Algoritmi Euristici

Corso di Laurea in Informatica e Corso di Laurea in Matematica

Roberto Cordone

DI - Università degli Studi di Milano

Lezioni: Lunedì 13.30 - 15.30 in Aula G30
Giovedì 13.30 - 15.30 in Aula G30
Ricevimento: su appuntamento
Tel.: 02 503 16235
E-mail: roberto.cordone@unimi.it
Web page: http://homes.di.unimi.it/~cordone/courses/2018-ae/2018-ae.html

Gli algoritmi di scambio

In Ottimizzazione Combinatoria ogni soluzione x è un sottoinsieme di B

Un'euristica di scambio aggiorna passo per passo un sottoinsieme $x^{(t)}$

1. parte da una soluzione ammissibile $x^{(0)} \in X$ trovata in qualche modo (*spesso con un'euristica costruttiva*)
2. scambia coppie (A, D) di sottoinsiemi, A esterno e D interno a $x^{(t)}$ (*add/drop*), generando una famiglia di soluzioni ammissibili

$$x'_{A,D} = x \cup A \setminus D \text{ con } A \subseteq B \setminus x \text{ e } D \subseteq x$$

3. ad ogni passo t, sceglie quali sottoinsiemi scambiare in base a un opportuno criterio di scelta $\varphi(x, A, D)$

$$(A^*, D^*) = \arg \min_{(A,D)} \varphi(x, A, D)$$

4. genera la nuova soluzione corrente eseguendo le modifiche scelte

$$x^{(t+1)} := x^{(t)} \cup A^* \setminus D^*$$

5. se vale una condizione di fine, termina; altrimenti, torna al punto 2

Intorno

Un'euristica di scambio è definita da:

- le coppie di sottoinsiemi (A, D) disponibili per uno scambio, e quindi le soluzioni generabili con uno scambio a partire da x
- il criterio di scelta $\varphi(x, A, D)$

Intorno $N : X \to 2^X$ è una funzione che associa a ogni soluzione ammissibile $x \in X$ un sottoinsieme di soluzioni ammissibili $N(x) \subseteq X$

Possiamo formalizzare la situazione con un grafo di ricerca in cui

- i nodi rappresentano le soluzioni ammissibili $x \in X$
- gli archi collegano ogni soluzione x a quelle del suo intorno $N(x)$

Dato un grafo di ricerca

- un'esecuzione dell'algoritmo corrisponde a un cammino
- un arco viene anche detto mossa, perché trasforma una soluzione in un'altra muovendo elementi

Come si definisce un intorno e come si sceglie una mossa?

Intorni basati sulla distanza

Ogni soluzione $x \in X$ si può rappresentare col proprio vettore di incidenza

$$x_i = \begin{cases} 1 \text{ se } i \in x \\ 0 \text{ se } i \in B \setminus x \end{cases}$$

Distanza di Hamming tra due soluzioni x e x'
è il numero di elementi per cui i loro vettori di incidenza differiscono

$$d_H(x, x') = \sum_{i \in B} |x_i - x'_i|$$

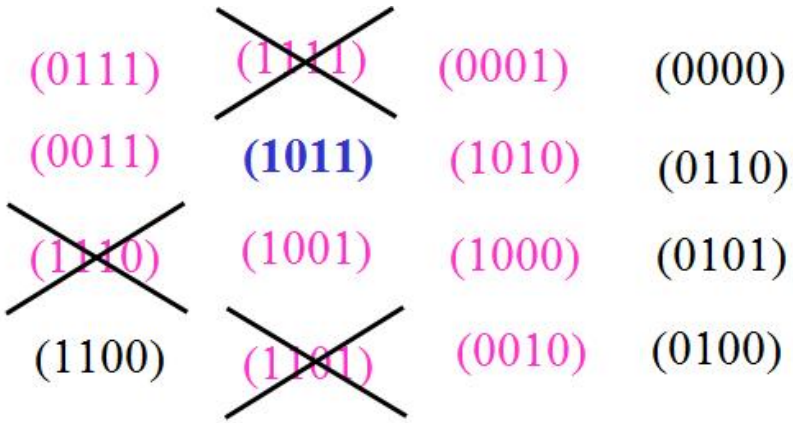
Riferendosi ai sottoinsiemi, è come dire $d_H(x, x') = |x \setminus x'| + |x' \setminus x|$

Una tipica definizione di intorno di x, con un parametro intero k, è l'insieme delle soluzioni con distanza di Hamming non superiore a k da x

$$N_{H_k}(x) = \{x' \in X : d_H(x, x') \leq k\}$$

Esempio: il *KP*

L'istanza del *KP* con $B = \{1, 2, 3, 4\}$, $w = [\ 5\ 4\ 3\ 2\ ]$ e $W = 10$, ha le seguenti soluzioni



dove i sottoinsiemi $\{1, 2, 3, 4\}$, $\{1, 2, 3\}$ e $\{1, 2, 4\}$ non sono ammissibili.

La soluzione $x = \{1, 3, 4\}$ (in blu) ha un intorno $N_{H_2}(x)$ di 7 elementi (in rosa)

I sottoinsiemi in nero non fanno parte dell'intorno, perché la loro distanza di Hamming da x è > 2

Intorni basati su operazioni

Un'altra comune definizione di intorno si ottiene definendo

- una famiglia $\mathcal{O}$ di operazioni sulle soluzioni del problema
- l'insieme delle soluzioni generate eseguendo su x le operazioni di $\mathcal{O}$

$$N_{\mathcal{O}}(x) = \{x' \in X : \exists o \in \mathcal{O} : o(x) = x'\}$$

Considerando ancora l'esempio del KP, si può definire $\mathcal{O}$ come

- aggiunta a x di un elemento di $B \setminus x$
- eliminazione di un elemento di x
- scambio di un elemento di x con uno di $B \setminus x$

L'intorno $N_{\mathcal{O}}$ che ne deriva è legato a quelli definiti dalla distanza di Hamming, ma non coincide con nessuno di loro

$$N_{H_1} \subset N_{\mathcal{O}} \subset N_{H_2}$$

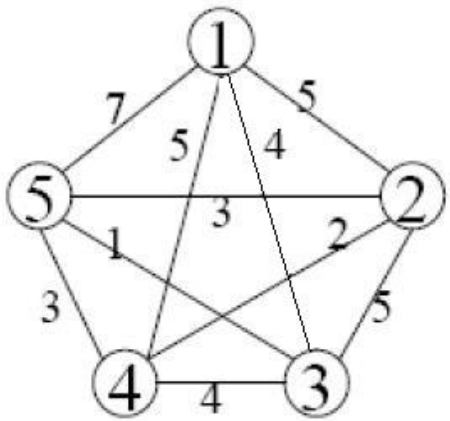
Questi intorni sono parametrizzabili eseguendo sequenze di k operazioni di $\mathcal{O}$ anziché una sola, esattamente come gli intorni basati su distanza

$$N_{\mathcal{O}_k}(x) = \{x' \in X : \exists o_1, \ldots, o_k \in \mathcal{O} : o_k(o_{k-1}(\ldots o_1(x))) = x'\}$$

Differenza tra intorni basati su distanza e su operazioni

In generale, un intorno basato su operazioni include soluzioni a distanza di Hamming diversificata

Per il *TSP* possiamo definire un intorno $N_{\mathcal{S}_1}$ che contiene le soluzioni ottenibili scambiando due nodi nel loro ordine di visita



La soluzione $x = (3, 1, 4, 5, 2)$ ha intorno:

$$N_{\mathcal{S}_1}(x) = \{(1,3,4,5,2), (4,1,3,5,2), (5,1,4,3,2), (2,1,4,5,3), (3,4,1,5,2), (3,5,4,1,2), (3,2,4,5,1), (3,1,5,4,2), (3,1,2,5,4), (3,1,4,2,5)\}$$

Rispetto a x cambiano $3+3$ archi se i due nodi sono adiacenti, $4+4$ altrimenti

Relazioni fra intorni basati su distanza e su operazioni

Talvolta gli intorni ottenuti con le due definizioni coincidono

- per l'*MDP* coincidono
  - l'intorno N_{H_2} delle soluzioni a distanza di Hamming pari a 2
  - l'intorno $N_{\mathcal{S}_1}$ degli scambi di un elemento di x con uno di $B \setminus x$
- per il *BPP* coincidono
  - l'intorno N_{H_2} delle soluzioni a distanza di Hamming pari a 2
  - l'intorno $N_{\mathcal{T}_1}$ dei trasferimenti di un oggetto a un altro contenitore
- per *SAT* coincidono
  - l'intorno N_{H_2} delle soluzioni a distanza di Hamming pari a 2
  - l'intorno $N_{\mathcal{F}_1}$ dei "flip", che sostituiscono l'assegnamento di verità di una variabile con l'assegnamento opposto

Questo è tipico dei problemi con soluzioni a cardinalità fissata:

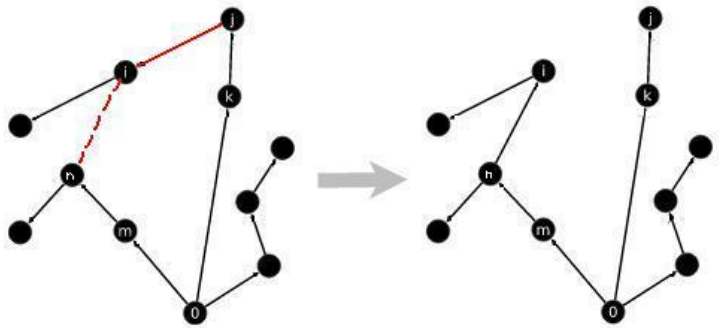
- si esegue una sequenza di k scambi fra elementi singoli ($|A| = |D| = 1$): k elementi entrano in x e k elementi ne escono
- la distanza di Hamming fra le due soluzioni estreme è $\leq 2k$
  (*se si scelgono sempre elementi diversi, è esattamente* $2k$)

Intorni diversi per lo stesso problema: il *CMST*

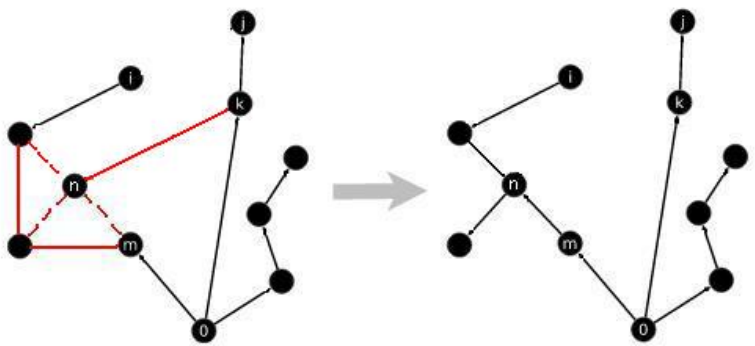
Un problema può ammettere intorni diversi basati su operazioni diverse

Nel *CMST* si possono

- scambiare lati: esce (i, j), entra (i, n)



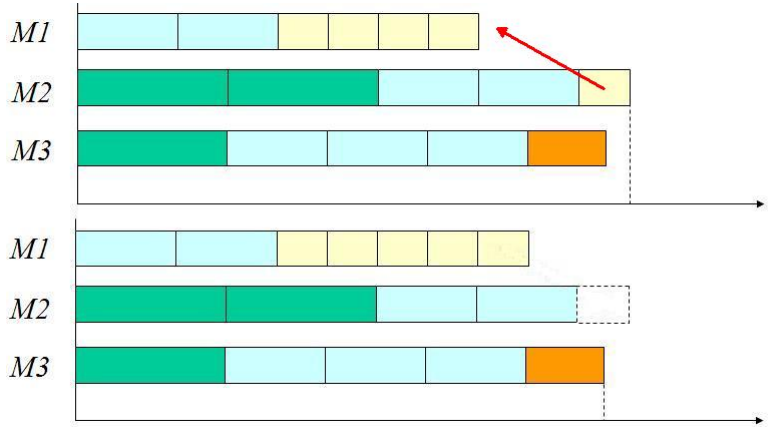
- scambiare vertici: n passa dal sottoalbero 2 al sottoalbero 1 (ricalcolando i lati per connettere ogni sottoalbero a costo minimo):



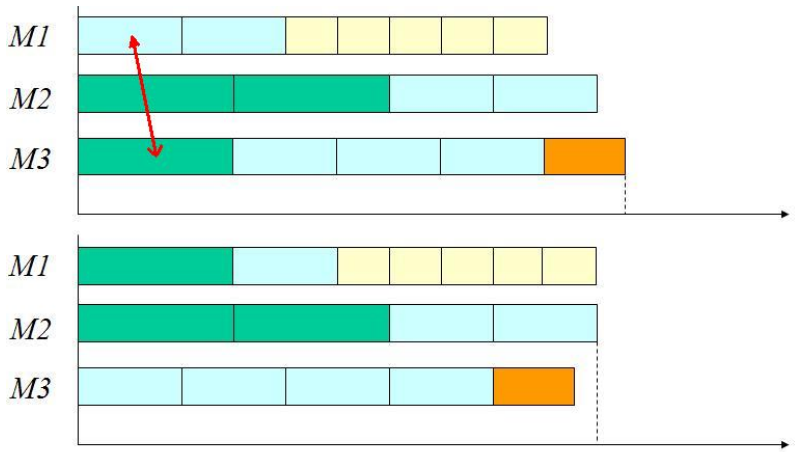
Intorni diversi per lo stesso problema: il *PMSP*

Per il *PMSP* si può definire un

- intorno di trasferimento $N_{\mathcal{T}_1}$, basato sull'insieme $\mathcal{T}_1$ dei trasferimenti di una lavorazione su un'altra macchina



- intorno di scambio $N_{\mathcal{S}_1}$, basato sull'insieme $\mathcal{S}_1$ degli scambi di due lavorazioni fra due macchine (una per macchina)



Connettività dello spazio delle soluzioni

Un'euristica di scambio può trovare soluzione ottima solo se
almeno una soluzione ottima è raggiungibile da ogni soluzione iniziale

Si dice che il grafo di ricerca è connesso all'ottimo quando
contiene per ogni soluzione $x \in X$ un cammino da x a X^*

Poiché non conosciamo X^*, spesso si usa una condizione più forte:
si dice che il grafo di ricerca è fortemente connesso quando
contiene per ogni coppia di soluzioni $x, y \in X$ un cammino da x a y

Un'euristica di scambio dovrebbe garantire una di tali condizioni

Questo non è sempre possibile

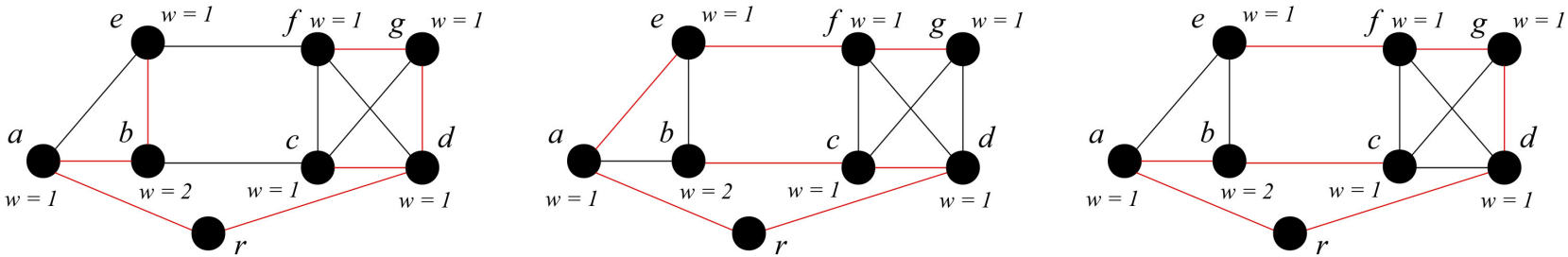
- nel *MDP*, l'intorno N_{S_1} permette di collegare qualsiasi coppia di soluzioni in al più k passi
- nel *KP* e nel *SCP*, nessun intorno N_{S_k} lo garantisce

  Le soluzioni ammissibili possono avere cardinalità qualsiasi
- il grafo di ricerca diventa connesso anche nel *KP* e nel *SCP*
  se oltre agli scambi si ammettono anche eliminazioni e aggiunte

Connettività dello spazio delle soluzioni

Se l'ammissibilità è definita in modo sofisticato, gli intorni $N_{\mathcal{S}_k}$ sono connessi solo per k abbastanza grande: i sottoinsiemi non ammissibili possono interrompere i percorsi fra soluzioni ammissibili

Consideriamo un CMST su grafo non completo e scambi di lati



Per $W = 4$, solo tre soluzioni sono ammissibili, tutte con due sottoalberi:

- $x = \{(r,a),(a,b),(b,e),(r,d),(c,d),(d,g),(f,g)\}$
- $x = \{(r,a),(a,e),(e,f),(f,g),(r,d),(c,d),(b,c)\}$
- $x = \{(r,a),(a,b),(b,c),(r,d),(d,g),(f,g),(e,f)\}$

Le tre soluzioni sono mutuamente raggiungibili solo scambiando due lati per volta; scambiandone uno solo, si ottengono sottoinsiemi inammissibili

Euristiche *steepest descent* (*hill-climbing*)

Molto spesso il criterio di scelta $\varphi(x, A, D)$ della nuova soluzione in $N(x)$ è la funzione obiettivo, cioè ad ogni passo dell'euristica
ci si sposta dalla soluzione corrente alla miglior soluzione del suo intorno

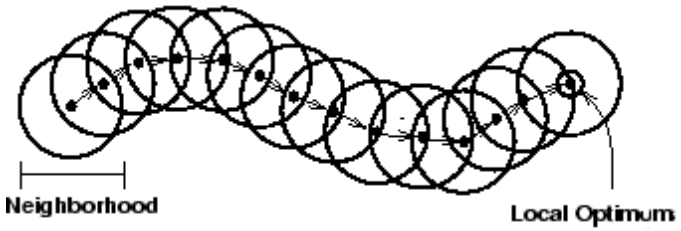
Per evitare comportamenti ciclici, si accettano solo soluzioni miglioranti

Algorithm SteepestDescent$\left(I, x^{(0)}\right)$
$x := x^{(0)}$;
Fine := *false*;
While Fine = *false do*
  $\tilde{x} := \arg \min_{x' \in N(x)} f(x')$;
  If $f(\tilde{x}) \geq f(x)$ *then* Fine := *true*; *else* $x := \tilde{x}$;
EndWhile;
Return $(x, f(x))$;

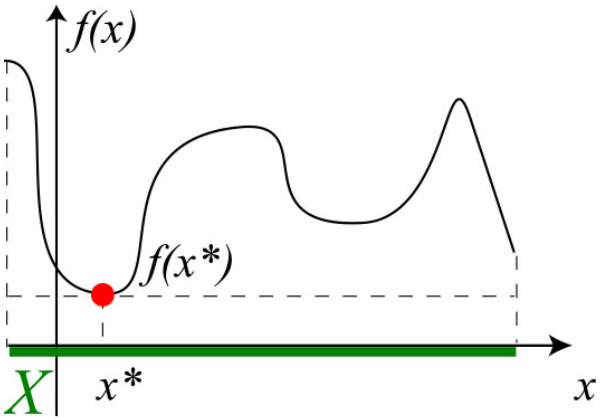
Ottimalità locale e globale

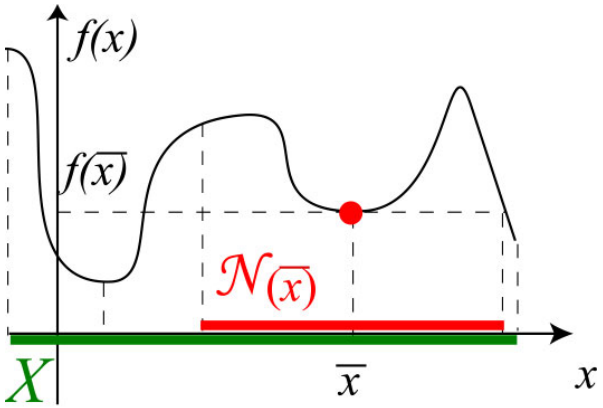
Un'euristica *steepest descent* termina quando trova una soluzione di ottimo locale, cioè una soluzione $\bar{x} \in X$ tale che

$$f(\bar{x}) \leq f(x) \text{ per ogni } x \in N(x)$$



Una soluzione di ottimo globale è sempre anche di ottimo locale, ma non sempre avviene il contrario: $X^* \subseteq \bar{X}_N \subseteq X$





Intorno esatto

Intorno esatto è una funzione intorno $N : X \to 2^X$ tale che ogni ottimo locale è anche ottimo globale

$$\bar{X}_N = X^*$$

Un caso banale si ha quando l'intorno di ogni soluzione coincide con l'intero insieme delle soluzioni ammissibili ($N(x) = X$ per ogni $x \in X$)

È un intorno inutile: troppo lungo da esplorare

Gli intorni esatti sono estremamente rari

- un caso rilevante è lo scambio fra variabili di base e fuori base usato dall'algoritmo del simplesso per la Programmazione Lineare
- un altro caso è lo scambio fra lati per il problema dell'albero ricoprente minimo

In generale, l'euristica *steepest descent* non trova un ottimo globale

La sua efficacia dipende dalle proprietà del grafo di ricerca e dell'obiettivo

Proprietà del grafo di ricerca

Alcune proprietà rilevanti sono

- la dimensione dello spazio di ricerca $|X|$
- la connettività del grafo di ricerca (come discusso sopra)
- il diametro del grafo di ricerca, cioè il numero di archi del più lungo percorso minimo fra due soluzioni: un intorno più ricco produce grafi di diametro inferiore (*ma non è tutto lì: effetto "smallworld"*)

Ad esempio, per il *TSP* simmetrico su grafo completo con l'intorno N_{S_1}:

- lo spazio di ricerca contiene $|X| = (n-1)!$ soluzioni
- l'intorno N_{S_1} (scambio fra due nodi) contiene $\binom{n}{2} = \frac{n(n-1)}{2}$ soluzioni
- il grafo di ricerca è fortemente connesso e ha diametro $\leq n-2$: ogni soluzione si trasforma in ogni altra con al più $n-2$ scambi

  Ad esempio, $x = (1,5,4,2,3)$ diventa $x' = (1,2,3,4,5)$ in 3 passi

  $$x = (1,5,4,2,3) \to (1,2,4,5,3) \to (1,2,3,5,4) \to (1,2,3,4,5) = x'$$

  (*il primo nodo è sempre* 1, *e l'ultimo va a posto da solo*)

Proprietà del grafo di ricerca

Altre proprietà rilevanti

- la densità delle soluzioni di ottimo globale ($\frac{|X^*|}{|X|}$) e locale ($\frac{|\bar{X}_N|}{|X|}$): se gli ottimi locali sono molti, è difficile trovare quello globale
- la distribuzione della qualità $\delta(\bar{x})$ delle soluzioni di ottimo locale (diagramma *SQD*): se gli ottimi locali sono buoni, importa meno trovare quello globale
- la distribuzione delle soluzioni di ottimo locale nello spazio di ricerca: se gli ottimi locali sono vicini, non occorre esplorare tutto lo spazio

Questi indici richiederebbero l'esplorazione esaustiva del grafo di ricerca

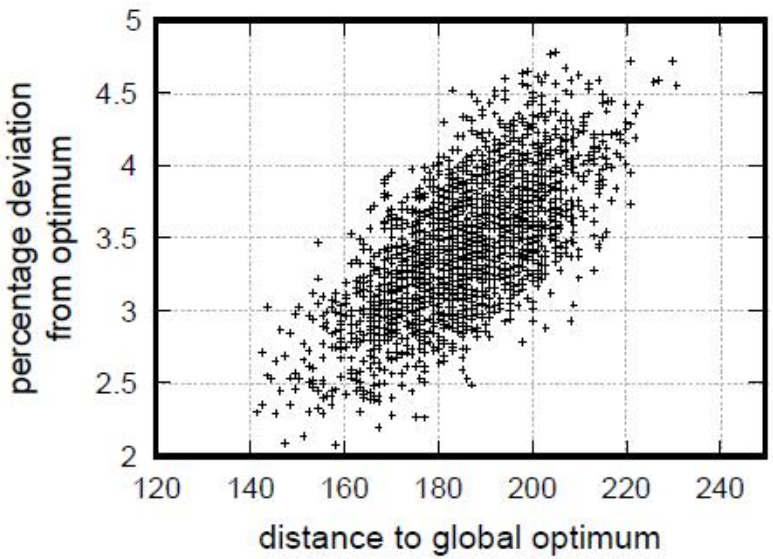
In pratica, ci si limita ad effettuare campionamenti: queste analisi

- richiedono tempi molto lunghi
- possono avere esiti fuorvianti, specie se non son noti gli ottimi globali

Esempio: il *TSP*

Tipici risultati per il *TSP* su grafo simmetrico completo con costi euclidei

- la distanza di Hamming fra due ottimi locali è in media $\ll n$:
  gli ottimi locali si concentrano in una piccola regione di X
- la distanza di Hamming fra ottimi locali in media supera quella fra ottimi locali e globali:
  gli ottimi globali tendono a stare in mezzo a quelli locali
- il diagramma *FDC* (*Fitness-Distance Correlation*)
  lega la qualità $\delta(\bar{x})$ con la distanza dagli ottimi globali $d_H(\bar{x}, X^*)$:
  se c'è correlazione, gli ottimi locali migliori sono più vicini ai globali



Fitness-Distance Correlation

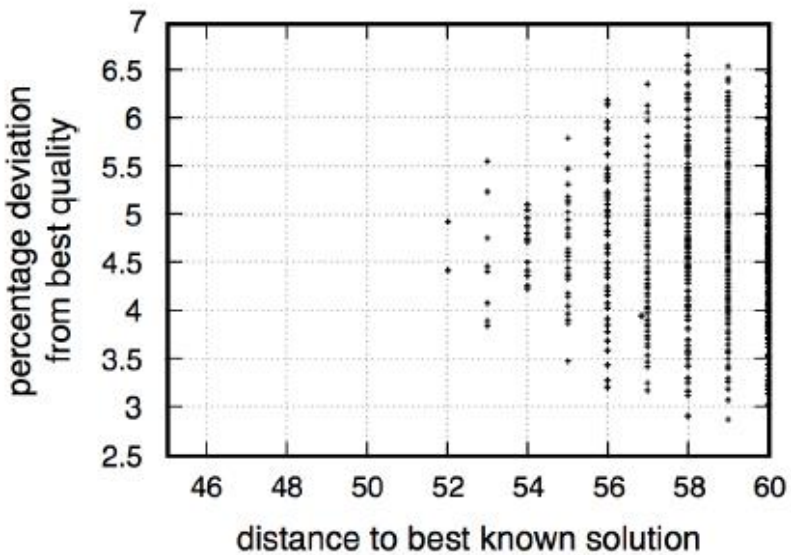
Se la correlazione fra qualità e vicinanza agli ottimi globali è forte

- conviene costruire buone soluzioni iniziali,
  perché avvicinano la ricerca locale a buoni ottimi locali
- conviene intensificare piuttosto che diversificare

Al contrario, se la correlazione è debole

- una buona inizializzazione è meno importante
- conviene diversificare piuttosto che intensificare

Questo avviene ad esempio per il *Quadratic Assignment Problem* (*QAP*)



Landscape

Si definisce *landscape* la terna (X, N, f), dove

- X è lo spazio di ricerca, o insieme delle soluzioni ammissibili
- $N : X \to 2^X$ è la funzione intorno
- $f : X \to \mathbb{N}$ è la funzione obiettivo

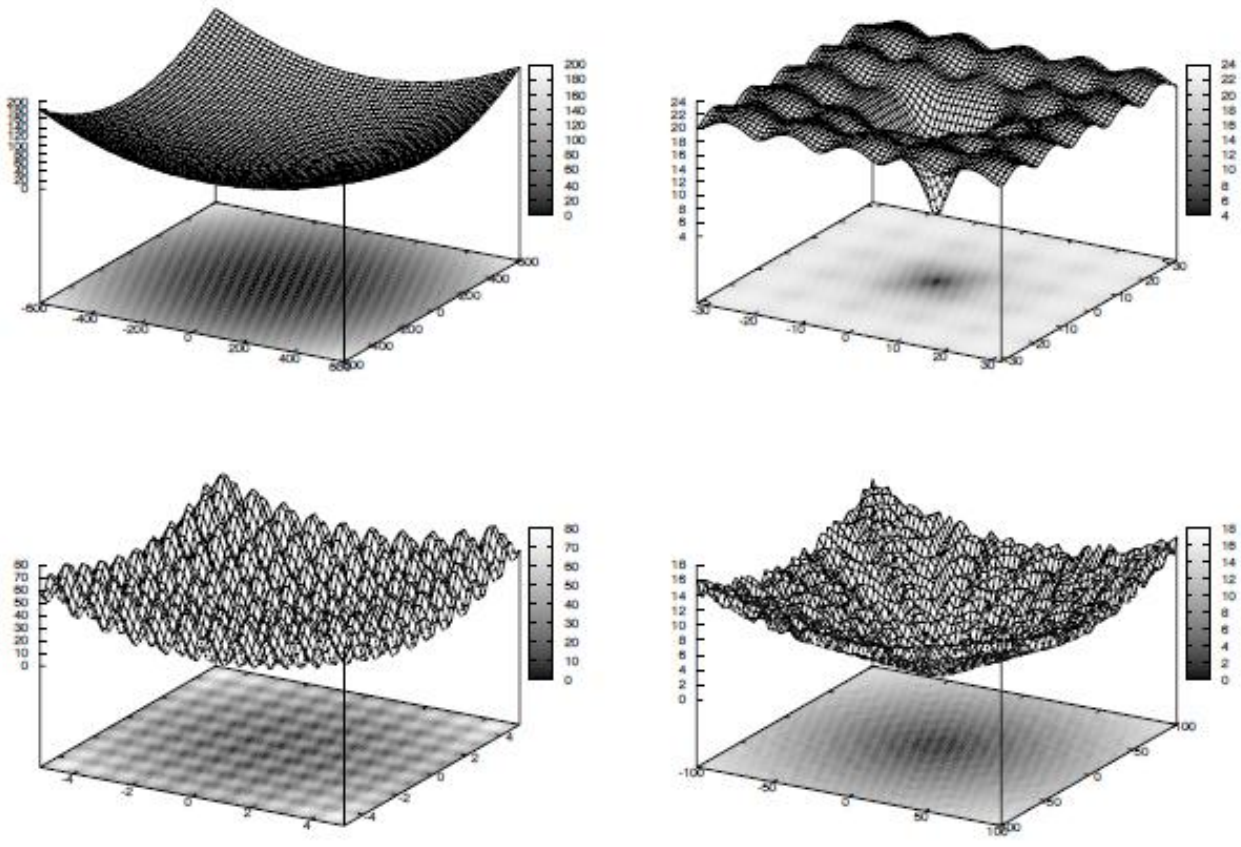
Si può vedere come il grafo di ricerca pesato sui nodi con l'obiettivo

L'efficacia delle euristiche di scambio dipende molto dal *landscape*,

Landscape accidentati sono associati a molti ottimi locali,
dunque a euristiche meno efficaci

Tipi diversi di *landscape*

Esiste una grande varietà di *landscape* molto differenti tra loro



Coefficiente di autocorrelazione

La complessità di un *landscape* si può stimare empiricamente

1. eseguendo una *random walk* nel grafo di ricerca
2. determinando la sequenza di valori dell'obiettivo $f^{(1)}, \ldots, f^{(t_{\max})}$
3. calcolandone il valor medio campionario $\bar{f} = \sum_{t=1}^{t_{\max}} f^{(t)}$
4. calcolando il coefficiente empirico di autocorrelazione

$$r(i) = \frac{\frac{\sum_{t=1}^{t_{\max}-i} \left(f^{(t)} - \bar{f}\right)\left(f^{(t+i)} - \bar{f}\right)}{t_{\max} - i}}{\frac{\sum_{t=1}^{t_{\max}} \left(f^{(t)} - \bar{f}\right)^2}{t_{\max}}}$$

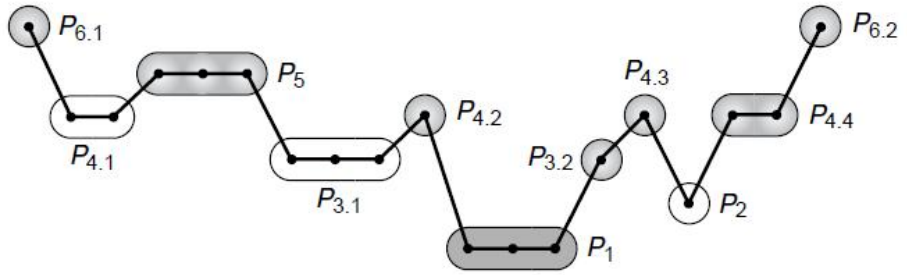
Si tratta di una funzione di i che parte da $r(0) = 1$; in genere va calando

- se rimane $r(i) \approx 1$, il *landscape* è liscio:
  - le soluzioni dell'intorno hanno valori vicini a quella corrente
  - ci sono pochi ottimi locali
  - l'euristica *steepest descent* è efficace
- se varia bruscamente, il *landscape* è accidentato:
  - le soluzioni dell'intorno hanno valori lontani da quella corrente
  - ci sono molti ottimi locali
  - l'euristica *steepest descent* è poco efficace

Plateau

Si può analizzare il grafo di ricerca dividendolo per livelli di obiettivo

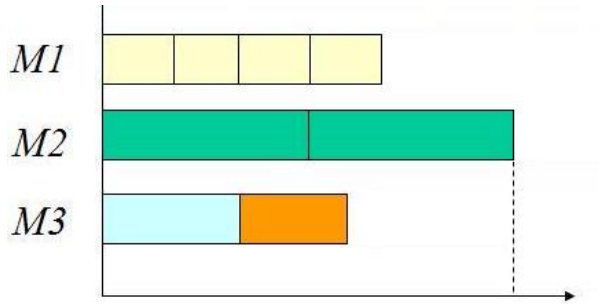
- plateau di valore f è ciascun sottoinsieme di soluzioni di valore f che siano adiacenti nel grafo di ricerca



Plateau molto ampii ostacolano la scelta della soluzione nell'euristica *steepest descent*, perché la fanno dipendere dall'ordine di visita di $N(x)$

Quindi un landscape troppo uniforme non è un vantaggio!

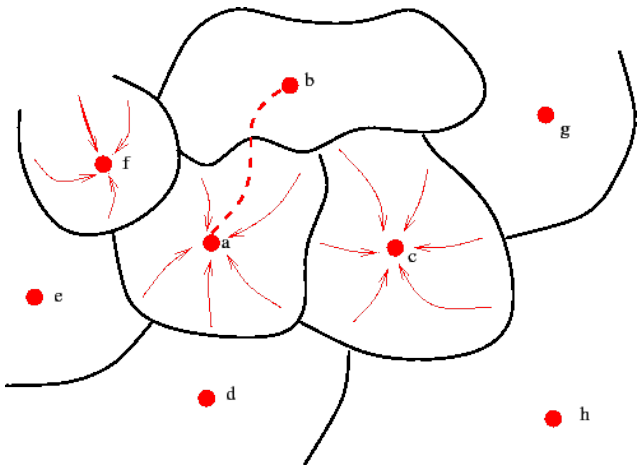
Esempio: ogni trasferimento o scambio fra le macchine $M1$ e $M3$ lasciano invariato il valore dell'obiettivo (*i trasferimenti su M2 lo peggiorano*)

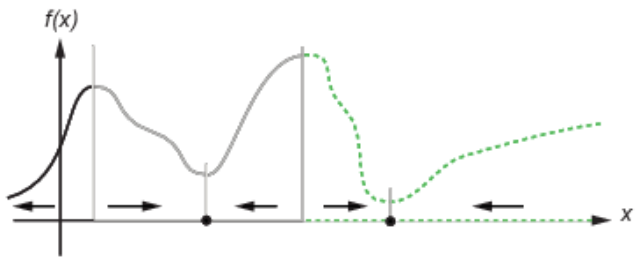


Bacini di attrazione

Una suddivisione alternativa del grafo di ricerca si basa sul concetto di:

- bacino di attrazione di una soluzione di ottimo locale $\bar{x}$, cioè l'insieme delle soluzioni $x^{(0)} \in X$ tali che l'euristica *steepest descent* partendo da $x^{(0)}$ produca come risultato $\bar{x}$





L'euristica *steepest descent* è

- efficace se i bacini di attrazione sono pochi e ampi
  (specialmente se gli ottimi globali hanno bacini più ampi)
- inefficace se i bacini di attrazione sono molti e piccoli
  (specialmente se gli ottimi globali hanno bacini più piccoli)

Complessità

Algorithm SteepestDescent$\left(I, x^{(0)}\right)$

$x := x^{(0)}$;

Fine := *false*;

While Fine = *false do*

$\tilde{x} := \arg \min_{x' \in N(x)} f(x)$;

If $f(\tilde{x}) \geq f(x)$ *then* Fine := *true*; *else* $x := \tilde{x}$;

EndWhile;

Return $(x, f(x))$;

La complessità dell'euristica *steepest descent* dipende da

1. numero di passi: dipende dalla struttura del grafo di ricerca (ampiezza dei bacini di attrazione), difficile da prevedere a priori
2. ricerca della miglior soluzione nell'intorno: dipende da come viene condotta la ricerca stessa

L'esplorazione dell'intorno

Due strategie sono possibili

1. ricerca esaustiva: si valutano tutte le soluzioni dell'intorno;
   la complessità del singolo passo è il prodotto di
   - numero di soluzioni dell'intorno ($|N(x)|$)
   - valutazione del costo di ogni soluzione ($\gamma_N(|B|, x)$)

   Talvolta non è facile limitarsi a visitare solo le soluzioni dell'intorno:
   - si visita un sovrainsieme dell'intorno $\tilde{N}(x) \supseteq N(x)$
   - si valuta l'ammissibilità per ogni elemento $x' \in \tilde{N}(x)$
   - per quelli ammissibili (se $x' \in N(x)$) si valuta il costo $f(x')$
2. esplorazione efficiente dell'intorno: anziché visitare tutto l'intorno,
   si trova la sua soluzione ottima risolvendo un problema ausiliario

   Solo intorni particolari lo consentono

Visita esaustiva dell'intorno

Algorithm SteepestDescent$(I, x^{(0)})$

$x := x^{(0)}$;

Fine := *false*;

While Fine = *false do*

  $\tilde{x} := x$; $\{\ \tilde{x} := \arg \min_{x' \in N(x)} f(x')\ \}$

  For each $x' \in N(x)$ *do*

    If $f(x') < f(\tilde{x})$ *then* $\tilde{x} := x'$;

  EndFor;

  If $f(\tilde{x}) \geq f(x)$ *then* Fine := *true*; *else* $x := \tilde{x}$;

EndWhile;

Return $(x, f(x))$;

La complessità è il prodotto di tre termini

1. il numero di iterazioni $t_{\max}$ eseguite per raggiungere l'ottimo locale
2. il numero di soluzioni $\left|N\left(x^{(t)}\right)\right|$ visitate ad ogni iterazione
3. il tempo di valutazione dell'obiettivo γ_N

$|N|$ e γ_N dipendono da $|B|$, e spesso da t (*allora si stima per eccesso*)

Tempo di valutazione dell'obiettivo: il caso additivo

Il primo accorgimento per accelerare un algoritmo di scambio è minimizzare il tempo di valutazione dell'obiettivo: in generale, aggiornare il valore di $f(x)$ costa molto meno che ricalcolarlo

Per esempio, se $f(x)$ è additiva e il numero di elementi aggiunti ($|A|$) ed eliminati ($|D|$) sono limitati, la valutazione prende tempo costante:

- si somma ϕ_j per ogni elemento $j \in A$, aggiunto a x
- si sottrae ϕ_i per ogni elemento $i \in D$, tolto da x

Passando dalla soluzione x a $x' = x \setminus D \cup A$, l'obiettivo cambia di

$$\delta f(x, A, D) = f(x \setminus D \cup A) - f(x) = \sum_{j \in A} \phi_j - \sum_{i \in D} \phi_i$$

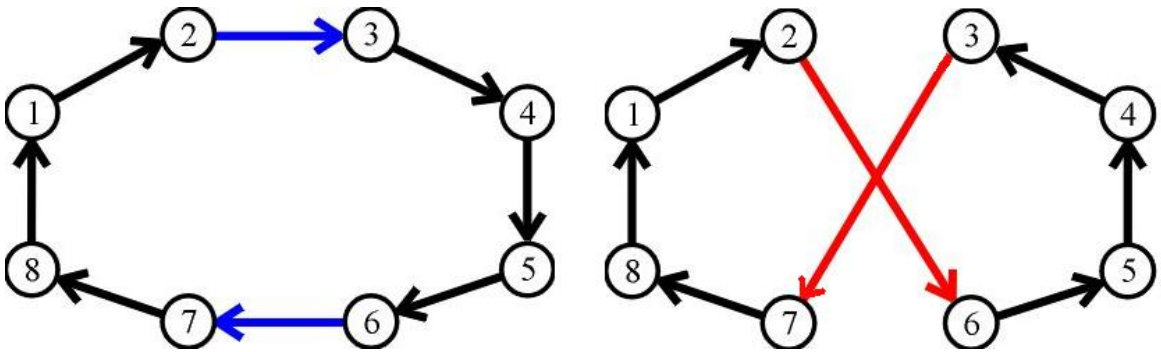
Si noti che $\delta f(x, A, D)$ non dipende da x: ne riparleremo

Esempi: lo scambio semplice di oggetti nel *KP* e lati nel *CMSTP*

Esempio: il *TSP* simmetrico

L'intorno $N_{\mathcal{R}_2}$ per il *TSP*

- elimina due archi (s_i, s_{i+1}) e (s_j, s_{j+1}) non consecutivi
- aggiunge i due archi (s_i, s_j) e (s_{i+1}, s_{j+1})
- inverte il percorso $(s_{i+1}, \ldots, s_j)$ (modificando $O(n)$ archi!)



Se il grafo e la funzione costo sono simmetrici, la variazione di $f(x)$ è

$$\delta f(x, i, j) = c_{s_i, s_j} + c_{s_{i+1}, s_{j+1}} - c_{s_i, s_{i+1}} - c_{s_j, s_{j+1}}$$

In molti altri casi, però, la funzione non è additiva

Funzioni quadratiche

Nell'*MDP* la funzione obiettivo è quadratica: il ricalcolo costa $O\left(n^2\right)$

Se si passa da x a $x' = x \setminus \{i\} \cup \{j\}$ (intorno N_{S_1}), l'obiettivo cambia di

$$\delta f(x,i,j) = f(x \setminus \{i\} \cup \{j\}) - f(x) = \sum_{h,k \in x \setminus \{i\} \cup \{j\}} d_{hk} - \sum_{h,k \in x} d_{hk}$$

La differenza riguarda le distanze relative a due punti, che sono $O(n)$

Esiste un trucco generale per le funzioni quadratiche simmetriche

$$\begin{aligned} \delta f(x,i,j) &= \sum_{h \in x} \sum_{k \in x} d_{hk} - \sum_{h \in x \setminus \{i\} \cup \{j\}} \sum_{k \in x \setminus \{i\} \cup \{j\}} d_{hk} \Rightarrow \\ \Rightarrow \delta f(x,i,j) &= 2 \sum_{k \in x} d_{jk} - 2 \sum_{k \in x} d_{ik} - 2 d_{ij} = 2\left(D_j(x) - D_i(x) - d_{ij}\right) \end{aligned}$$

Se si conosce $D_\ell(x) = \sum_{k \in x} d_{\ell k}$ per ogni $\ell \in B$, il calcolo richiede $O(1)$

Esempio: il *MDP*

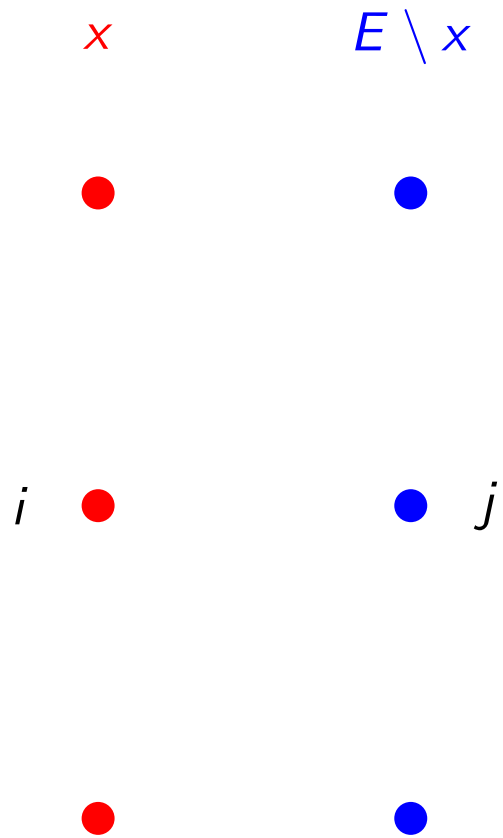
Per brevità, dimezziamo l'obiettivo $f(x)$
Valutiamo lo scambio

$$x \to x' = x \setminus \{i\} \cup \{j\}$$

con $i \in x$ e $j \in B \setminus x$

$$f(x') = f(x) - D_i + D_j - d_{ij}$$

- Si perdono le coppie contenenti i
- Si acquistano le coppie contenenti j
- Ma la coppia (i, j) è di troppo



Esempio: il *MDP*

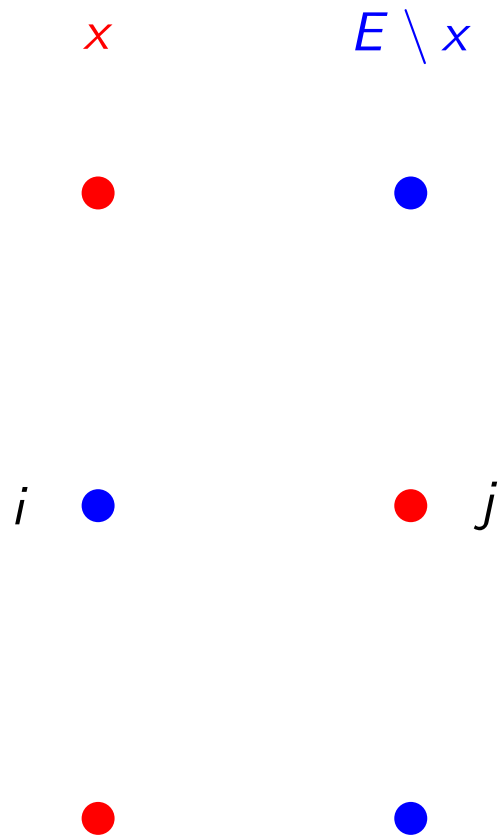
Per brevità, dimezziamo l'obiettivo $f(x)$
Valutiamo lo scambio

$$x \to x' = x \setminus \{i\} \cup \{j\}$$

con $i \in x$ e $j \in B \setminus x$

$$f(x') = f(x) - D_i + D_j - d_{ij}$$

- Si perdono le coppie contenenti i
- Si acquistano le coppie contenenti j
- Ma la coppia (i, j) è di troppo



Esempio: il *MDP*

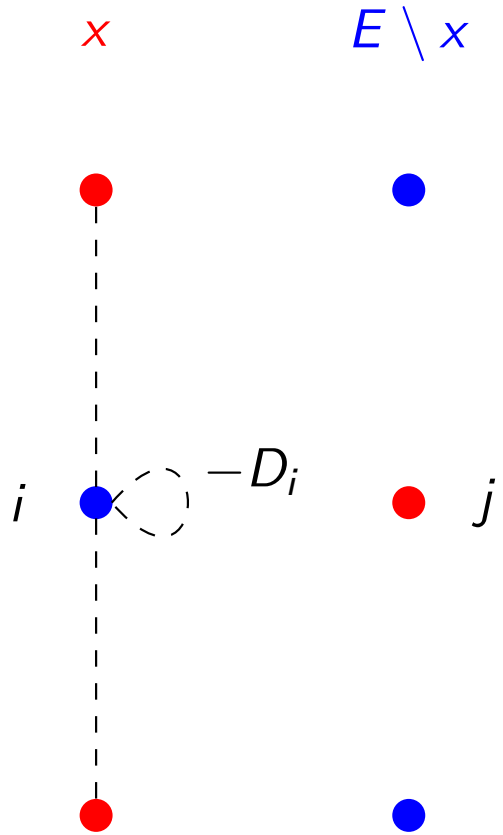
Per brevità, dimezziamo l'obiettivo $f(x)$
Valutiamo lo scambio

$$x \to x' = x \setminus \{i\} \cup \{j\}$$

con $i \in x$ e $j \in B \setminus x$

$$f(x') = f(x) - D_i + D_j - d_{ij}$$

- Si perdono le coppie contenenti i
- Si acquistano le coppie contenenti j
- Ma la coppia (i, j) è di troppo



Esempio: il *MDP*

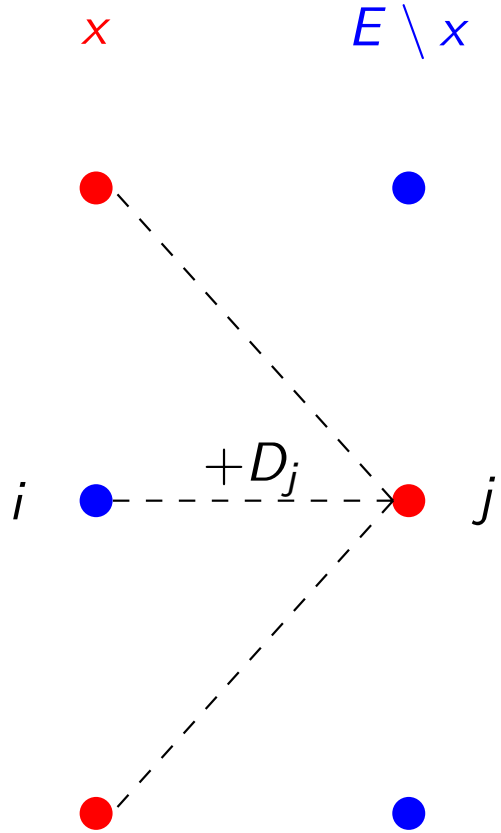
Per brevità, dimezziamo l'obiettivo $f(x)$
Valutiamo lo scambio

$$x \to x' = x \setminus \{i\} \cup \{j\}$$

con $i \in x$ e $j \in B \setminus x$

$$f(x') = f(x) - D_i + D_j - d_{ij}$$

- Si perdono le coppie contenenti i
- Si acquistano le coppie contenenti j
- Ma la coppia (i, j) è di troppo



Esempio: il *MDP*

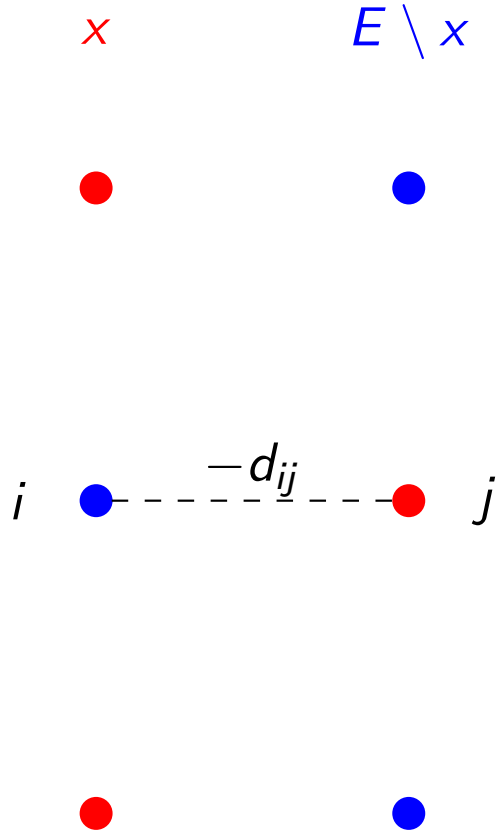
Per brevità, dimezziamo l'obiettivo $f(x)$
Valutiamo lo scambio

$$x \to x' = x \setminus \{i\} \cup \{j\}$$

con $i \in x$ e $j \in B \setminus x$

$$f(x') = f(x) - D_i + D_j - d_{ij}$$

- Si perdono le coppie contenenti i
- Si acquistano le coppie contenenti j
- Ma la coppia (i, j) è di troppo



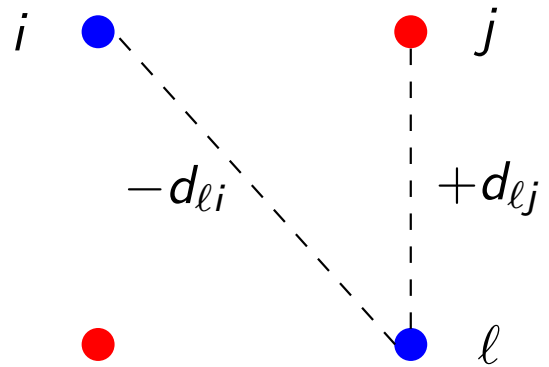
Esempio: il *MDP*

Update delle strutture dati:

- $D_\ell = D_\ell - d_{\ell i} + d_{\ell j}$, $\ell \in E$

Ogni elemento ℓ vede

- sparire $d_{\ell i}$
- comparire $d_{\ell j}$



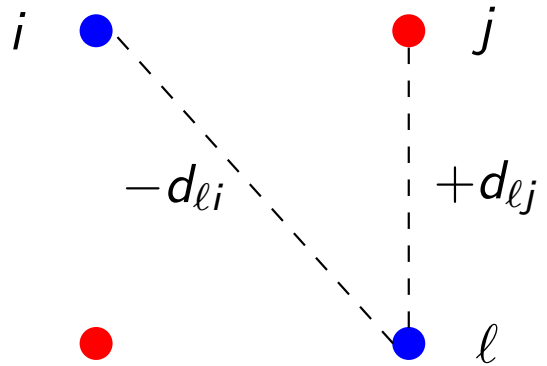
Esempio: il *MDP*

Update delle strutture dati:

- $D_\ell = D_\ell - d_{\ell i} + d_{\ell j}\,,\ \ell \in E$

Ogni elemento ℓ vede

- sparire $d_{\ell i}$
- comparire $d_{\ell j}$



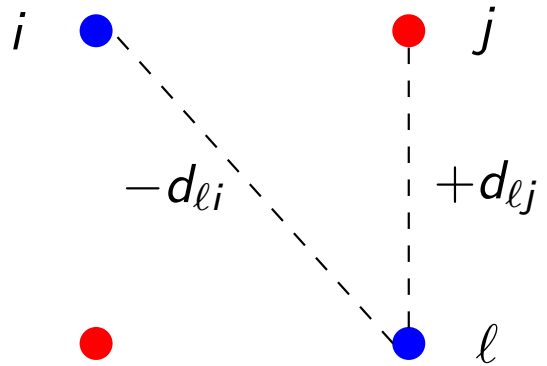
Esempio: il *MDP*

Update delle strutture dati:

- $D_\ell = D_\ell - d_{\ell i} + d_{\ell j}$, $\ell \in E$

Ogni elemento ℓ vede

- sparire $d_{\ell i}$
- comparire $d_{\ell j}$



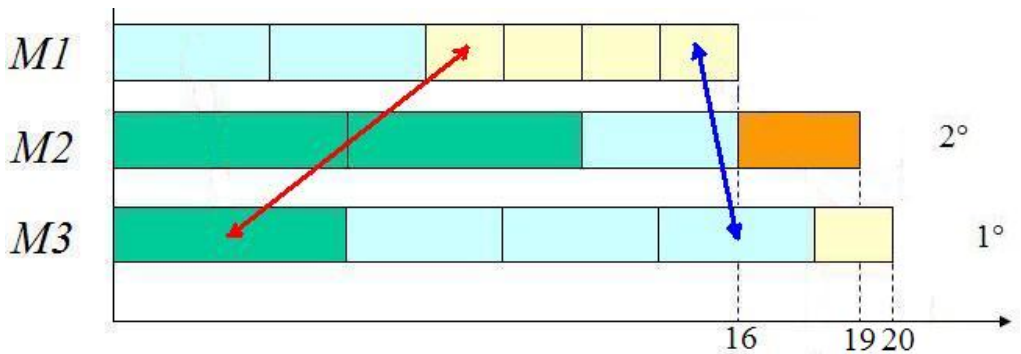
Uso di informazioni ausiliarie

Molte funzioni non lineari si possono aggiornare in modo simile

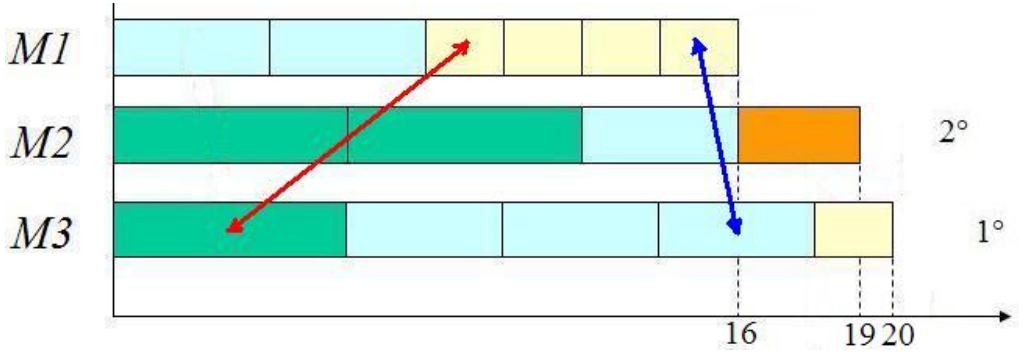
- si conservano informazioni aggregate sulla soluzione corrente $x^{(t)}$
- si usano per calcolare $f(x')$ in modo efficiente per ogni $x' \in N\left(x^{(t)}\right)$
- si aggiornano quando si passa alla soluzione seguente $x^{(t+1)}$

Per il *PMSP* con gli intorni di trasferimento $N_{\mathcal{T}_1}$ e scambio $N_{\mathcal{S}_1}$, si può valutare l'obiettivo in tempo costante conservando e aggiornando

- il tempo di completamento per ogni macchina
- gli indici delle macchine con il tempo massimo e con il secondo



Esempio: il *PMSP*



Consideriamo lo scambio $o = (i, j)$ fra le lavorazioni i e j
(i sulla macchina M_i, j sulla macchina M_j)

- i nuovi tempi di completamento si calcolano in tempo costante: uno dei due cresce, l'altro cala (o rimangono costanti)
- si verifica in tempo costante se uno dei due supera il massimo
- se cala il tempo massimo, si verifica in tempo costante se l'altro tempo oppure il secondo diventa il massimo

Visitato tutto l'intorno e scelto lo scambio, bisogna aggiornare

- i due tempi di completamento modificati (ognuno in tempo costante)
- la loro posizione nel max-heap (ognuno in tempo $O(\log |M|)$)

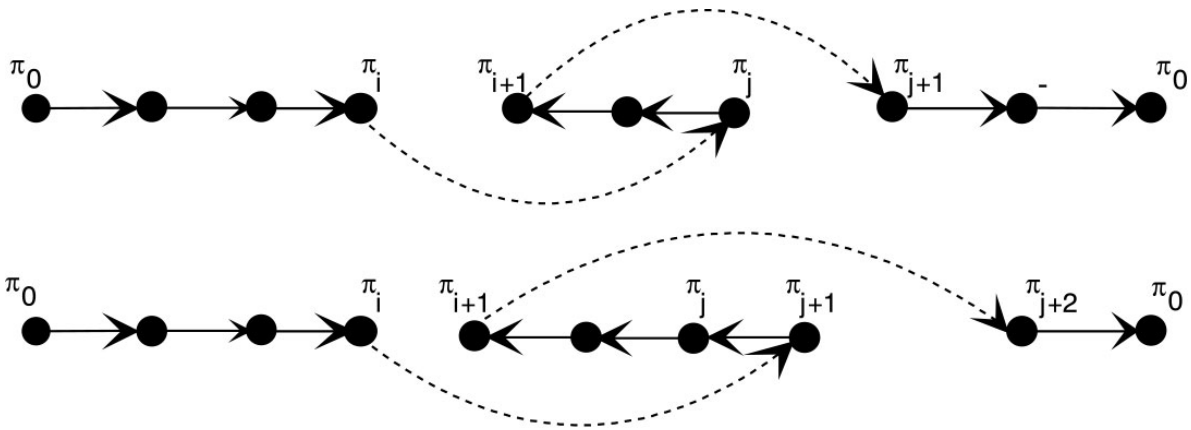
Uso di informazioni ausiliarie

Le informazioni possono riferirsi

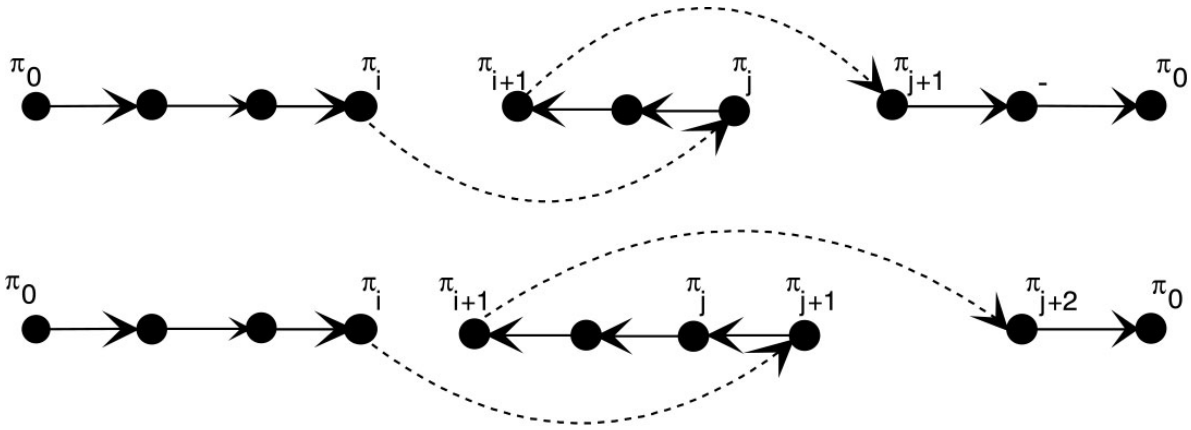
- alla soluzione corrente x
- alla precedente soluzione dell'intorno, seguendo un ordine opportuno

Consideriamo l'intorno $N_{\mathcal{R}_2}$ per il *TSP* asimmetrico:

- le soluzioni dell'intorno differiscono da x per $O(n)$ archi
- le soluzioni dell'intorno differiscono fra loro per $O(n)$ archi
- se le coppie di archi (s_i, s_{i+1}) e (s_j, s_{j+1}) seguono l'ordine lessicografico, il percorso invertito cambia di un solo arco



Esempio: il *TSP* asimmetrico



In generale, la variazione di $f(x)$ è

$$\delta f^{(ij)}(x) = c_{s_i,s_j} + c_{s_{i+1},s_{j+1}} - c_{s_i,s_{i+1}} - c_{s_j,s_{j+1}} + c_{s_j \ldots s_{i+1}} - c_{s_{i+1} \ldots s_j}$$

Passando dallo scambio (s_i, s_j) allo scambio (s_i, s_{j+1})

- i primi quattro termini cambiano, ma sono dati del problema
- gli ultimi due termini sono aggiornabili in tempo costante

$$\begin{cases} c_{s_{j+1} \ldots s_{i+1}} = c_{s_j \ldots s_{i+1}} + c_{s_{j+1},s_j} \\ c_{s_{i+1} \ldots s_{j+1}} = c_{s_{i+1} \ldots s_j} + c_{s_j,s_{j+1}} \end{cases}$$

Che impatto ha imporre un ordine predefinito di visita dell'intorno?

E l'ammissibilita'?

Alcune operazioni dell'intorno possono generare sottoinsiemi inammissibili

$$\tilde{N}_{\mathcal{O}}(x) = \{x' \subseteq B : \exists o \in \mathcal{O} : o(x) = x'\} \supseteq N_{\mathcal{O}}(x)$$
$$N_{\mathcal{O}}(x) = \{x' \subseteq X : \exists o \in \mathcal{O} : o(x) = x'\} = \tilde{N}_{\mathcal{O}}(x) \cap X$$

(*per esempio, questo vale per il KP, BPP, SCP, CMSTP...*)

In tal caso, per ogni elemento di $\tilde{N}_{\mathcal{O}}(x)$ bisogna

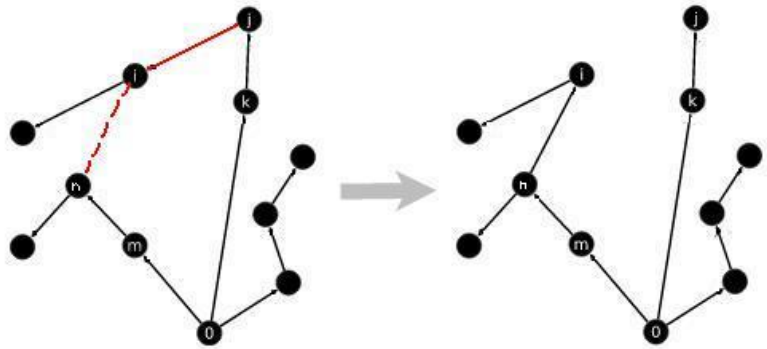
- valutare l'ammissibilità
- per quelli ammissibili valutare il costo

Per valutare l'ammissibilità si usano le stesse tecniche usate per il costo

Esempio: il *CMSTP*

Consideriamo l'intorno N_{S_1} che aggiunge un lato e ne toglie un altro

- se i due lati sono nello stesso ramo, la soluzione rimane ammissibile
- se sono in rami diversi, uno perde peso, l'altro lo acquista:
  la variazione è pari al peso del sottoalbero trasferito



Se si conserva per ogni vertice il peso del sottoalbero appeso,
basta confrontare tale peso con la capacità residua del ramo che lo riceve

Eseguito lo scambio ottimo, questa informazione va aggiornata in tempo $O(n)$ visitando l'albero (*ogni vertice ha un sottoalbero appeso*)

Uno schema generale di esplorazione sofisticata

L'uso di informazioni ausiliarie comporta

1. l'inizializzazione di apposite
   - strutture dati locali, relative all'esplorazione del singolo intorno
   - strutture dati globali, relative all'intero procedimento di ricerca
2. il loro aggiornamento da una soluzione o da un'iterazione all'altra

Algorithm SteepestDescent$\left(I, x^{(0)}\right)$

$x := x^{(0)}$; $GD :=$ InitializeGD$\left(x^{(0)}\right)$; Fine := *false*;

While Fine = *false do*

  $\tilde{x} := x$; $LD :=$ InitializeLD$(\tilde{x})$

  For each $x' \in N(x)$ *do*

    If $f(x') < f(\tilde{x})$ *then* $\tilde{x} := x'$;

    $LD :=$ UpdateLD(LD, x')

  EndFor;

  If $f(\tilde{x}) < f(x)$

    then $x := \tilde{x}$; $GD :=$ UpdateGD$(GD, \tilde{x})$

    else Fine := *true*;

  EndIf

EndWhile;

Return $(x, f(x))$;

Conservazione parziale dell'intorno

Eseguendo un'operazione $o \in \mathcal{O}$ su due soluzioni simili $x, x' \in X$, spesso

1. l'operazione è ammissibile per entrambe le soluzioni

$$o(x) \in X \Leftrightarrow o(x') \in X \text{ per ogni } o \in \mathcal{O}_{xx'}$$

2. la variazione della funzione obiettivo è la stessa

$$\delta f^{(o)}(x) = \delta f^{(o)}(x') \text{ per ogni } o \in \mathcal{O}_{xx'}$$

dove $\mathcal{O}_{xx'} \subset \mathcal{O}$ è un sottoinsieme ampio di $\mathcal{O}$

Questo perché ammissibilità e valore dell'obiettivo spesso dipendono solo dagli elementi aggiunti e tolti, che sono gli stessi per x e x'

In tali casi, conviene

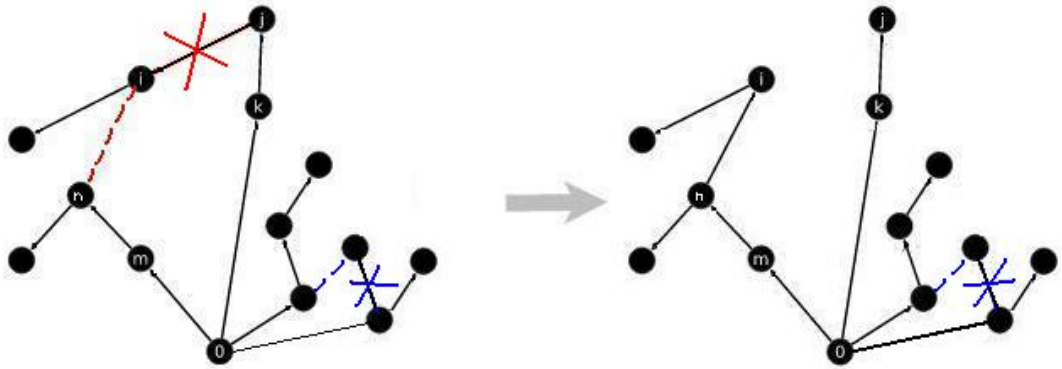
1. calcolare $\delta f^{(o)}(x)$ per ogni $o \in \mathcal{O}$ e conservare i valori a parte
2. eseguire l'operazione o^* migliore, generando la nuova soluzione x'
3. calcolare $\delta f^{(o)}(x')$ solo per $o \in \mathcal{O} \setminus \mathcal{O}_{xx'}$ e recuperare i valori salvati per $o \in \mathcal{O}_{xx'}$, che sono ancora validi
4. tornare al punto 2

Esempio: il *CMST*

Si consideri l'intorno N_{S_1} per il *CMST*:

- aggiunta di un lato $j \in E \setminus x$
- cancellazione di un lato $i \in x$

Sono coinvolti due rami: uno acquista un sottoalbero, l'altro lo perde



Eseguito lo scambio, l'effetto di scambi che riguardano altri rami rimane invariato

$$\delta f^{(i,j)}(x) = \delta f^{(i,j)}(x')$$

Quindi si può

- conservare l'insieme degli scambi ammissibili, con i relativi δf
- eseguire lo scambio ottimo $(\tilde{i}, \tilde{j})$ e cancellare dall'insieme gli scambi che coinvolgono uno dei due rami coinvolti in esso
- ricalcolare gli scambi che coinvolgono uno di tali rami e aggiungerli all'insieme

Se i rami sono molti, il risparmio è molto consistente

Compromesso efficienza-efficacia

La complessità dipende da tre fattori

1. numero dei passi di ricerca locale
2. ampiezza dell'intorno visitato
3. calcolo del costo della singola soluzione

Supponendo pari l'efficacia, i primi due fattori sono in evidente conflitto:

- intorno piccolo: è veloce da visitare, ma richiede molti passi per raggiungere un ottimo locale
- intorno ampio: richiede pochi passi, ma è lento da visitare

Il compromesso ottimo sta da qualche parte nel mezzo: serve un intorno

- abbastanza ampio da garantire la connessione forte
- abbastanza piccolo da essere visitato rapidamente

A priori non è facile localizzare il compromesso migliore, perché

- l'efficacia non è pari: intorni ampi hanno ottimi locali migliori
- l'efficienza cala rapidamente al crescere dell'intorno

Modulazione fine degli intorni

È anche possibile definire un intorno N e modularne la dimensione

- si esplora solo un sottointorno $N' \subset N$ promettente
  Per esempio, se la funzione obiettivo è additiva, si possono
  - aggiungere solo elementi $j \in B \setminus x$ di costo ϕ_j basso
  - togliere solo elementi $i \in x$ di costo ϕ_i alto
- si termina la visita dopo aver trovato una soluzione promettente
  Per esempio, la strategia first-best interrompe l'esplorazione alla prima soluzione migliore di quella corrente

$$\text{If } f(\tilde{x}) < f(x) \text{ then Stop} := \text{true;}$$

L'efficacia dipende dall'obiettivo

- se il costo di alcuni elementi influisce molto sull'obiettivo, vale la pena di tenerne conto, fissandoli o vietandoli

e dalla struttura dell'intorno

- se il *landscape* è liscio, la prima soluzione migliorante approssima bene la miglior soluzione dell'intorno: conviene fermarsi
- se il *landscape* è accidentato, la soluzione migliore dell'intorno potrebbe essere molto migliore: conviene proseguire