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

Proseguire la ricerca locale peggiorando

Anziché ripetere la ricerca locale, si può proseguirla, ma se non cambiano intorno e obiettivo, bisogna accettare soluzioni non minime

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

ed eventualmente anche non miglioranti

Il problema principale è il rischio di visitare ciclicamente soluzioni uguali

Le due principali strategie che consentono di controllarlo sono

- il *Simulated Annealing* (*SA*), che usa passi casuali
- il *Tabu Search* (*TS*), che usa memoria

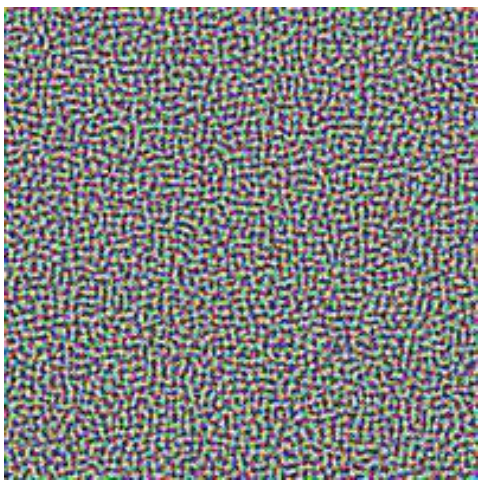
Annealing

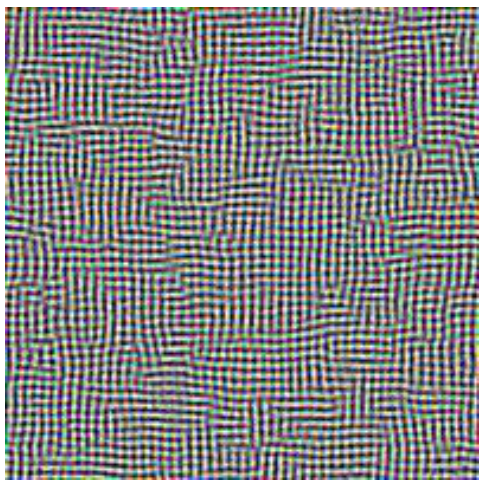
Il *SA* deriva dall'algoritmo di Metropolis (1953), che intende simulare il processo di "ricottura" dei metalli:

- si porta il metallo a temperatura vicina a quella di fusione, così che le particelle si dispongano in modo casuale
- si raffredda molto lentamente, così che l'energia cali, ma ci sia sempre il tempo di convergere all'equilibrio termico

Lo scopo del processo è di ottenere

- un reticolo cristallino molto regolare e privo di difetti, che corrisponde allo stato base (configurazione di energia minima)
- un materiale dotato di proprietà fisiche utili





Simulated Annealing

La situazione ha analogie coi problemi di Ottimizzazione Combinatoria

- gli stati del sistema fisico corrispondono alle soluzioni
- l'energia corrisponde alla funzione obiettivo
- lo stato base corrisponde alle soluzioni globalmente ottime (minime)
- le transizioni di stato corrispondono a mosse di ricerca locale
- la temperatura corrisponde a un parametro numerico

Questo suggerisce di usare l'algoritmo di Metropolis per l'ottimizzazione

Secondo la termodinamica all'equilibrio termico ogni stato i, di energia E_i ha probabilità

$$\pi'_T(i) = \frac{e^{\frac{-E_i}{k\,T}}}{\sum_{j \in S} e^{\frac{-E_j}{k\,T}}}$$

con S insieme degli stati, T temperatura e k costante di Boltzmann

È un equilibrio dinamico, con continue transizioni da stato a stato

L'algoritmo di Metropolis

L'algoritmo di Metropolis genera una sequenza casuale di stati

- lo stato corrente i ha energia E_i
- l'algoritmo perturba i, generando uno stato j con energia E_j
- lo stato corrente passa da i a j con probabilità

$$\pi_T(i,j) = \begin{cases} 1 & \text{se } E_j < E_i \\ e^{\frac{E_i - E_j}{k\,T}} = \frac{\pi'(j)}{\pi'(i)} & \text{se } E_j \geq E_i \end{cases}$$

cioè lo spostamento è

- deterministico se migliora (che è lo scopo finale)
- basato sulla probabilità condizionata se peggiora

Il *Simulated Annealing* applica esattamente lo stesso procedimento

Schema del *Simulated Annealing*

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

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

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

While Fine() $=$ *false do*

$x' :=$ RandomExtract(N, x); { estrazione casuale uniforme }

If $f(x') < f(x)$ *or* $U[0;1] < e^{\frac{f(x)-f(x')}{T}}$ *then* $x := x'$;

If $f(x') < f(x^*)$ *then* $x^* := x'$;

$T :=$ Aggiorna(T);

EndWhile;

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

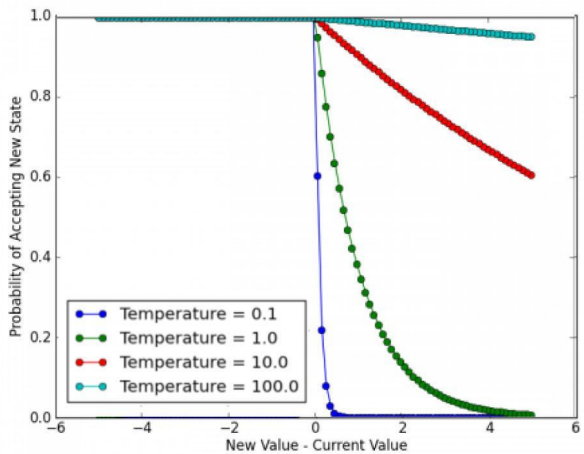
Si noti che è possibile peggiorare anche se esistono soluzioni miglioranti perché l'intorno non viene esplorato interamente

Criterio di accettazione

$$\pi_T(x, x') = \begin{cases} 1 & \text{se } f(x') < f(x) \\ e^{\frac{f(x)-f(x')}{T}} & \text{se } f(x') \geq f(x) \end{cases}$$

La "temperatura" T regola la probabilità di accettare peggioramenti

- con $T \gg 0$ vengono accettati quasi sempre: si tende a diversificare, al limite come in una *random walk*
- con $T \approx 0$ vengono rifiutati quasi sempre: si tende ad intensificare, al limite come in una *steepest descent*



Notate l'analogia con la ILS?

Convergenza all'ottimo

La probabilità che la soluzione corrente sia x' è la somma su tutti i possibili predecessori x delle probabilità di

- estrarre la mossa (x, x'), che è uniforme
- accettare la mossa, che è

$$\pi_T(x, x') = \begin{cases} 1 & \text{se } f(x') < f(x) \\ e^{\frac{f(x)-f(x')}{T}} & \text{se } f(x') \geq f(x) \end{cases}$$

dunque, ad ogni passo dipende solo dalla probabilità al passo prima: la variabile aleatoria x forma nel tempo una catena di Markov

Ad ogni temperatura fissata, le probabilità di transizione sono uniformi: si ha una catena di Markov omogenea

Se lo spazio di ricerca è connesso rispetto all'intorno N, la probabilità di raggiungere ogni stato è > 0 e si ha una catena di Markov irriducibile

Sotto queste ipotesi, la probabilità tende a una distribuzione stazionaria indipendente dalla soluzione iniziale

Convergenza all'ottimo

La distribuzione stazionaria è quella indicata dalla termodinamica per l'equilibrio termico dei sistemi fisici, che privilegia le soluzioni "buone"

$$\pi_T(x) = \frac{e^{\frac{-f(x)}{T}}}{\sum_{x \in X} e^{\frac{-f(x)}{T}}} \quad \text{per ogni } x \in X$$

con X insieme delle soluzioni ammissibili e T parametro "temperatura"

Se $T \to 0$, la distribuzione tende a una distribuzione limite

$$\pi(x) = \lim_{T \to 0} \pi_T(x) = \begin{cases} \frac{1}{|X^*|} & \text{per } x \in X^* \\ 0 & \text{per } x \in X \setminus X^* \end{cases}$$

che corrisponde a una convergenza certa a una soluzione ottima globale

Convergenza all'ottimo

Il risultato però vale all'equilibrio, e valori bassi di T implicano

- alta probabilità di visitare un ottimo globale
- convergenza lenta all'ottimo (*molti scambi vengono rifiutati*)

In un tempo finito, non sempre usare T più basso migliora il risultato

D'altra parte, non è necessario visitare spesso le soluzioni ottime:
basta aver visitato almeno una volta una soluzione ottima

In pratica, la temperatura T viene aggiornata: parte alta, e va calando

Il valore iniziale $T^{[0]}$ viene scelto

- abbastanza alto da consentire di accettare molte mosse
- abbastanza basso da rifiutare le mosse peggiori

Un modo classico è campionare il primo intorno $N\left(x^{(0)}\right)$ e fissare $T^{[0]}$ tale da accettare una data frazione di $N\left(x^{(0)}\right)$ (ad es., il 90%)

Aggiornamento della temperatura

Si procede per fasi successive ($r = 0, \ldots, m$)

- ogni fase applica un valore $T^{[r]}$ costante per $\ell^{[r]}$ iterazioni
- $T^{[r]}$ viene via via aggiornata con un profilo esponenziale

$$T^{[r]} := \alpha^r T^{[0]}$$

- $\ell^{[r]}$ viene aggiornato
  - crescendo da una fase all'altra (spesso linearmente)
  - con valori legati al diametro del grafo di ricerca (quindi alla dimensione dell'istanza)

Avendo T variabile, la catena di Markov x non è omogenea, ma

- se T cala abbastanza lentamente, converge all'ottimo globale
- i parametri per definire il calo dipendono dall'istanza (in particolare da $f(\tilde{x}) - f(x^*)$, dove $\tilde{x}$ è il miglior ottimo locale non globale)

Ma questi valori non sono noti

Efficienza computazionale e varianti

Calcolare la probabilità con un'esponenziale può essere pesante: conviene precalcolare una tavola dei valori di $e^{\frac{\delta f}{T}}$ per ogni $\delta f = f(x) - f(x')$

Se la temperatura T dipende dai risultati ottenuti, si parla di SA adattivo

- T è tarato in modo che una data frazione di $N(x)$ sia lecita
- T cresce se la soluzione non migliora da molto, altrimenti cala

Tabu Search

Il *Tabu Search* (*TS*) proposto da Glover (1986) prosegue la ricerca confermando il meccanismo di scelta dell'euristica *steepest descent*

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

cioè scegliendo sempre la miglior soluzione nell'intorno di quella corrente

Senza accorgimenti, questo produce comportamenti ciclici nella ricerca, che ritorna a soluzioni già visitate

L'idea è vietare le soluzioni già visitate, imponendo alla ricerca un tabu

$$x' := \arg \min_{x \in N(x) \setminus V} f(x)$$

dove V è l'insieme delle soluzioni già visitate

Il principio in sé è semplicissimo; il punto fondamentale da discutere è come rendere efficiente l'imposizione del divieto

Ricerca locale con tabù

Un'euristica di scambio basata sull'esplorazione esaustiva dell'intorno con un tabù sulle soluzioni visitate richiede i seguenti passi:

1. valutare l'ammissibilità di ogni sottoinsieme prodotto dagli scambi (se non è possibile garantirla a priori)
2. valutare il costo di ogni soluzione ammissibile
3. valutare la condizione tabù di ogni soluzione ammissibile promettente
4. scegliere la miglior soluzione ammissibile non tabù

Un modo elementare per realizzare la valutazione del tabù è

- salvare le soluzioni visitate in una struttura opportuna (lista tabù)
- confrontare ogni soluzione esplorata con quelle tabù

Imporre il tabù in modo efficiente

Questa valutazione elementare del tabù però è molto inefficiente

- il confronto delle soluzioni al passo t richiede tempo $O(t)$ (*riducibile usando strutture come hash table e alberi di ricerca*)
- il numero di soluzioni visitate cresce indefinitamente nel tempo
- l'occupazione di memoria cresce indefinitamente nel tempo

Il *Cancellation Sequence Method* e il *Reverse Elimination Method* affrontano questi problemi, sfruttando il fatto che in generale

- le soluzioni visitate formano una catena con piccole variazioni
- poche soluzioni visitate cadono nell'intorno di quella corrente

L'idea è concentrarsi sulle variazioni

- conservare liste di mosse, anziché di soluzioni
- valutare le variazioni complessive eseguite, anziché i singoli passaggi
- trovare le soluzioni che hanno subito piccole variazioni complessive (soluzioni recenti o soggette a molte variazioni poi annullate)

Perché un tabù sulle soluzioni può essere inefficace

Vi sono poi fenomeni più sottili, che impattano sull'efficacia del metodo

Vietare le soluzioni visitate può avere due effetti negativi diversi:

- può sconnettere il grafo di ricerca, creando delle "cortine di ferro" invalicabili che bloccano la ricerca

  (*quindi sarebbe meglio evitare divieti assoluti*)

- può rallentare l'uscita dai bacini di attrazione, creando un effetto di "riempimento graduale" che rallenta la ricerca

  (*quindi sarebbe meglio allargare il divieto ad altre soluzioni*)

I due fenomeni suggeriscono rimedi esattamente opposti

Come combinarli?

Esempio

Un esempio molto degenere è fornito dal seguente problema

- l'insieme base B contiene i primi n numeri interi: $B = \{1, \ldots, n\}$
- tutti i sottoinsiemi sono ammissibili: $X = 2^B$
- l'obiettivo combina un termine additivo quasi uniforme $\phi_i = 1 + \epsilon i$ $(0 < \epsilon \ll 1)$ e un termine molto negativo se $x = B$, nullo altrimenti

$$f(x) = \begin{cases} \sum_{i \in x} (1 + \epsilon i) & \text{per } x \subset B \\ \sum_{i \in x} (1 + \epsilon i) - n - 1 & \text{per } x = B \end{cases}$$

Se consideriamo l'intorno delle soluzioni a distanza di Hamming ≤ 1

$$N_{H_1}(x) = \left\{ x' \in 2^B : d_H(x, x') \leq 1 \right\}$$

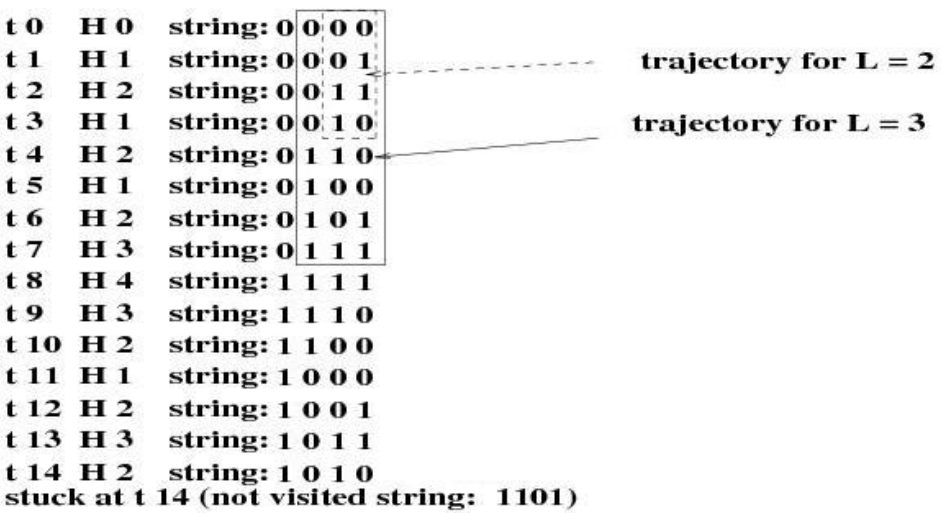
il problema ha

- un ottimo locale $\bar{x} = \emptyset$ con $f(\bar{x}) = 0$,
  il cui bacino di attrazione contiene tutte le soluzioni con $|x| \leq n - 2$
- un ottimo globale $x^* = B$, con $f(x^*) = n(n+1)\epsilon/2 - 1 < 0$,
  il cui bacino di attrazione contiene tutte le soluzioni con $|x| \geq n - 1$

Esempio

Se si parte da $x^{(0)} = \bar{x} = \emptyset$ e si applica il Tabu Search vietando le soluzioni visitate, si percorre una traiettoria che:

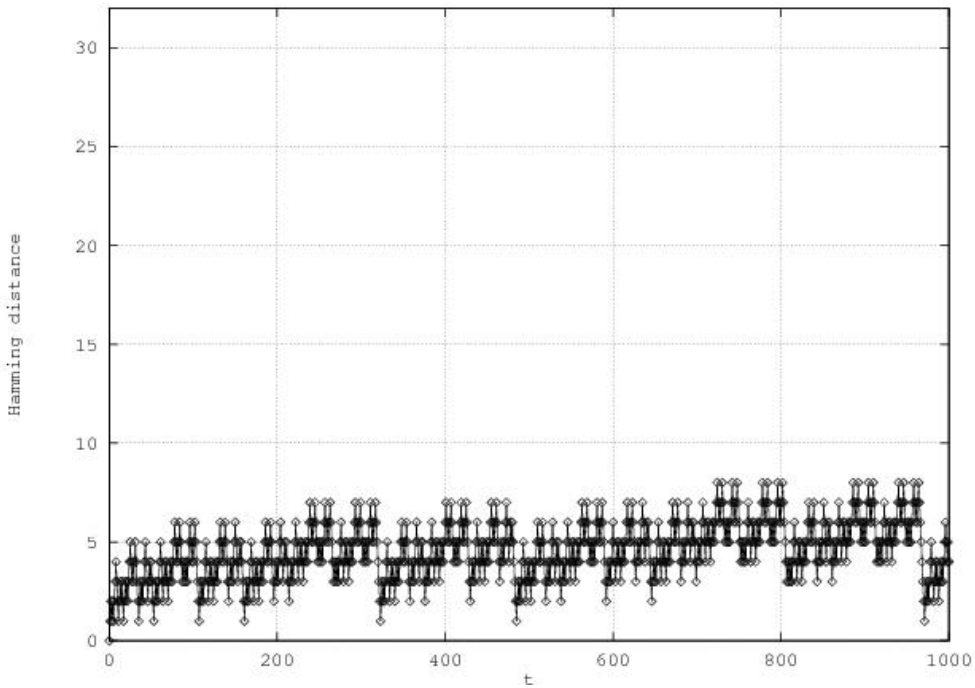
- scorre metodicamente buona parte dello spazio 2^B, allontanandosi da $\tilde{x}$ e riavvicinandovisi, con un continuo saliscendi di f
- per valori di $n \geq 4$ si blocca in una soluzione il cui intorno è stato interamente visitato, pur esistendo ancora soluzioni non visitate
- per valori di $n > 4$, non riesce a raggiungere l'ottimo globale



Nota: Nella figura gli elementi sono numerati da destra a sinistra

Esempio

L'andamento della funzione obiettivo conferma le difficoltà del metodo



La soluzione x si allontana da $x^{(0)} = \bar{x}$ e le si riavvicina ripetutamente

- percorre quasi interamente il bacino di attrazione di $\bar{x}$
- al termine non ne esce, ma rimane bloccata in una soluzione il cui intorno è completamente tabù
- se si rimuove il tabù più antico, l'esplorazione procede a vuoto, e ritorna il rischio di ciclare

Tabù basati su attributi

Per controllare questi problemi, si adottano alcuni semplici accorgimenti

1. vietare le soluzioni con "attributi" comuni con le soluzioni visitate, anziché limitarsi a vietare le soluzioni visitate
   - si definisce un insieme A di attributi
   - si gestisce un sottoinsieme $\bar{A}$ di attributi vietati (inizialmente vuoto)
   - sia $A_y \subseteq A$ il sottoinsieme di attributi posseduti dalla soluzione $y \in X$
   - si vietano tutte le soluzioni dotate di attributi vietati

$$A_y \cap \bar{A} \neq \emptyset \Leftrightarrow y \text{ è tabù}$$

   - se si esegue una mossa che trasforma la soluzione corrente da x a y, si aggiungono ad $\bar{A}$ gli attributi che x aveva e y non ha

$$\bar{A} := \bar{A} \cup (A_x \setminus A_y)$$

(*in questo modo, x diventa vietata*)

Questo significa che

- si evitano anche soluzioni simili a quelle visitate
- ci si allontana più in fretta dagli ottimi locali visitati

Tabù temporanei e criteri di aspirazione

Siccome il tabù crea zone difficili o impossibili da raggiungere

2. il divieto ha una durata limitata L
   - le soluzioni vietate tornano accessibili dopo un po'
   - si possono rivisitare le stesse soluzioni

   (*ma, se $\bar{A}$ è diverso, l'evoluzione futura sarà diversa*)

La tabu tenure L è un parametro fondamentale per l'efficacia del metodo

Siccome il tabù potrebbe vietare ottimi globali per semplice somiglianza

3. si introduce un criterio di aspirazione: una soluzione tabù che sia migliore della miglior soluzione nota viene comunque accettata

   (*ovviamente, non c'è rischio di ciclare*)

Esistono criteri di aspirazione più laschi, ma sono di uso poco comune

Infine, può capitare che tutte le soluzioni dell'intorno siano tabù

4. se tutte le soluzioni dell'intorno sono tabù, si accetta quella col tabù più vecchio (si può pensare come un altro criterio di aspirazione)

Schema del *Tabu Search*

Algorithm TabuSearch$(I, x^{(0)}, L)$

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

$\bar{A} := \emptyset$;

While Fine() $=$ *false do*

$x' := \arg \min_{y \in N(x): A_y \cap \bar{A} = \emptyset} f(y)$; { Trova la miglior soluzione non tabù }

$x'' := \arg \min_{y \in N(x): A_y \cap \bar{A} \neq \emptyset} f(y)$; { Trova la miglior soluzione tabù }

If $f(x'') < f(x^*)$ *and* $f(x'') < f(x')$

then $x := x''$;

else $x := x'$;

EndIf

$\bar{A} :=$ Aggiorna$(\bar{A}, x, L)$;

If $f(x) < f(x^*)$ *then* $x^* := x$;

EndWhile

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

Attributi tabù

Il concetto di "attributo" è volutamente generico; i più semplici sono

- appartenenza di un elemento alla soluzione ($A_x = x$):
  quando la mossa da x a y fa uscire un elemento i dalla soluzione, il tabù proibisce il reinserimento di i in soluzione
  - x ha l'attributo "presenza di i" e y non ce l'ha
  - l'attributo "presenza di i" entra in $\bar{A}$
  - ogni soluzione contenente i diventa tabù
- non appartenenza di un elemento alla soluzione ($A_x = B \setminus x$):
  quando la mossa da x a y fa entrare un elemento i dalla soluzione, il tabù proibisce l'eliminazione di i dalla soluzione
  - x ha l'attributo "assenza di i" e y non ce l'ha
  - l'attributo "assenza di i" entra in $\bar{A}$
  - ogni soluzione priva di i diventa tabù

Spesso si usano più attributi insieme, ognuno con la sua tenure e lista (per es., dopo aver sostituito i con j, si vieta di eliminare j per L^{in} passi e di aggiungere i per L^{out} passi, con $L^{\text{in}} \neq L^{\text{out}}$)

Attributi tabù

Altri esempi di attributo, meno frequenti

- il valore della funzione obiettivo: si vietano soluzioni di un dato valore, già assunto in precedenza dall'obiettivo
- il valore di una funzione ausiliaria (ad es., la distanza dalla miglior soluzione nota)

Attributi complessi si possono ottenere combinando attributi semplici

- la compresenza in soluzione di due elementi (o la loro separazione)
- oppure, se una mossa sostituisce l'elemento i con l'elemento j, il tabù può proibire l'uscita di j per far entrare i, consentendo invece la semplice uscita di j e il semplice ingresso di i

Valutazione efficiente del tabù

Come ammissibilità e costo, la valutazione del tabù deve essere efficiente: scorrere l'intera soluzione non è accettabile

- gli attributi sono associati alle mosse, e non alle soluzioni
  non si verifica se la soluzione contiene i, ma se la mossa reintroduce i

Si può valutare il tabù in tempo costante con una struttura che associa ad ogni attributo iterazione di inizio del tabù ($-\infty$ se mai tabù)

Per vietare gli inserimenti ($A = x$), ad ogni iterazione t

- valutando le mosse, è tabù inserire $i \in B \setminus x$ per ogni $t \leq T_i^{\text{in}} + L^{\text{in}}$
- eseguita la mossa, si pone $T_i^{\text{in}} := t$ per ogni i eliminato da x

Per vietare le eliminazioni ($A = B \setminus x$)

- valutando le mosse, è tabù eliminare $i \in x$ per ogni $t \leq T_i^{\text{out}} + L^{\text{out}}$
- eseguita la mossa, si pone $T_i^{\text{out}} := t$ per ogni i inserito in x

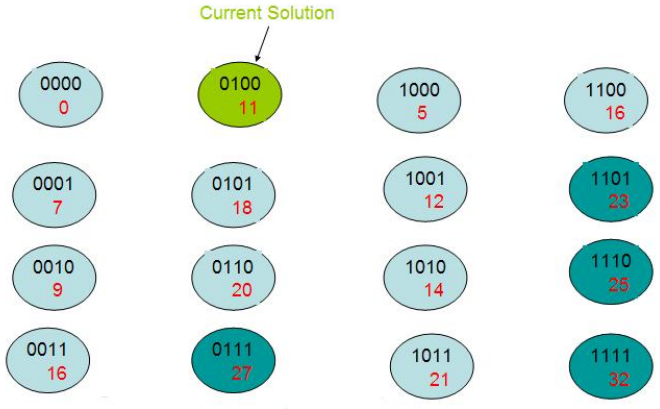
Basta un vettore solo per vietare entrambe, dato che o $i \in x$ o $i \in B \setminus x$

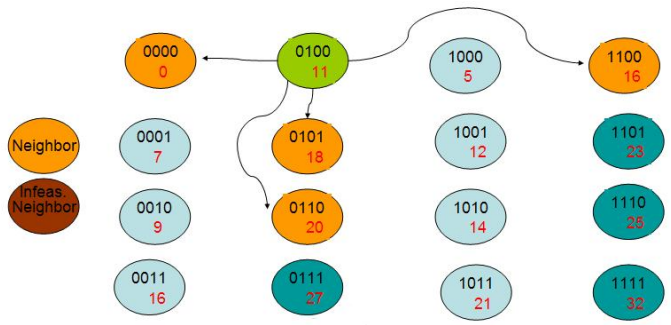
Per attributi più sofisticati, occorrono matrici o strutture più complesse

Esempio: il *KP*

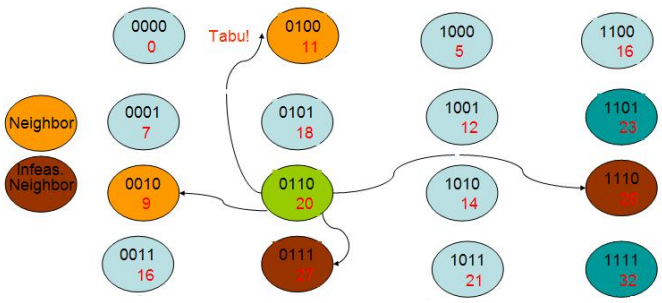
L'intorno $N_{\mathcal{H}_1}$ contiene le soluzioni a distanza di Hamming ≤ 1

Per semplicità usiamo l'attributo "presenza/assenza di un elemento i":
il vettore T registra l'iterazione dell'ultima mossa compiuta su ogni $i \in B$
Sia $L = 3$

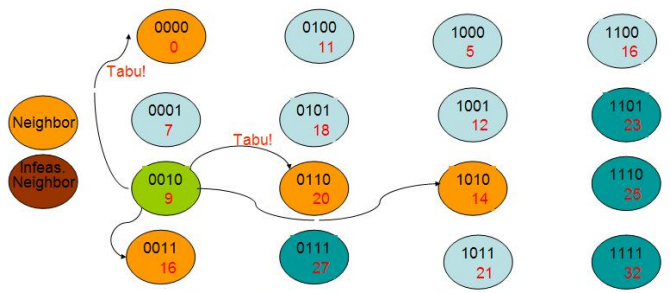




$t = 1 \quad T = [-\infty \quad -\infty \quad -\infty \quad -\infty]$



$t = 2 \quad T = [-\infty \quad -\infty \quad 1 \quad -\infty]$



$t = 3 \quad T = [-\infty \quad 2 \quad 1 \quad -\infty]$

Esempio: il *TSP*

Consideriamo l'intorno $N_{\mathcal{R}_2}$ generato dagli scambi 2-opt e
usiamo come attributi sia la presenza sia l'assenza di archi dalla soluzione

- all'inizio si pone $T_{ij} = -\infty$ per ogni arco $(i,j) \in A$
- ad ogni passo t, si esplorano le $n(n-1)/2$ coppie di archi eliminabili e le corrispondenti coppie di archi che le sostituirebbero
- la mossa (i,j), che sostituisce (s_i, s_{i+1}) e (s_j, s_{j+1}) con (s_i, s_j) e (s_{i+1}, s_{j+1}), è tabù al passo t se vale una delle seguenti condizioni:
  1. $t \leq T_{s_i,s_{i+1}} + L^{\text{out}}$
  2. $t \leq T_{s_j,s_{j+1}} + L^{\text{out}}$
  3. $t \leq T_{s_i,s_j} + L^{\text{in}}$
  4. $t \leq T_{s_{j+1},s_{i+1}} + L^{\text{in}}$

  Quindi, all'inizio tutte le mosse sono lecite
- scelta la mossa (i^*, j^*), si aggiornano le strutture ausilarie ponendo
  1. $T_{s_{i^*},s_{i^*+1}} := t$
  2. $T_{s_{j^*},s_{j^*+1}} := t$
  3. $T_{s_{i^*},s_{j^*}} := t$
  4. $T_{s_{j^*+1},s_{i^*+1}} := t$

Poiché n archi sono in soluzione e $n(n-2)$ fuori, conviene porre $L^{\text{out}} \ll L^{\text{in}}$

Esempio: il *Max-SAT*

Consideriamo l'intorno $N_{\mathcal{F}_1}$, che contiene le soluzioni ottenute invertendo il valore di una variabile (n soluzioni tutte ammissibili)

Poiché $|x| = |B \setminus x|$ per ogni $x \in X$

- non occorre differenziare il tabù per inserimenti ed eliminazioni
- è sufficiente vietare il cambio di valore di una variabile
  L'attributo è la coppia (variabile,valore opposto all'attuale)

L'algoritmo procede come segue

- all'inizio si pone $T_i = -\infty$ per ogni variabile $i = 1, \ldots, n$
- ad ogni passo t, si esplorano le n soluzioni ottenute assegnando ad ogni variabile il valore complementare
- la mossa i, che assegna $x_i := \bar{x}_i$ è tabù al passo t se $t \leq T_i + L$
  Quindi, all'inizio tutte le mosse sono lecite
- scelta la mossa i^*, si pone $T_{i^*} := t$

Taratura della *tabu tenure*

Il valore della *tabu tenure* L è un parametro fondamentale

- tenure troppo alte possono nascondere l'ottimo globale e nel caso peggiore bloccano del tutto la ricerca
- tenure troppo basse possono attardare l'esplorazione in regioni inutili e nel caso peggiore producono comportamenti ciclici

Il valore più efficace di L

- in generale è legato alla dimensione dell'istanza
- spesso cresce lentamente (molti suggeriscono $L \in O\left(\sqrt{n}\right)$)
- valori quasi costanti funzionano bene su ampi intervalli di dimensione

Estrarre L a caso da un intervallo $[L_{\min}; L_{\max}]$ rompe gli andamenti ciclici

Le tabu tenure adattive reagiscono ai risultati della ricerca aggiornando L entro un prefissato intervallo $[L_{\min}; L_{\max}]$

- L diminuisce quando la soluzione corrente x migliora: si pensa di avvicinarsi a un ottimo locale nuovo e si vuole favorire la ricerca (intensificazione)
- L aumenta quando la soluzione corrente x peggiora: si pensa di allontanarsi da un ottimo locale visitato e non si vuole rallentare (diversificazione)

Varianti

Sul lungo periodo, anche i metodi adattivi perdono efficacia

Si adottano allora strategie di lungo termine

- Tabu Search reattivo:
  - usa strutture efficienti per conservare le soluzioni visitate (*hash table*)
  - rileva eventuali comportamenti ciclici (ritorni frequenti)
  - se le soluzioni si ripetono troppo spesso, sposta l'intervallo $[L_{\min}; L_{\max}]$ verso valori più alti
- frequency-based Tabu Search:
  - conserva la frequenza di ogni attributo in soluzione in strutture analoghe a quelle usate per la recentezza (ad es., F_i per ogni $i \in E$)
  - se un attributo si presenta molto spesso
    - favorisce le mosse che lo introducono modificando f come nella *DLS*
    - vieta le mosse che lo introducono, o le sfavorisce modificando f
- Exploring Tabu Search: reinizializza la ricerca da soluzioni di buona qualità esplorate, ma mai assunte come soluzione corrente
  (*si tratta delle "seconde migliori soluzioni" di qualche intorno*)
- Granular Tabu Search: modifica l'intorno allargandolo via via