diversi.

Per arrestare l'algoritmo si
possono implementare **una o**
più condizioni di arresto.

Arresto forzato

Arresto naturale

Per max num. M di iterazioni raggiunto

Si può imporre che l'algoritmo si arresti
dopo **M iterazioni**: la soluzione finale sarà
la soluzione associata all'ultima iterazione.

Per insuccesso: $N(x) \cap X = \emptyset$

In questo caso l'algoritmo termina perché
ha analizzato tutte le soluzioni in $N(x) \cap X$
(senza individuare miglioramenti): la
soluzione corrente sarà la migliore
“localmente”.

Progettazione di un algoritmo di RL

2. Struttura e dimensione degli intorni

Struttura

La struttura dell'intorno dipende dal problema di ottimizzazione specifico che si sta analizzando (PL, PNL, PLI, problemi su grafi, ecc.) e dalla topologia dello spazio delle sue soluzioni.

Dimensione

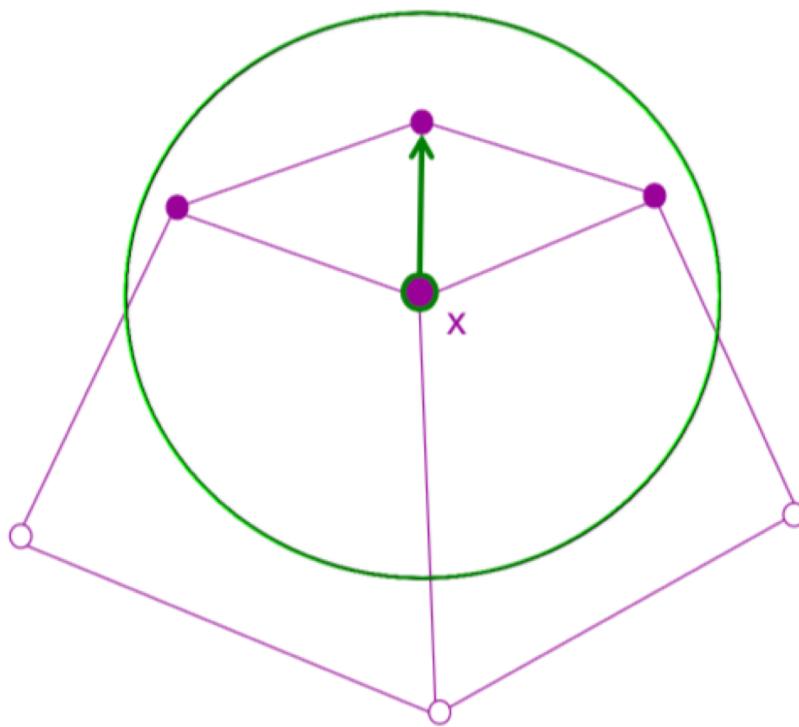
La dimensione dell'intorno dipende dalla sua struttura.
Possono essere previsti intorni di dimensioni variabili (*Variable Neighborhood Search*).

Programmazione Lineare

(1)

$$\min f(x)$$
$$x \in X$$

\mathbb{R}^3

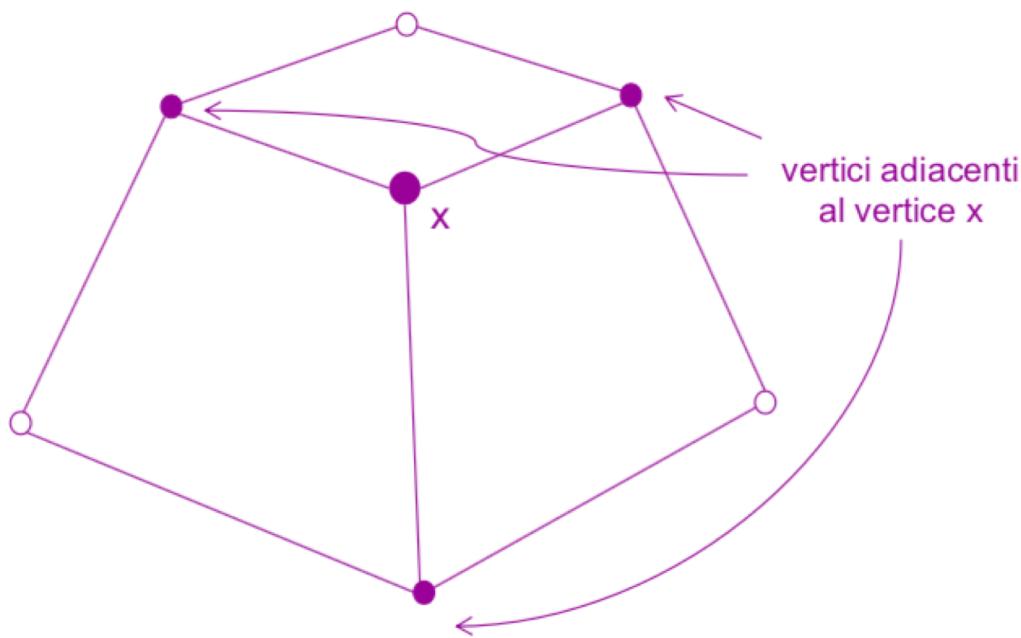


$X = P$ **poliedro ammissibile** (insieme **continuo**).

$N(x) =$ insieme dei punti di P nella **sfera** di centro x e raggio fissato.

Programmazione Lineare

$$(1) \quad \min f(x)$$
$$x \text{ in } X$$

$$\mathbb{R}^3$$


X= **insieme dei vertici** del poliedro ammissibile (insieme **discreto**).

N(x) = insieme dei **vertici “adiacenti”** al vertice x.

Programmazione Lineare

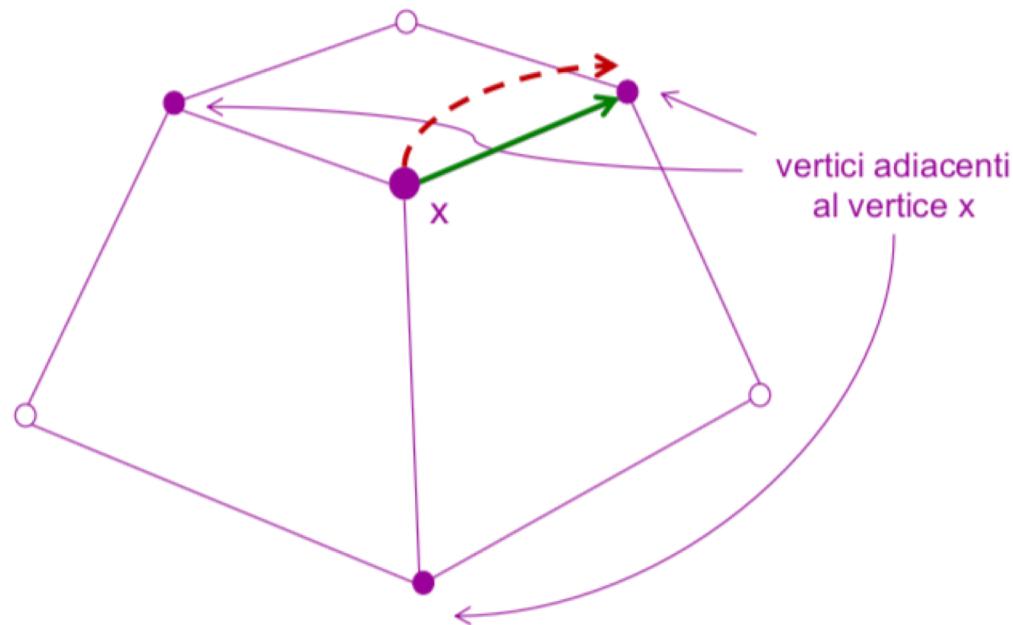
(1)

$$\min f(x) \\ x \text{ in } X$$

\mathbb{R}^3

NOTA

I vertici adiacenti a x possono essere raggiunti sia con uno spostamento continuo che con uno spostamento discreto



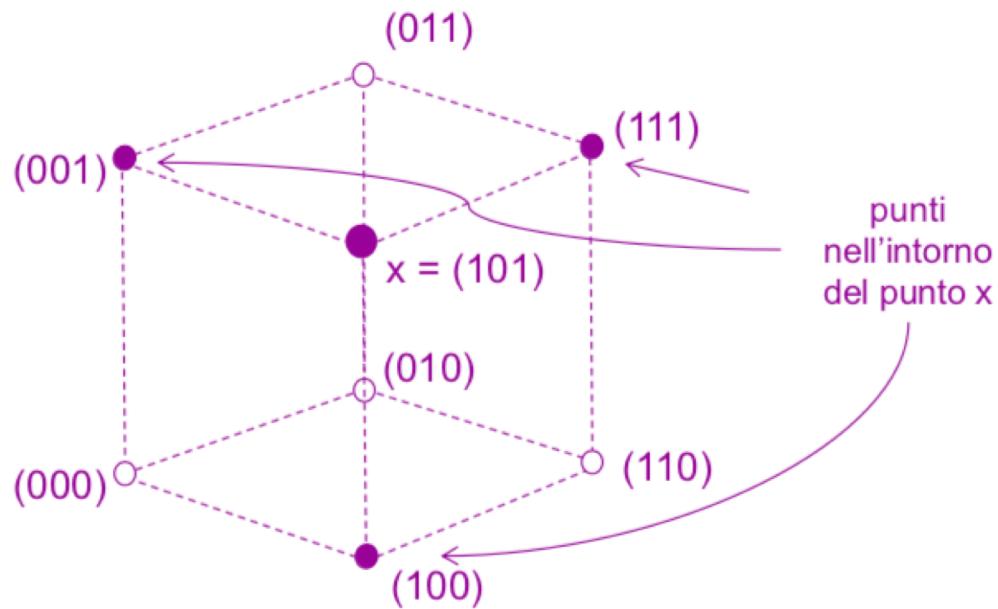
$X = \text{insieme dei vertici}$ del poliedro ammissibile (insieme **discreto**).

$N(x) = \text{insieme dei vertici "adiacenti" al vertice } x.$

Ottimizzazione binaria (Esempio 1)

$$(1) \quad \begin{array}{l} \min f(x) \\ x \text{ in } X \end{array}$$

\mathbb{R}^3



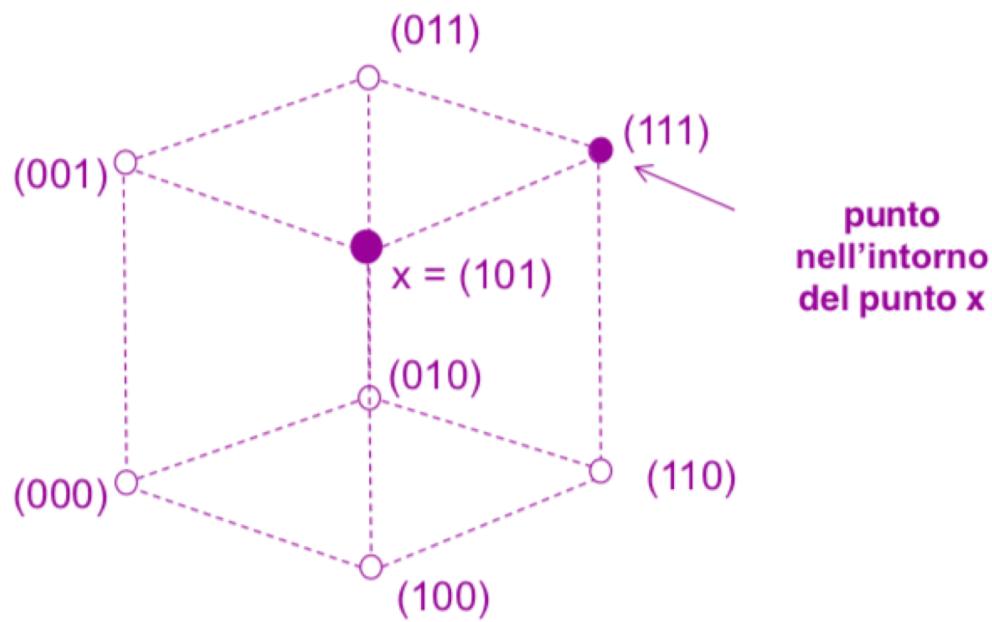
$X = B^n$, dove $B = \{0,1\}$

$N(x) = \{ y \text{ in } B^n : d(x,y) = 1\}$, dove $d(x,y) = | \{ i : x_i \neq y_i \} |$ **Distanza di Hamming**

Ottimizzazione binaria (Esempio 2)

$$(1) \quad \min f(x) \\ x \text{ in } X$$

\mathbb{R}^3



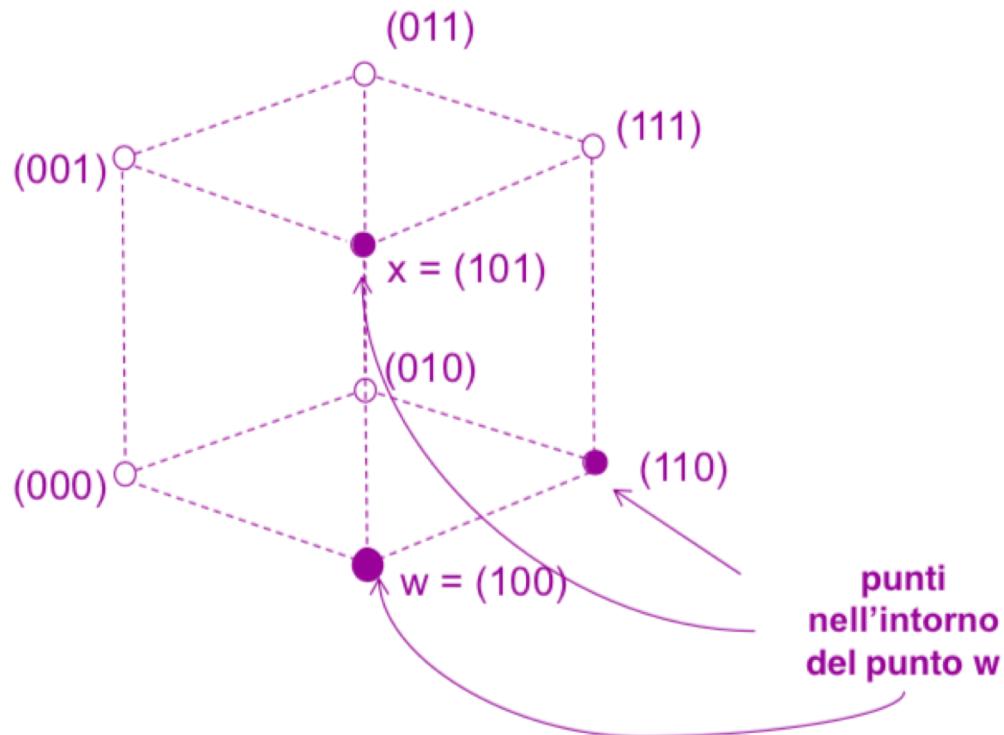
$X = B^n$, dove $B = \{0,1\}$

$N(x) = \{ y \text{ in } B^n : \sum y_i = \sum x_i + 1, y_i = x_i, \text{ per ogni } x_i = 1 \}$

Ottimizzazione binaria (Esempio 2)

(1) $\min f(x)$
 $x \text{ in } X$

\mathbb{R}^3



$X = B^n$, dove $B = \{0,1\}$

$N(x) = \{ y \text{ in } B^n : \sum y_i = \sum x_i + 1, y_i = x_i, \text{ per ogni } x_i = 1\}$

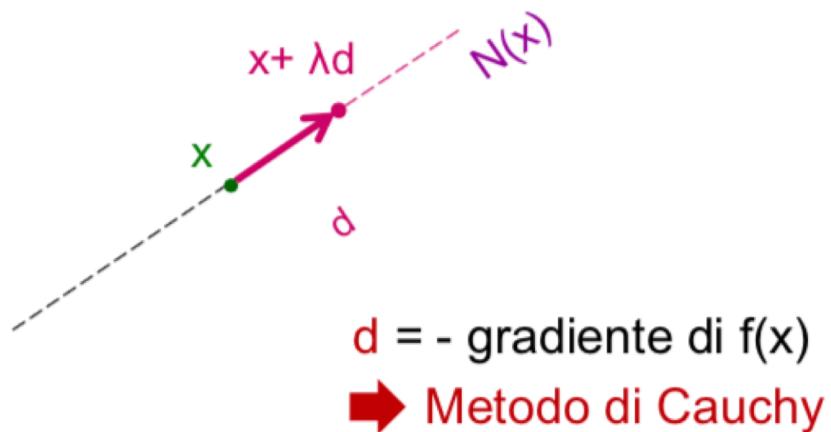
Ottimizzazione non lineare **non vincolata**

$$(1) \quad \begin{aligned} & \min f(x) \\ & x \text{ in } X \end{aligned}$$

L'intorno è una semiretta uscente da x

METODI ITERATIVI

$$\begin{cases} x^0 \in \mathbb{R}^n \\ x^{k+1} = x^k + \lambda^k d^k \end{cases} \quad \begin{array}{l} \text{inizio} \\ \text{iterazione } k \end{array}$$



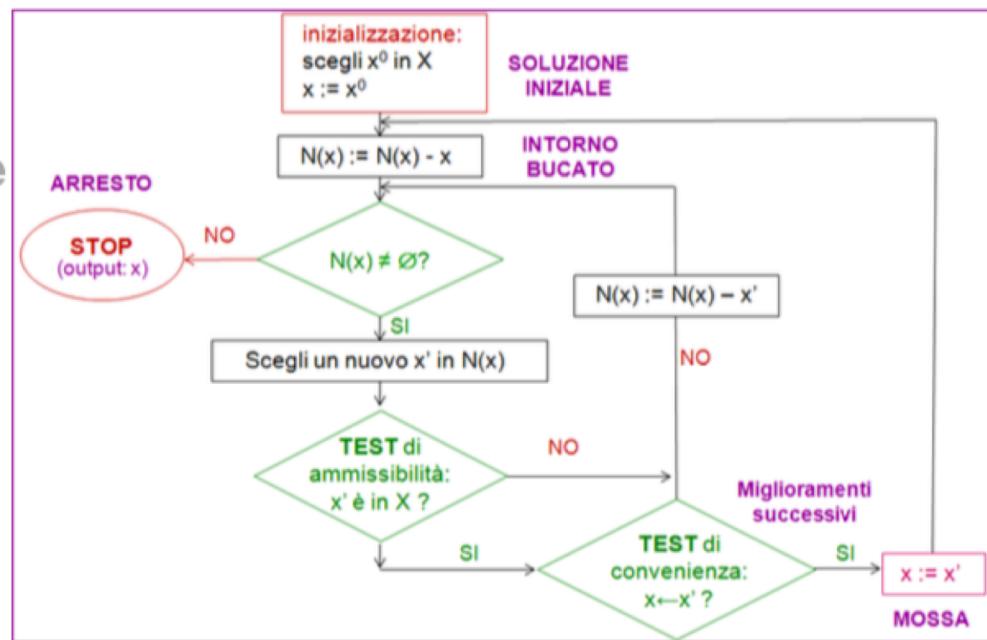
$$X = \mathbb{R}^n$$

$$N(x) = \{x + \lambda d: \lambda \geq 0\}$$

Progettazione di un algoritmo di RL

Nello schema generale di algoritmo di RL ci sono varie operazioni che si possono implementare utilizzando tecniche di tipo diverso:

1. Regola di scelta della soluzione iniziale.
2. Struttura e dimensione degli intorni.
3. Visita dell'intorno.
4. Scelta della mossa (TEST di convenienza).
5. Regola di arresto.



NOTA

- La diversa implementazione di queste operazioni conduce a **diversi algoritmi di RL**.
- Ciascuna di queste scelte **condiziona la performance** dell'algoritmo di RL.

Per semplicità
indichiamo l'intorno
bucato con $N(x)$.

Progettazione di un algoritmo di RL

3. Visita dell'intorno di x

Scelta RANDOM

La nuova soluzione x' in $N(x) \cap X$ viene scelta in maniera **casuale**.

Scelta FIRST

Si ordinano le soluzioni in $N(x) \cap X$ e si selezionano le x' in maniera sistematica **seguendo l'ordine** (ad esempio, numerando le soluzioni).

Scelta BEST

Tra tutte le soluzioni in $N(x) \cap X$ si sceglie quella che produce il **massimo miglioramento nella f.o.**

NOTA

In questo caso si fa già il **test di convenienza** al momento della scelta di x' .

Progettazione di un algoritmo di RL

4. TEST di convenienza

Il test di convenienza caratterizza la **strategia migliorativa** dell'algoritmo di RL:

DISCESA → miglioramento **nel breve**

GREEDY → **massimo** migl. in $N(x) \cap X$

TOLLERANZA → miglioramento **nel lungo**

NOTA

Tra gli algoritmi di RL che accettano peggioramenti occasionali e temporanei della f.o., citiamo: →

- SIMULATED ANNEALING
- TABU SEARCH
- THRESHOLD ACCEPTANCE

• DISCESA

Si sceglie x' tale che
 $f(x') < f(x)$

Il valore della f.o. si riduce.

• GREEDY (GHIOTTO)

Si sceglie x' tale che
 $f(x') = \min \{f(y) : y \in N(x) \cap X\} < f(x)$
→ **Algoritmo ghiotto**

• TOLLERANZA

Si sceglie x' tale che
 $f(x') \leq f(x) + \delta$

(quindi anche se $f(x') > f(x)$)

Si accetta la possibilità di un **temporaneo (e contenuto) degrado** della f.o. per evitare l'arresto "prematuro" per insuccesso.

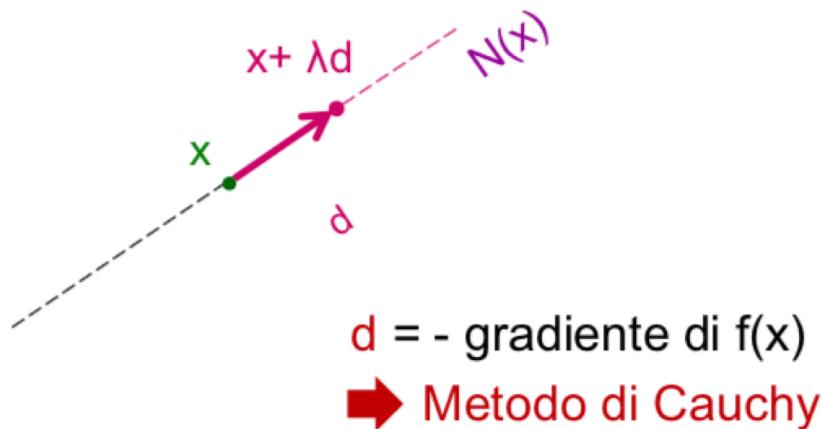
Ottimizzazione non lineare **non vincolata**

$$(1) \quad \min_{x \in X} f(x)$$

L'intorno è una semiretta uscente da x

METODI ITERATIVI

$$\begin{cases} x^0 \in \mathbb{R}^n & \text{inizio} \\ x^{k+1} = x^k + \lambda^k d^k & \text{iterazione k} \end{cases}$$



$X = \mathbb{R}^n$

$N(x) = \{x + \lambda d: \lambda \geq 0\}$

Strategia ghiotta



Se $x^{k+1} = x^k + \lambda^k d^k$ è scelto in $N(x)$ in modo tale che:

$$f(x^k + \lambda^k d^k) = \min \{ f(x^k + \lambda d^k): \lambda \geq 0 \}$$

➡ Line-search

Algoritmo generale di Discesa

INPUT: $\{\min f(x), x \in X\}; x^0 \in X$

OUTPUT: $x^* \in X$ (ottimo locale)

Passo 0 (inizializzazione)

$x := x^0 \in X$

Passo 1 (arresto)

Se $N(x) \cap X$ è vuoto

STOP: $x^* := x$

Altrimenti andare al **Passo 2**

Passo 2 (avanzamento)

Selezionare una x' in $N(x) \cap X$

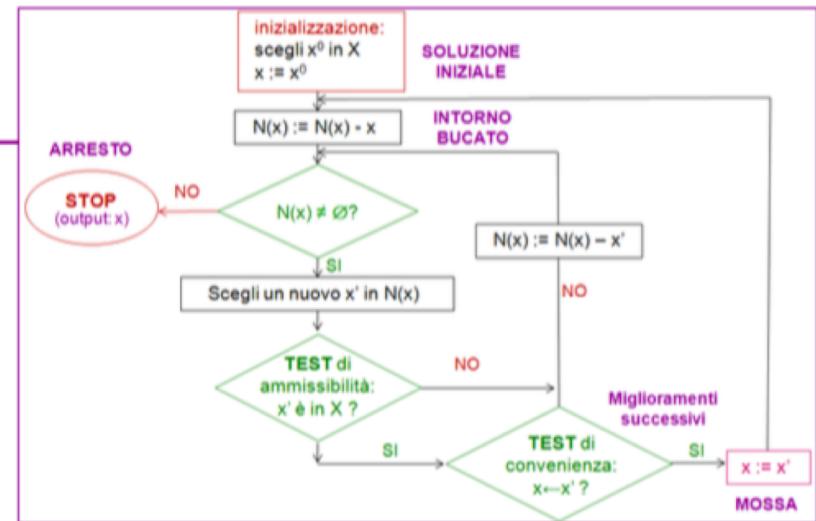
$N(x) := N(x) - \{x'\}$

Se $f(x') < f(x)$

$x := x'$

$x^* := x$

Tornare al **Passo 1**



NOTA

Il paradigma di algoritmo di RL **non specifica** alcuna delle operazioni citate sopra, e, dunque, è assolutamente **generale**.

Algoritmo di Discesa per la PL

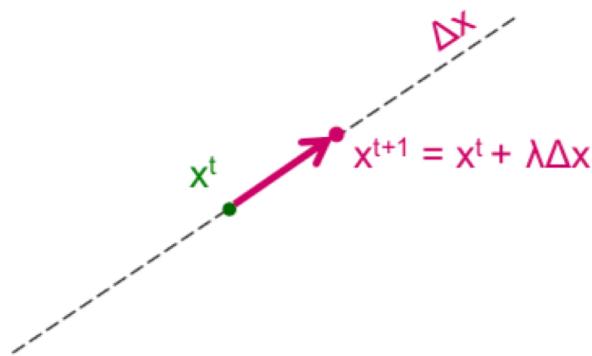
NOTA

Il paradigma di algoritmo di RL visto non specifica alcuna delle operazioni citate sopra, e, dunque, è assolutamente generale.

Nel caso di problemi di ottimizzazione nello spazio continuo, tali operazioni danno luogo al paradigma di ricerca detto “**passo-direzione**”:

Avanzamento da x^t a x^{t+1} di un passo $\lambda > 0$ lungo la direzione

$$\Delta x = x^{t+1} - x^t$$



Direzioni ammissibili e direzioni di discesa

Direzioni di discesa in R^n

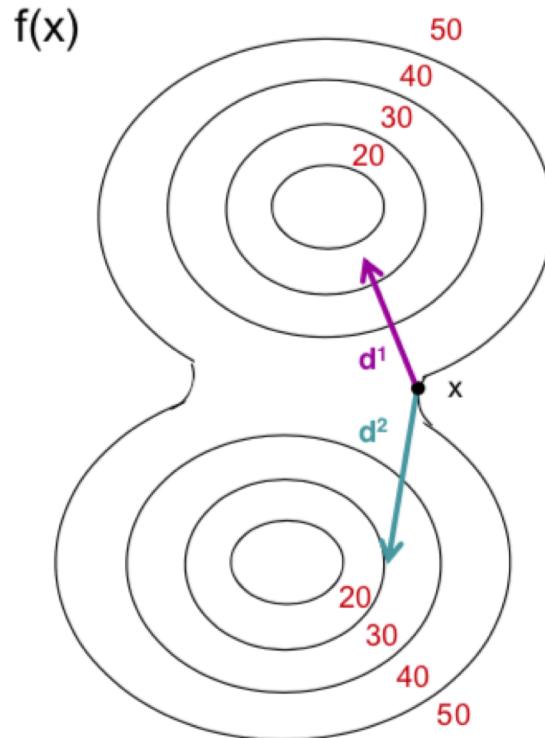
Definizione

Data $f: R^n \rightarrow R$ e $x \in R^n$, una direzione $d \in R^n$, $d \neq 0$, si dice di discesa (miglioramento) per $f(x)$ in x se esiste $\lambda^* > 0$ tale che:

$$f(x + \lambda d) < f(x) \quad \text{per ogni } \lambda \in (0, \lambda^*)$$

Esempio

- d^1 direzione di discesa per $f(x)$ in x
- d^2 una direzione di discesa per $f(x)$ in x
può non esserlo per passi molto grandi



Direzioni di discesa in R^n

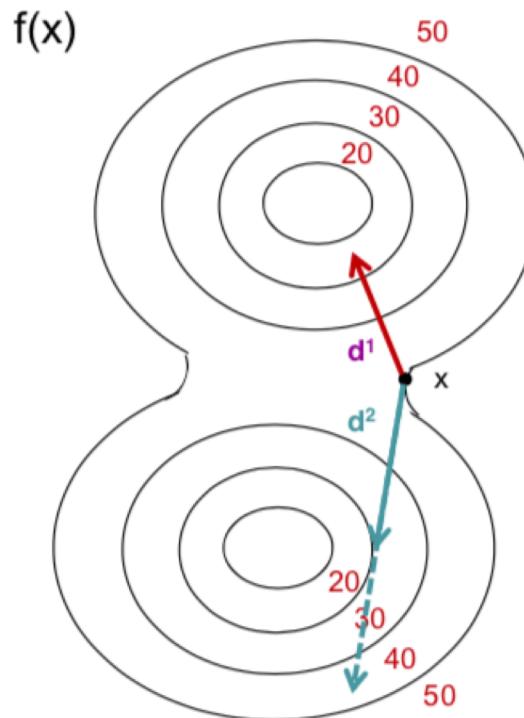
Definizione

Data $f: R^n \rightarrow R$ e $x \in R^n$, una direzione $d \in R^n$, $d \neq 0$, si dice di discesa (miglioramento) per $f(x)$ in x se esiste $\lambda^* > 0$ tale che:

$$f(x + \lambda d) < f(x) \quad \text{per ogni } \lambda \in (0, \lambda^*)$$

Esempio

- d^1 direzione di discesa per $f(x)$ in x
- d^2 una direzione di discesa per $f(x)$ in x
può non esserlo per passi molto grandi



Direzioni di discesa in \mathbb{R}^n

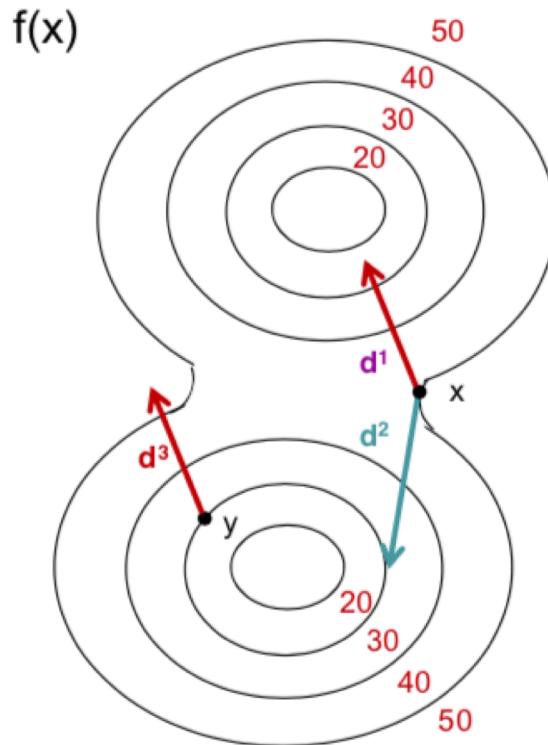
Definizione

Data $f: \mathbb{R}^n \rightarrow \mathbb{R}$ e $x \in \mathbb{R}^n$, una direzione $d \in \mathbb{R}^n$, $d \neq 0$, si dice **di discesa** (miglioramento) **per $f(x)$ in x** se esiste $\lambda^* > 0$ tale che:

$$f(x + \lambda d) < f(x) \quad \text{per ogni } \lambda \in (0, \lambda^*)$$

Esempio

- d^1 direzione di discesa per $f(x)$ in x
- d^2 una direzione di discesa per $f(x)$ in x
può **non esserlo per passi molto grandi**
- d^3 una direzione di discesa per $f(x)$ in x ,
può **non esserlo per $f(x)$ in y**



Direzioni di discesa in R^n

Definizione

Data $f: R^n \rightarrow R$ e $x \in R^n$, una direzione $d \in R^n$, $d \neq 0$, si dice di discesa (miglioramento) per $f(x)$ in x se esiste $\lambda^* > 0$ tale che:

$$f(x + \lambda d) < f(x) \quad \text{per ogni } \lambda \in (0, \lambda^*)$$

Esempio

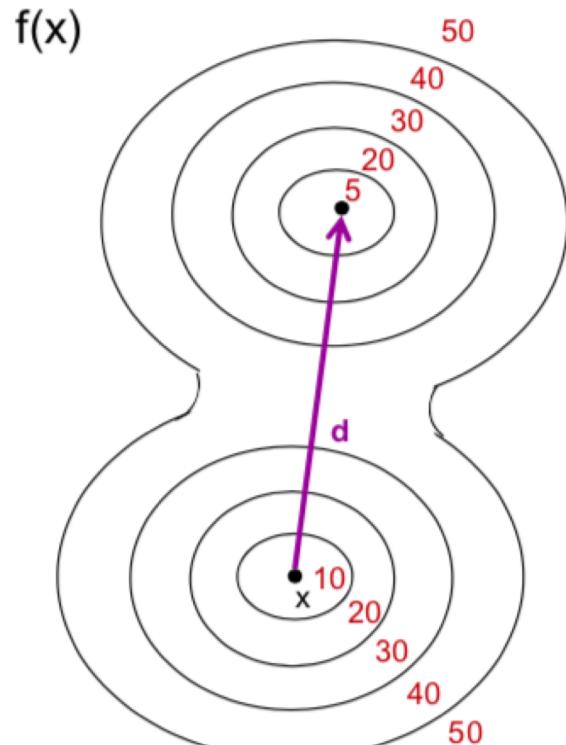
La direzione **d** non è una direzione di discesa in x .

In x non esistono direzioni di discesa.

→ **x è ottimo locale**

NOTA

In un punto di minimo locale non possono esistere direzioni di discesa.



Direzioni ammissibili in R^n

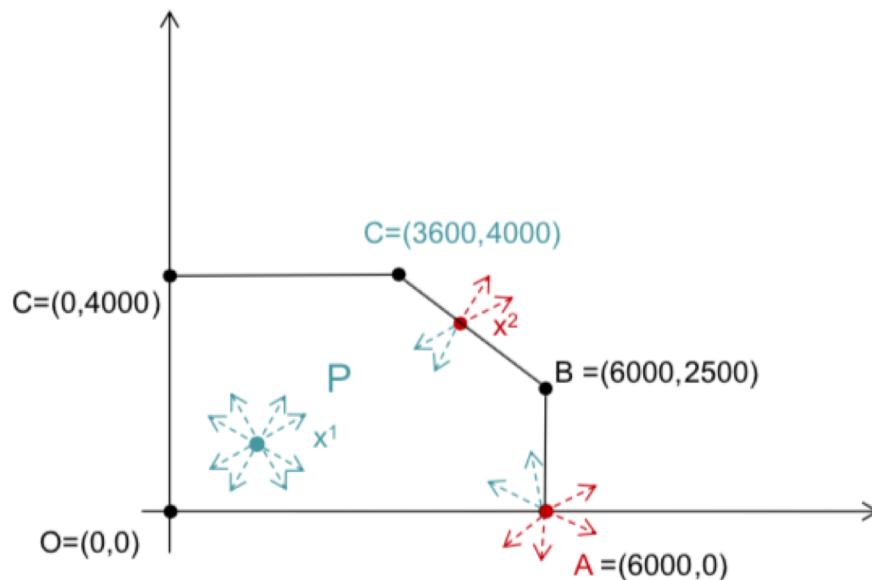
Definizione

Dato un sottoinsieme S di R^n e $x \in S$, una direzione $d \in R^n$, $d \neq 0$, si dice **ammissibile per S in x** se esiste $\lambda^* > 0$ tale che:

$$x + \lambda d \in S \quad \text{per ogni } \lambda \in [0, \lambda^*].$$

Esempio

Consideriamo $S = P$



Direzioni ammissibili in R^n

Definizione

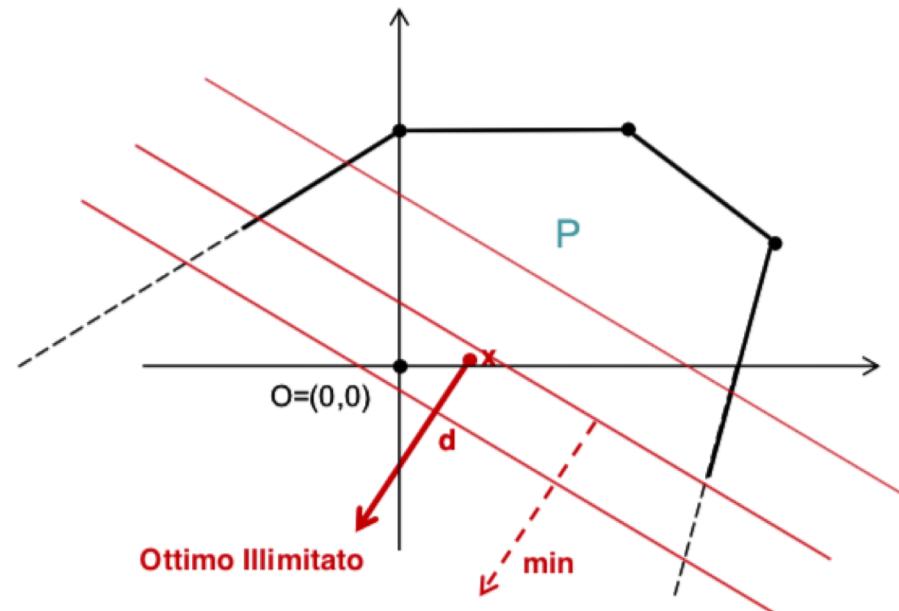
Dato un sottoinsieme S di R^n e $x \in S$, una direzione $d \in R^n$, $d \neq 0$, si dice **ammissibile per S in x** se esiste $\lambda^* > 0$ tale che:

$$x + \lambda d \in S \quad \text{per ogni } \lambda \in [0, \lambda^*].$$

Osservazione

Sia P il poliedro delle soluzioni ammissibili di un problema di PL. Può accadere che una direzione d sia ammissibile per P in x per ogni $\lambda^* > 0$.

Se in un punto x del poliedro ammissibile P di un problema di PL la direzione d è ammissibile per P per ogni $\lambda^* > 0$ ed è di discesa per $f(x) = c^T x$, allora si ha un ottimo illimitato.



Algoritmo di Discesa (per la PL)

ALGORITMO DI DISCESA PER LA PL (PARADIGMA PASSO/DIREZIONE)

Input: Problema di PL (di minimo) con regione ammissibile X , $x^0 \in X$

Output: Ottimo locale/globale $x^* \in X$



Algoritmo generale di Discesa

INPUT: $\{\min f(x), x \in X\}; x^0 \in X$

OUTPUT: $x^* \in X$ (ottimo locale)

Passo 0 (inizializzazione)

$x := x^0 \in X$

Passo 1 (arresto)

Se $N(x) \cap X$ è vuoto

STOP: $x^* := x$

Altrimenti andare al **Passo 2**

Passo 2 (avanzamento)

Selezionare una x' in $N(x) \cap X$

$N(x) := N(x) - \{x'\}$

Se $f(x') < f(x)$

$x := x'$

aggiornare $N(x)$

Tornare al **Passo 1**

NOTA

Il paradigma di algoritmo di RL non specifica alcuna delle operazioni citate sopra, e, dunque, è assolutamente generale.

Algoritmo generale di Discesa

L'algoritmo di discesa genera una sequenza di soluzioni ammissibili:

$$x^0, x^1, \dots, x^t, \dots$$

tale che

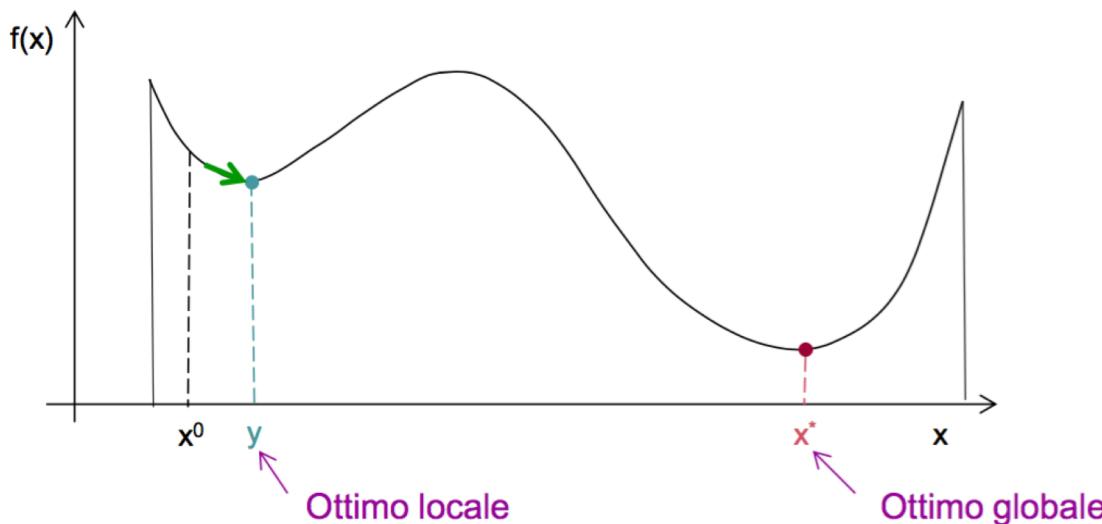
$$f(x^0) > f(x^1) > \dots > f(x^t) > \dots$$

Vantaggio

Nessuna soluzione è ripetuta (e si migliora sempre).

Svantaggio

Si rischia di rimanere “intrappolati” in un ottimo locale
(picco nel caso di **max**, valle nel caso di **min**).



Algoritmo generale di Discesa

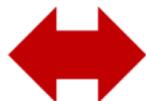
L'algoritmo di discesa genera una sequenza di soluzioni ammissibili:

$$x^0, x^1, \dots, x^t, \dots$$

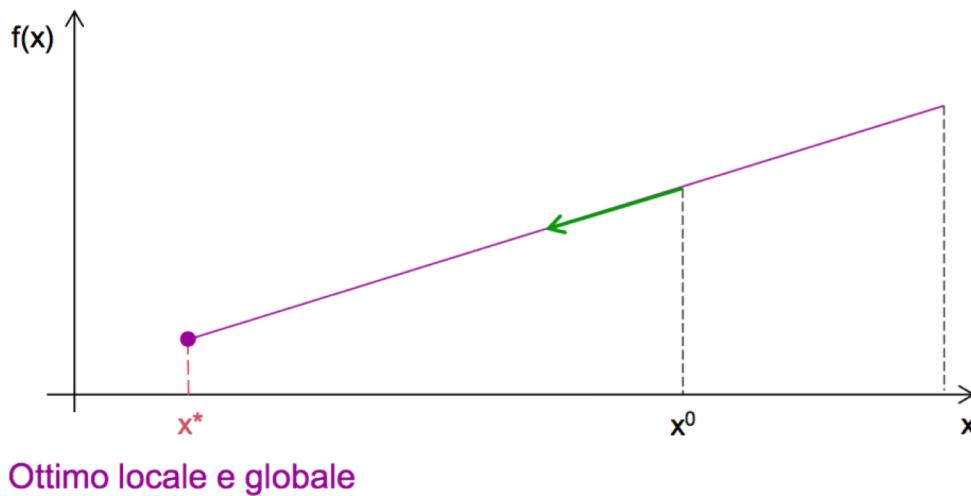
tale che

$$f(x^0) > f(x^1) > \dots > f(x^t) > \dots$$

La funzione obiettivo di un problema di **PL** non ha picchi né valli.



Un problema di **PL** ha solo ottimi globali.

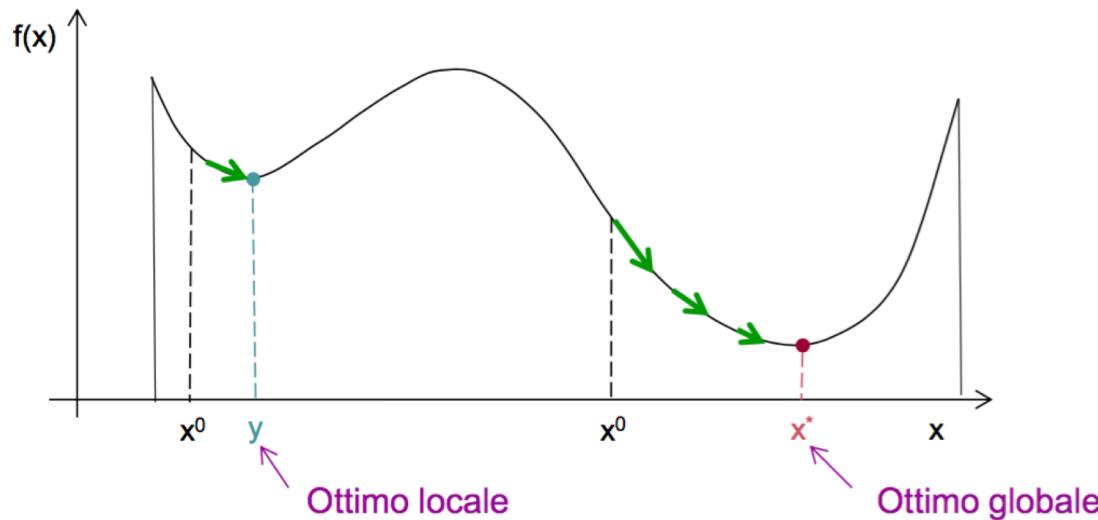


Algoritmo generale di Discesa

Strategie per evitare di rimanere intrappolati in un ottimo locale:

1. seguire un approccio **MULTISTART**

Inizializzando la ricerca **in più punti iniziali diversi**, si **ampliano le capacità di visita dello spazio delle soluzioni** e aumenta la possibilità di individuare un ottimo globale, o, almeno, **un ottimo locale di buona qualità**.

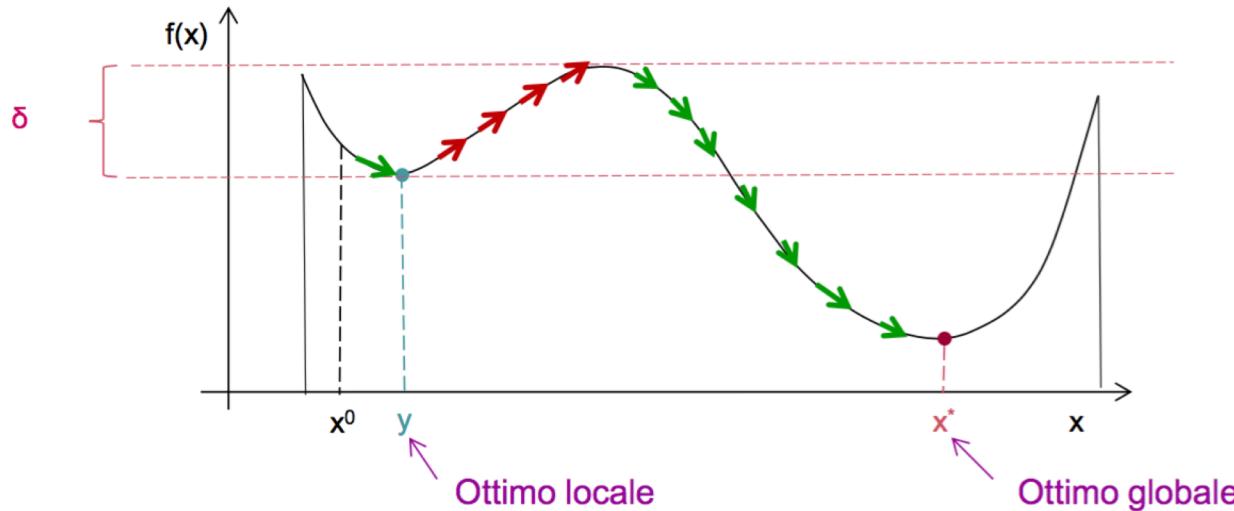


Algoritmo generale di Discesa

Strategie per evitare di rimanere intrappolati in un ottimo locale:

1. seguire un approccio **MULTISTART**
2. accettare occasionalmente **peggioramenti locali** del f.o. durante la ricerca (miglioramenti nel lungo periodo)

Supponiamo che il **massimo peggioramento accettato sia pari a δ** , allora nell'esempio in figura la strategia di miglioramento nel lungo periodo permette di individuare l'ottimo globale anche partendo da x^0 .

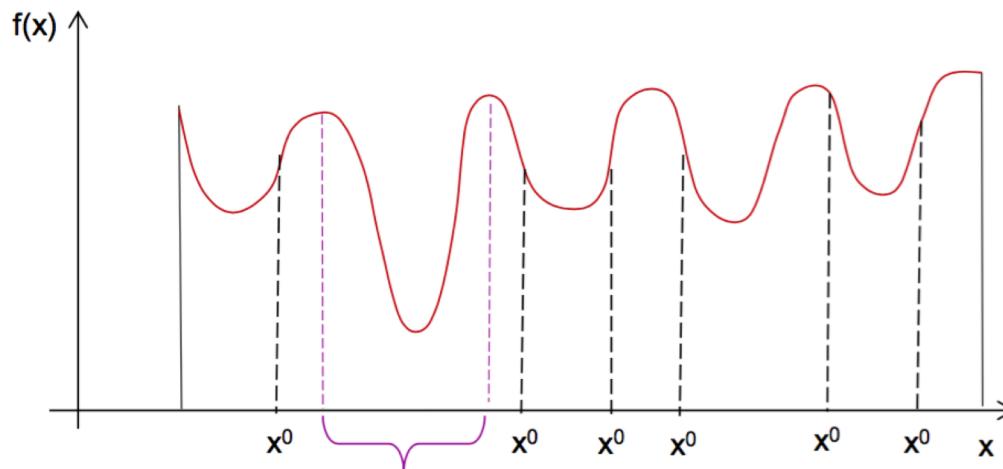


Algoritmo generale di Discesa

Entrambe le strategie possono comunque non avere successo e terminare prematuramente in un ottimo locale di cattiva qualità.

Esempio: MULTISTART

Anche inizializzando ripetutamente la ricerca da punti iniziali diversi è possibile che non si individui l'ottimo globale.



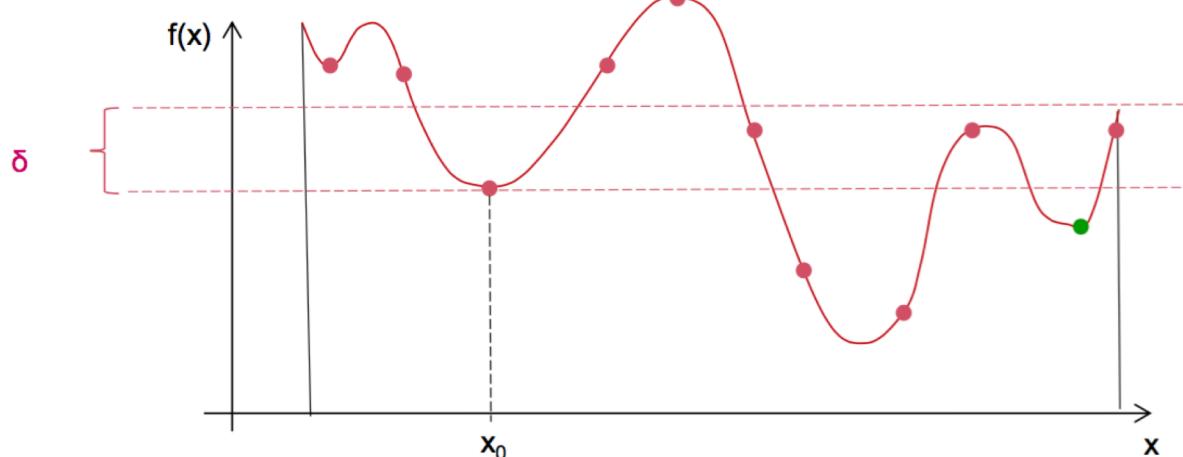
NOTA: L'ottimo globale si individua solo se si parte da un punto "vicino" al punto ottimo

Algoritmo generale di Discesa

Entrambe le strategie **possono comunque non avere successo** e terminare prematuramente in un ottimo locale di cattiva qualità.

Esempio: peggioramenti locali

Anche accettando un peggioramento locale di entità δ è possibile che non si individui l'ottimo globale (caso discreto).



Solo i punti evidenziati sulla curva corrispondono a soluzioni ammissibili.

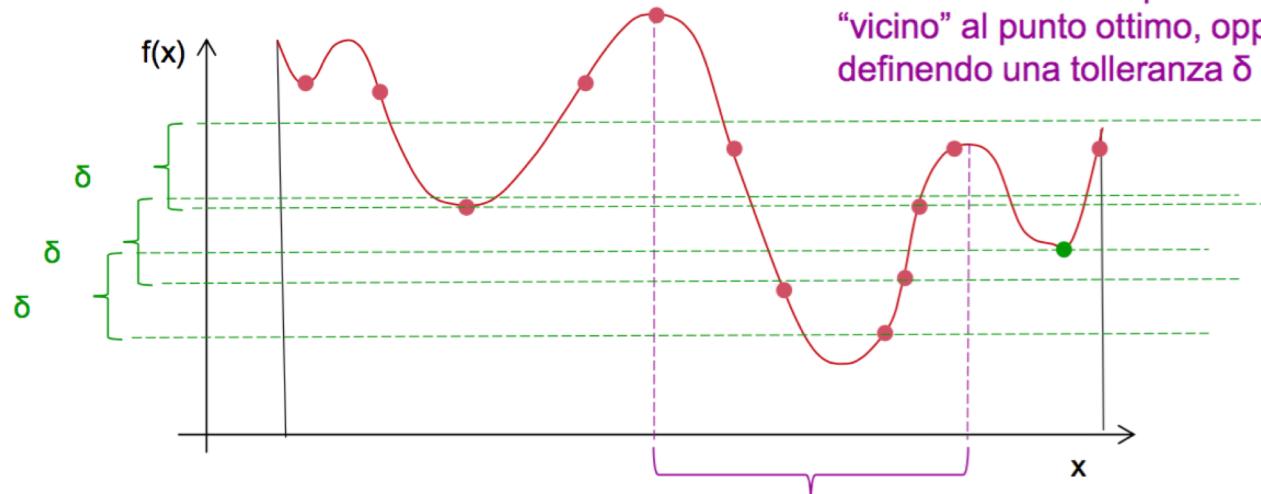
Nell'intorno di x_0 non esiste alcuna soluzione ammissibile con peggioramento al massimo pari a δ .

Algoritmo generale di Discesa

Entrambe le strategie possono comunque non avere successo e terminare prematuramente in un ottimo locale di cattiva qualità.

Esempio: peggioramenti locali

Anche accettando un peggioramento locale di entità δ è possibile che non si individui l'ottimo globale (caso discreto).



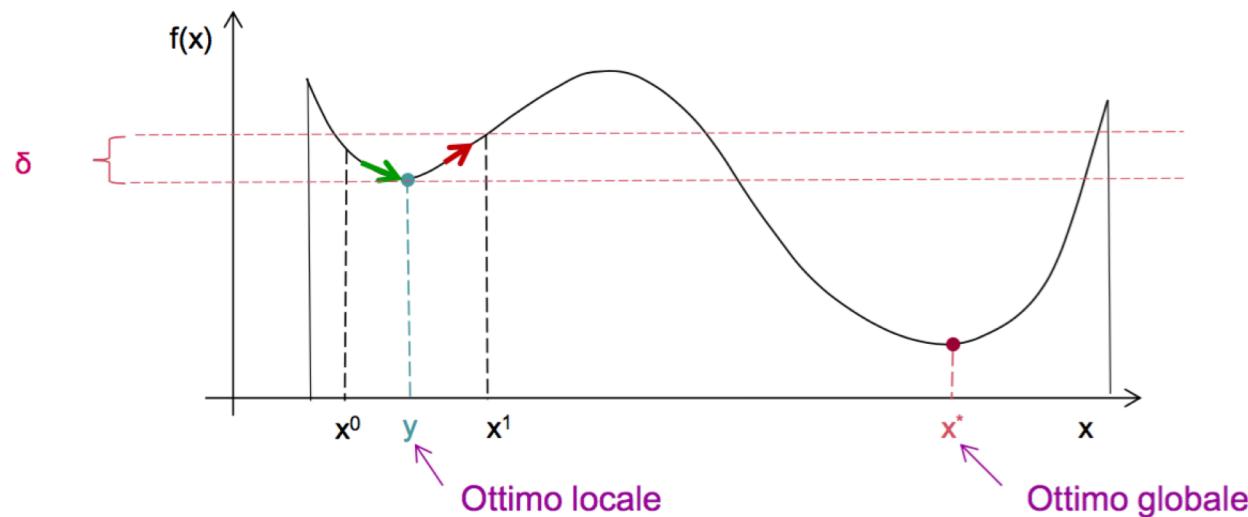
Solo i punti evidenziati sulla curva corrispondono a soluzioni ammissibili.

Algoritmo generale di Discesa

Entrambe le strategie **possono comunque non avere successo e terminare prematuramente in un ottimo locale di cattiva qualità.**

Esempio: peggioramenti locali

Inoltre, seguendo una strategia di questo tipo, c'è il pericolo che si ritorni a **soluzioni già visitate (ciclaggio)**.

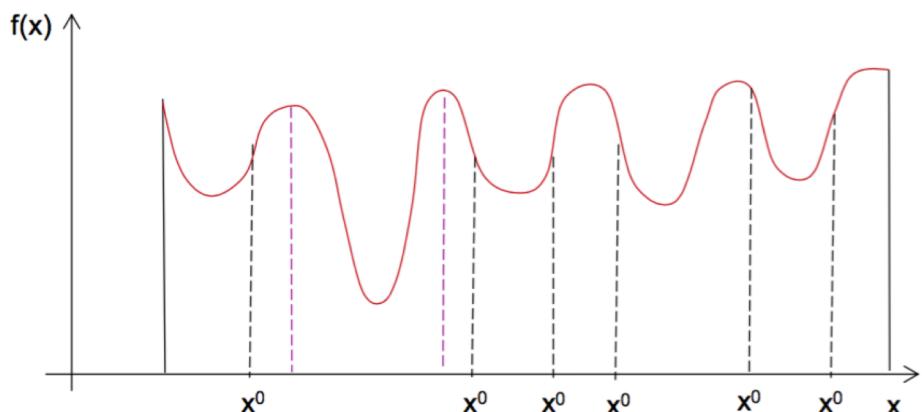


Algoritmo generale di Discesa

L'arresto prematuro in un ottimo locale dell'algoritmo di discesa è tanto più probabile quanti più numerosi sono i picchi (max) o le valli (min) della funzione da ottimizzare.

Picchi e valli corrispondono a punti di ottimo locale che possono **non essere di ottimo globale**.

L'algoritmo di discesa si arresta **al primo picco (o valle)** che individua.



Adottando le **strategie 1. e 2.** si tenta quantomeno di visitare il maggior numero possibile di ottimi locali (diversificazione della ricerca) **cercando di individuarne uno di buona qualità**.

MULTISTART

diversificazione all'**inizio** della ricerca

Peggioramenti locali

diversificazione **durante** la ricerca

Algoritmo generale di Discesa

Gli algoritmi euristici di ricerca locale di nuova generazione permettono la **ripetizione di soluzioni**, ma adottano anche strategie per evitare il ciclaggio.

Le tecniche di questo tipo più diffuse sono:

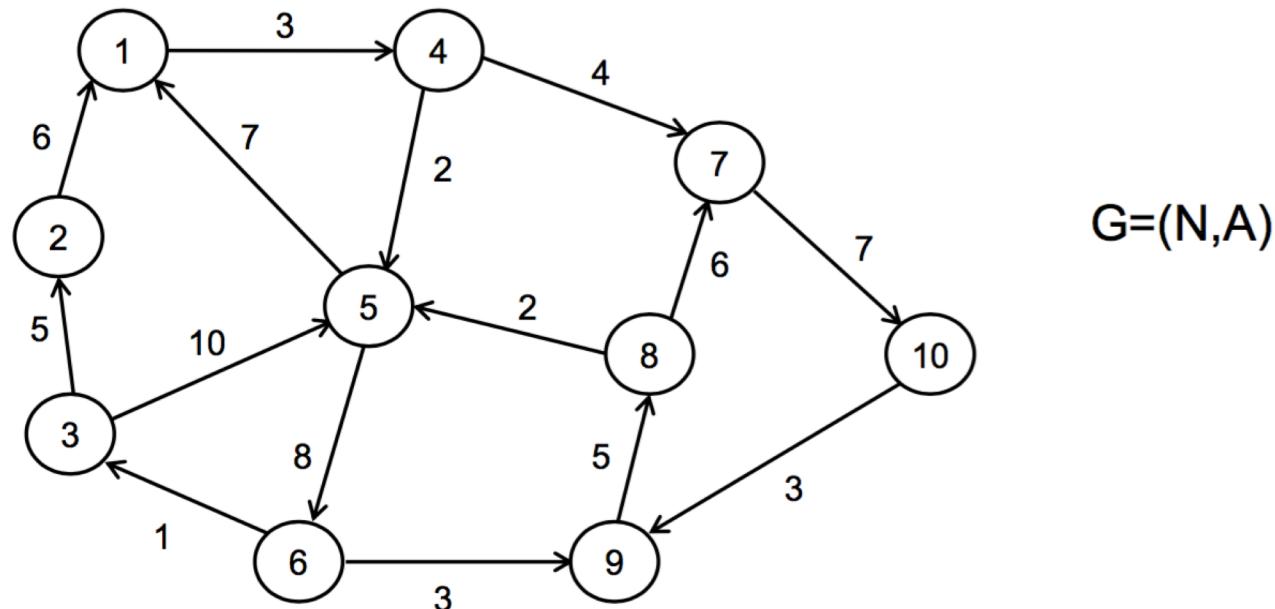
- **Simulated Annealing** (Kirkpatrick et al., 1983; Cerny, 1985)
simulazione del processo termodinamico di ‘ricottura’ (annealing) di materiali, come vetro e metallo, per ottenere materiali solidi con strutture particolarmente resistenti (acciaio)
- **Tabu Search** (Glover, 1989)
memorizzazione temporanea di soluzioni “proibite” (lista tabù)
- **Old Bachelor Acceptance** (Hu et al., 1995)
uso di soglie adattive sui peggioramenti tollerati
- **Algoritmi Genetici** (Holland, 1975, Goldberg, 1989)
simulazione del processo di evoluzione biologica (cromosomi)

Il Problema del Commesso Viaggiatore (Travel Salesman Problem)

Il problema del commesso viaggiatore (TSP)

Un problema di Ottimizzazione Combinatoria molto noto è quello del **Commesso Viaggiatore** o **Travel Salesman Problem (TSP)**.

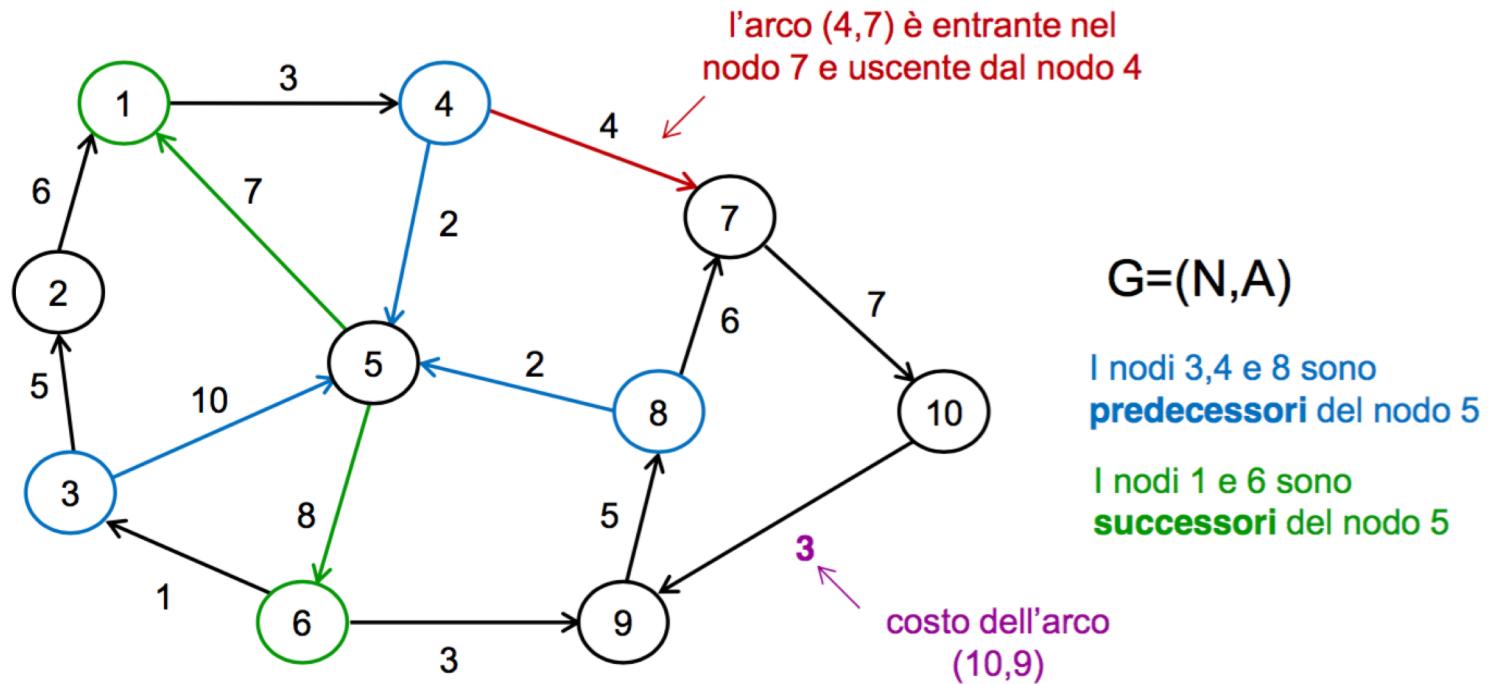
Il problema è definito su un grafo $G = (N, A)$, con $|N|=n$ e $|A| = m$, in cui ad ogni arco $(i,j) \in A$ è associato un **costo** o **lunghezza** $c_{ij} \geq 0$.



Il problema del commesso viaggiatore (TSP)

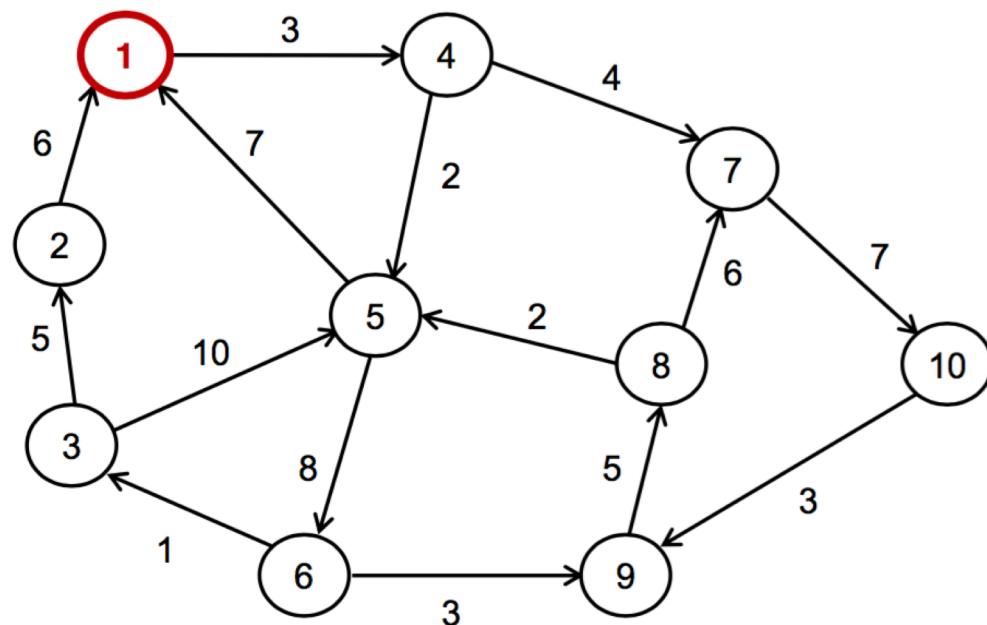
Un problema di Ottimizzazione Combinatoria molto noto è quello del **Commesso Viaggiatore** o **Travel Salesman Problem (TSP)**.

Il problema è definito su un grafo $G = (N, A)$, con $|N|=n$ e $|A| = m$, in cui ad ogni arco $(i,j) \in A$ è associato un costo o lunghezza $c_{ij} \geq 0$.



Il problema del commesso viaggiatore (TSP)

Problema: Un commesso viaggiatore (Travel Salesman), partendo dalla città in cui vive, ogni giorno deve visitare $n=10$ città. Egli deve passare in ciascuna città esattamente una volta e alla fine del percorso deve ritornare nella città di partenza minimizzando il *costo totale degli spostamenti*.

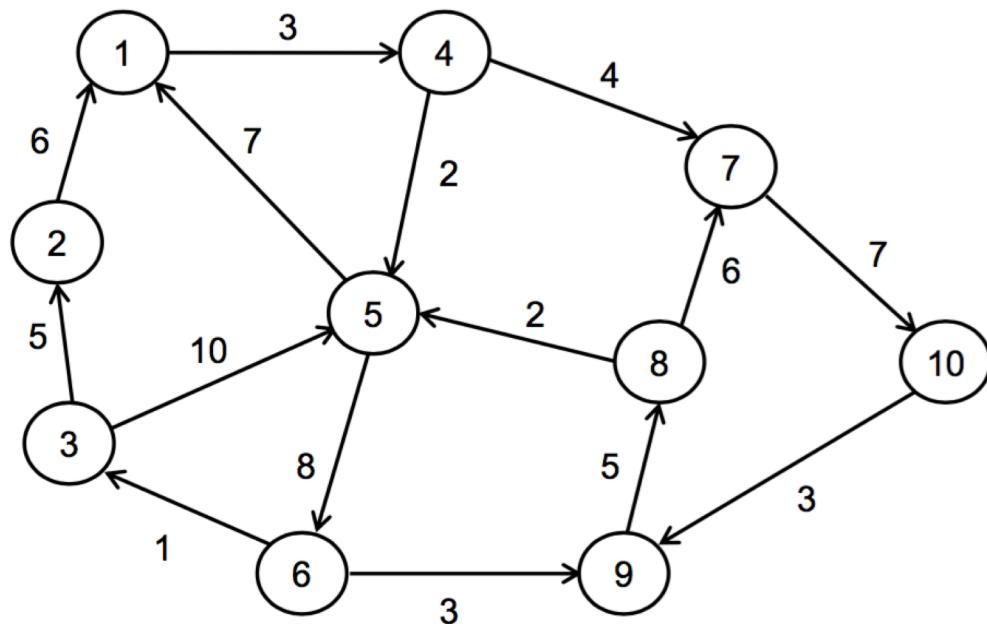


HP: $G=(N,A)$
rappresenta le città
(nodi in N) e i
collegamenti tra città
(archi in A).

Il problema del commesso viaggiatore (TSP)

TOUR

Fissato un nodo di G , ad esempio il nodo 1, un **tour** (o **ciclo hamiltoniano**) è un percorso (sequenza ordinata di nodi-archi) che inizia e termina nel nodo fissato (S.P.I.G. Il nodo 1) e che visita ogni nodo di G una e una sola volta.

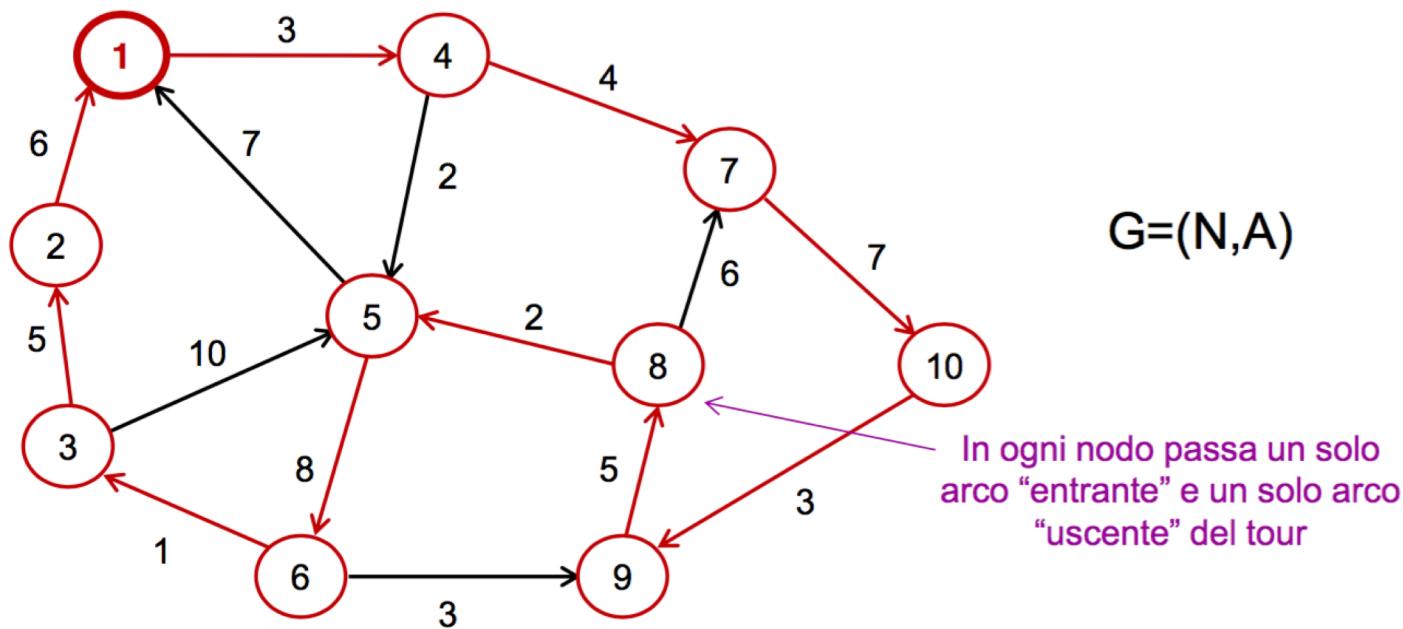


NOTA:
 $G=(N,A)$
deve essere
'connesso'

Il problema del commesso viaggiatore (TSP)

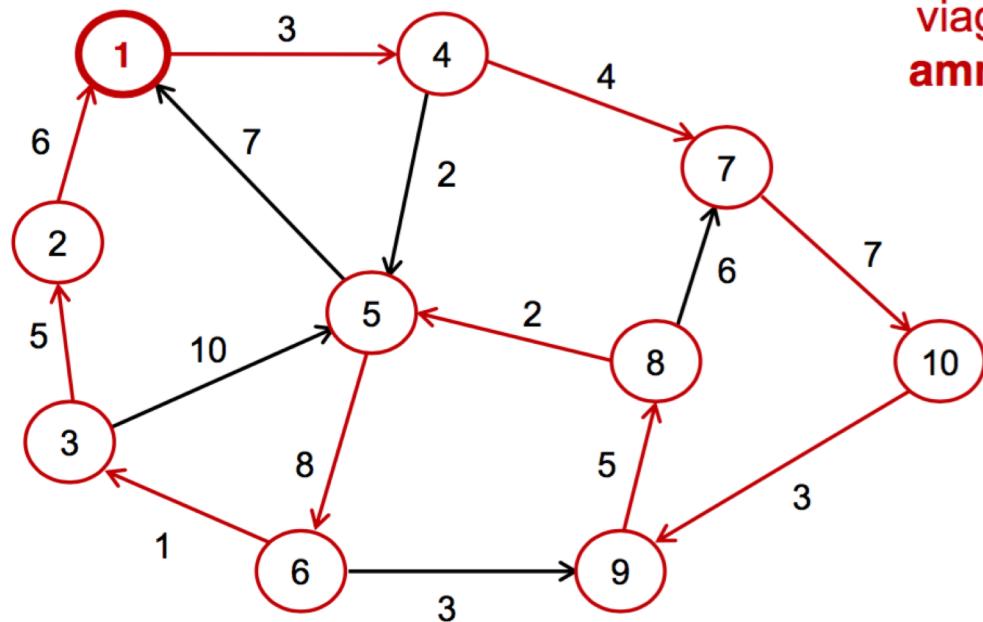
TOUR

Fissato un nodo di G, ad esempio il nodo 1, un **tour** (o **ciclo hamiltoniano**) è un percorso (sequenza ordinata di nodi-archi) che inizia e termina nel nodo fissato (S.P.I.G. Il nodo 1) e che visita ogni nodo di G una e una sola volta.



Il problema del commesso viaggiatore (TSP)

Problema: Un commesso viaggiatore (Travel Salesman), partendo dalla città in cui vive, ogni giorno deve visitare $n=10$ città. Egli deve passare in ciascuna città esattamente una volta e alla fine del percorso deve ritornare nella città di partenza minimizzando il *costo totale degli spostamenti*.



Nel problema del commesso viaggiatore le **soluzioni ammissibili** sono i tour.

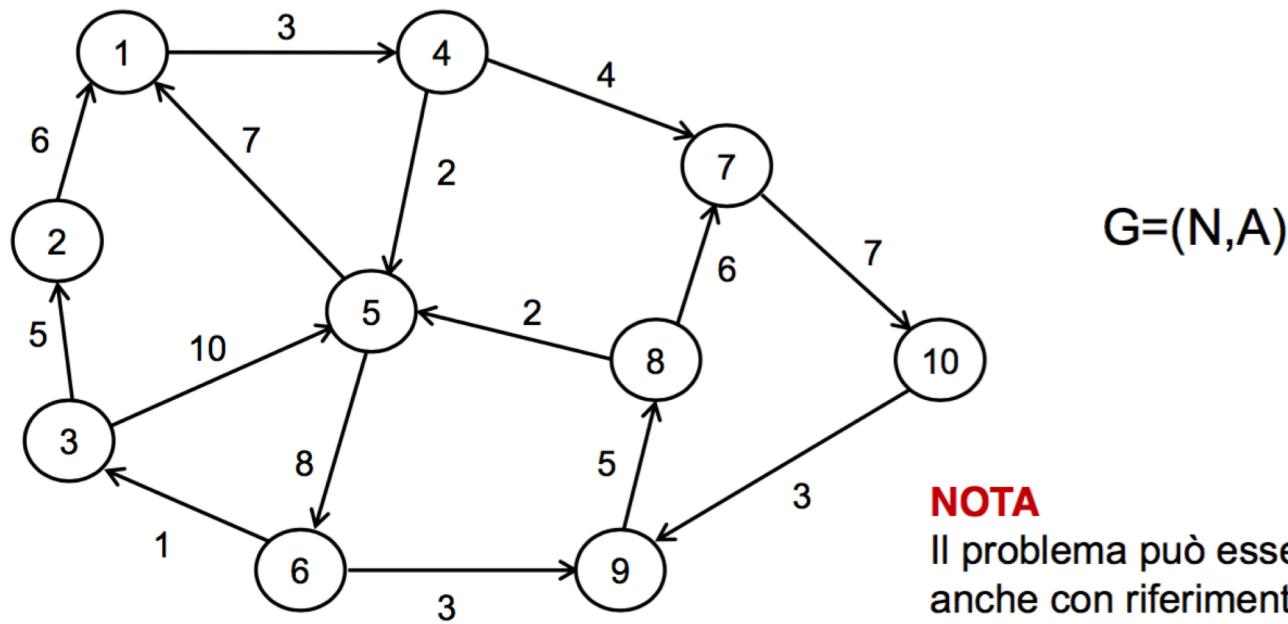
HP: $G=(N,A)$
rappresenta le città
(nodi in N) e i
collegamenti tra città
(archi in A).

Il problema del commesso viaggiatore (TSP)

Problema del TSP

Dato un grafo $G=(N,A)$, con $|N|=n$ e $|A|=m$, e con **costi non negativi associati agli archi**, individuare il tour di costo totale minimo.

Il **costo del tour** è dato dalla somma dei costi degli archi che sono stati inseriti nel ciclo.

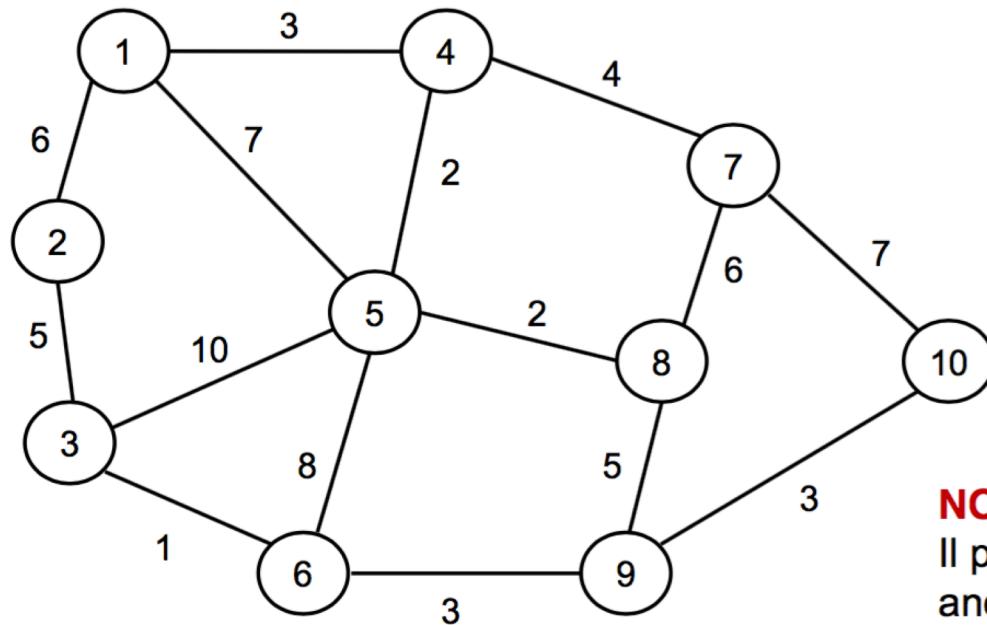


Il problema del commesso viaggiatore (TSP)

Problema del TSP

Dato un grafo $G=(N,A)$, con $|N|=n$ e $|A|=m$, e con **costi non negativi associati agli archi**, individuare il tour di costo totale minimo.

Il **costo del tour** è dato dalla somma dei costi degli archi che sono stati inseriti nel ciclo.



$G=(N,A)$

NOTA

Il problema può essere formulato anche con riferimento a un grafo non orientato (**TSP simmetrico**).

Il problema del commesso viaggiatore (TSP)

Il TSP descrive il problema combinatorio della **ricerca su G di un tour di costo minimo** ed incorpora per questo l'essenza di tutti quei problemi di ottimizzazione discreta noti con il nome di **problem di routing**.

Alcuni esempi sono:

- **Veichle Routing Problem** (consegne merci, pick-up passeggeri)
Un insieme di veicoli, ciascuno con la sua capacità di carico, deve visitare un insieme di clienti. Il problema è assegnare clienti a veicoli e poi determinare **per ciascun veicolo il tour dei suoi clienti di costo minimo**.
- **Scheduling di lavori su una macchina** (ad es., operazioni di set up in un macchinario tra la lavorazione di un pezzo e del successivo)
Un macchinario che vernicia sportelli di auto, **tra la lavorazione di un pezzo e quella del successivo**, deve eseguire alcune operazioni di ripristino o set up (come, ad es., **pulire le bocchette per la verniciature, ricaricare i colori, riposizionare gli spruzzatori**, ecc) e ciascuna di queste operazioni ha un costo (es.: tempo).
Stabilite tramite un grafo $G=(N,A)$ le **relazioni di precedenza** tra le operazioni di set up da eseguire, il problema consiste nell'**individuare la sequenza di operazioni (città)** che il macchinario (commesso viaggiatore) deve eseguire per effettuare il set up completo al minimo costo totale.

Il problema del commesso viaggiatore (TSP)

Il TSP descrive il problema combinatorio della **ricerca su G di un tour di costo minimo** ed incorpora per questo l'essenza di tutti quei problemi di ottimizzazione discreta noti con il nome di **problem di routing**.

Per il problema del TSP esistono **molte formulazioni** (di Programmazione Lineare a Intera o Mista) diverse, anche a seconda che si tratti della versione su grafo orientato (**TSP asimmetrico**) o non orientato (**TSP simmetrico**).

Purtroppo il TSP è un problema **computazionalmente difficile** e pertanto le formulazioni possono essere utilizzate per risolverlo attraverso tecniche di tipo B&B, ma senza garanzie sui tempi di calcolo.

In alternativa al B&B (o a tecniche simili) sono stati studiati algoritmi di soluzione ad hoc per il TSP che ovviamente **non sono di natura esatta**:

Algoritmo di **approssimazione**
di Christofides (1975).

Algoritmo **euristico di Ricerca Locale**
di Lin e Kernighan (1973).