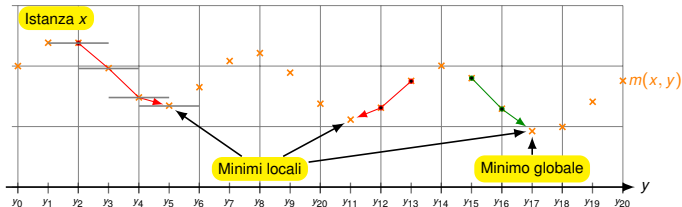


In questa lezione si richiamano i concetti fondamentali circa la tecnica della ricerca locale.

- 1 Si determina una soluzione ammissibile iniziale (soluzione corrente).
- 2 Si applica una trasformazione alla soluzione corrente per determinare una nuova soluzione "vicina".
- 3 Se il valore della nuova soluzione è migliore, si aggiorna la soluzione corrente alla nuova. Altrimenti si scarta.
- 4 Si ripetono i passi 2 e 3 fino a quando non sono possibili ulteriori trasformazioni.

Ottimo locale e ottimo globale

- L'algoritmo termina quando è determinato un ottimo locale che può non essere un ottimo globale.



- Diversi punti di partenza possono determinare diversi ottimi locali.
- Se $I(y)$ iniziale contiene tutte le soluzioni ammissibili, allora la ricerca determina sempre l'ottimo globale.
- Tale ricerca ha senso però solo se la dimensione di $I(y)$ è polinomiale nella dimensione dell'input.

Ordinamento per inserimento

	Costo
--	--------------

Introduzione

Caratteristiche fondamentali della ricerca locale:

- Esegue una ricerca esplorando un intorno della soluzione iniziale;
- Sceglie una nuova soluzione se questa è migliore di quella corrente;
- Termina quando non è possibile migliorare la soluzione corrente;
- Determina quindi un ottimo locale;
- Nulla di certo circa l'ottimalità globale della soluzione determinata;
- La complessità può essere esponenziale se la ricerca è molto fine.

Simulated annealing

Funzione simulatedAnnealing()

```

1:  $s = \text{init}(I)$ ,  $T = \text{initT}()$ ,  $t = 0$ ;
2: do // Raffredda il sistema gradualmente
3:   do // Data  $T$ , prova qualche soluzione
4:     selezione una soluzione  $s'$  nell'intorno di  $s$ ;
5:      $v = m(I, s) - m(I, s')$ ;
6:     if ( $v > 0$ ) then  $s = s'$  ; // Soluzioni migliori si accettano sempre
7:     else // le altre talvolta
8:       if ( $\text{rnd}[0, 1) < e^{\frac{v}{T}}$ ) then  $s = s'$ ;
9:   while ( $\neg \text{termCond}()$ );
10:   $T = \text{update}(T, t)$ ;  $t++$ ;
11: while ( $\neg \text{haltCond}()$ );
12: return  $s$ ;

```

Simulated annealing

- `initT()` sceglie la temperatura di partenza. Valore determinato sperimentalmente in base al problema.
- `termCond()` determina quante soluzioni si devono provare prima di abbassare la temperatura. Criterio determinato sperimentalmente.
- `update(T,t)` determina di quanto abbassare la temperatura. La velocità di abbassamento deve essere non troppo alta. Il criterio è determinato sperimentalmente in base al problema.
- `haltCond()` determina quando terminare la ricerca. Solitamente si termina quando T è sotto una soglia vicino allo 0.

Altri metodi di ricerca locale

Simulated annealing per SODDISFACIBILITÀ (Spears, 1993)

Problema della soddisfacibilità di espressioni booleane

PROBLEMA SODDISFACIBILITÀ (SAT)

DESCRIZIONE: Un'espressione booleana ϕ in forma normale congiunta.

QUESITO: ϕ è soddisfacibile? Ovvero, esiste un assegnamento di verità T per ϕ tale che $T \models \phi$?

Soluzione Simulated annealing di Spears (1993)

- La soluzione iniziale è un assegnamento di verità casuale.
- Una soluzione vicina è determinata invertendo il valore di una variabile.
- Propone come T iniziale 0.30 e come T finale 0.01.
- T è aggiornata come $T_{\max} \cdot e^{-t \cdot r}$ dove t è l'indice di ciclo e r è un fattore di decadimento $= \frac{1}{\#variabili \cdot \#tentativo}$

Altri metodi di ricerca locale

Simulated annealing per SODDISFACIBILITÀ (Spears, 1993)

Funzione SA-SAT(I , MAX_TRIES)

```

1:  $tries = 1$ ,  $T_{\max} = 0.3$ ,  $T_{\min} = 0.01$ ;
2: do
3:    $s$  = assegnamento casuale,  $r = 1/(|s| \cdot tries)$ ,  $t = 0$ ;
4:   do
5:     if ( $s$  soddisfa  $I$ ) then return  $s$ ;
6:      $T = T_{\max} \cdot e^{-t \cdot r}$ ;
7:     for ( $k = 1$ ;  $k \leq |s|$ ;  $k++$ ) do
8:        $\delta$  = variazione # clausole soddisfatte se si invertisse  $s_k$ ;
9:       inverti  $s_k$  con probabilità  $1/(1 + e^{-\frac{\delta}{T}})$ ;
10:    endfor
11:     $t++$ ;
12:  while ( $T > T_{\min}$ ) ;
13:   $tries++$ ;
14: while ( $tries \neq MAX\_TRIES$ ) ;
15: return nessun assegnamento;

```

Simulated annealing per SODDISFACIBILITÀ (Spears, 1993)

Altri metodi di ricerca locale

Simulated annealing per SODDISFACIBILITÀ (Spears, 1993)

Complessità SA-SAT

- $n = \# \text{variabili}$, $c = \# \text{clausole/variabile}$, $l = \# \text{letterali/clausola}$
- Costo soluzione casuale = $O(n)$;
- Costo verifica soluzione $q(n) = O(l \cdot c \cdot n)$;
- Costo ciclo for = $O(n \cdot q(n))$;
- Costo ciclo raffreddamento = $O(t \cdot n \cdot q(n))$;
- Costo globale = $O(MAX_TRIES \cdot t \cdot n \cdot q(n))$

Altri metodi di ricerca locale

Simulated annealing per SODDISFACIBILITÀ (Spears, 1993)

Indici sperimentali di SA-SAT

n	Clauses	δs	Flips	%Solved	MAX_TRIES
100	425	581,400	31,863	58	200,000
200	850	7,735,000	396,341	44	400,000
300	1275	37,474,500	1,924,040	48	800,000
400	1700	42,951,600	2,269,500	45	1,000,000
500	2125	86,680,500	4,438,820	41	1,600,000

Indici sperimentali di G-SAT (algoritmo greedy per SAT)

n	Clauses	δs	Flips	%Solved	MAX_FLIPS
100	425	541,875	21,250	50	500
200	850	12,673,500	497,000	-	2,000
300	1275	35,465,400	1,390,800	-	6,000
400	1700	89,943,600	3,527,200	-	8,000
500	2125	253,929,000	9,958,000	20-33	10,000

Altri metodi di ricerca locale

Simulated annealing

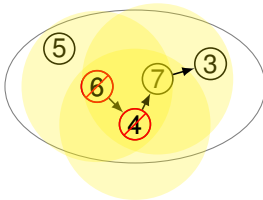
Considerazioni finali

- Spears ha dimostrato sperimentalmente che il vantaggio di SA-SAT rispetto all'algoritmo di ricerca locale deriva dal fatto che SA-SAT può fare più spostamenti consecutivi verso soluzioni peggiori;
- Simulated annealing è stato applicato con un certo successo anche al problema COMMESSO VIAGGIATORE e al problema PROGRAMMAZIONE NON LINEARE;

Altri metodi di ricerca locale

Tabu search

- Idea di base semplice: si forza la ricerca in nuove aree dello spazio delle soluzioni **proibendo** (**tabù**) di ritornare, anche in modo parziale, sui propri passi nell'immediato;
- L'obiettivo è raggiunto *memorizzando* le scelte fatte circa le componenti della soluzione corrente e proibendo di riutare/modificare tali scelte;
- Data una soluzione, si ricerca quindi una nuova soluzione, anche peggiore, nell'intorno di quella corrente evitando quelle che hanno portato a quest'ultima anche se sono migliori.



Altri metodi di ricerca locale

Tabu search

Esistono diversi tipi di **proibizioni** (tabu):

- Proibizione **fissa**: si impediscono le mosse recentemente fatte per un periodo fissato, poi si riammettono;
- Proibizione **randomizzata**: si impediscono le mosse recentemente fatte per un periodo casuale, poi si riammettono;
- Proibizione **reattiva**: si impediscono le mosse recentemente fatte per un periodo dipendente dalla qualità delle mosse; Date due mosse, se la prima aveva portato un miglioramento della soluzione maggiore della seconda, il suo tempo di proibizione è inferiore a quello della seconda.

Di seguito, consideriamo gli aspetti principali di questi tipi.

Altri metodi di ricerca locale

Tabu search: proibizione **fissa**

- $M = \{\mu_1, \dots, \mu_p\}$ l'insieme di mosse possibili;
- S_t la soluzione al tempo t e $I(S_t)$ le soluzioni vicine a S_t

$$I(S_t) = \{S \mid S \text{ è ottenibile da } S_t \text{ applicando 1 mossa } \mu_i \in M\}$$

- T , parametro di **proibizione fissa**, è la quantità di tempo di proibizione di riuso di una mossa.
- $I_P(S_T) \subset I(S_t)$ è insieme soluzioni **permesse**, determinate a partire da S_t applicando una mossa che non è stata usata durante le ultime T iterazioni.
- La ricerca della soluzione S_{t+1} avviene all'interno dell'insieme $I_P(S_T)$.

Altri metodi di ricerca locale

Tabu search: proibizione fissa

- $UU[\mu_i] =$ l'Ultima iterazione t nella quale è stato Usata μ_i ;
- Sia μ_i^{-1} la mossa inversa a μ_i ;
- Criterio della migliore mossa permessa:

$$I_P(S_t) = \{S \mid S \text{ è ottenibile da } S_t \text{ applicando 1 mossa } \mu_i \in M \\ \text{tale che } UU[\mu_i^{-1}] < t - T\}$$

$$S_{t+1} = S \in I_P(S_t) \text{ tale che } m(I, S) \leq m(I, S') \forall S' \in I_P(S_t)$$

- Non è garantito $m(I, S_{t+1}) \leq m(I, S_t)$: ci sono mosse proibite;
- Se $T = 0$ non ci sono proibizioni \implies ricerca locale classica;
- Maggiore T = maggiore # iterazioni prima di rivisitare una soluzione;
- T non può essere troppo grande: tutte le mosse diventerebbero proibite e la ricerca si bloccherebbe;
- T deve garantire almeno 2 mosse per ogni iterazione.

Altri metodi di ricerca locale

Tabu search: proibizione fissa

Esempio 4 (Minimizzare un intero dato da una sequenza di bit)

- S = sequenza di n bit;
- μ_i complementa il i -esimo bit $\implies \mu_i^{-1} = \mu_i$;
- $T \leq n - 2 \implies$ ogni iterazione ci sono almeno 2 mosse possibili;
- Se si eseguono solo mosse permesse, min intervallo di ripetizione $R = 2(T + 1)$ ovvero $S_{t+R} = S_t \implies R \geq 2(T + 1)$
- Sia $m(I, S)$ l'intero rappresentato da S . Il valore min è quando S rappresenta 0.

Altri metodi di ricerca locale

Tabu search: proibizione fissa

continua esempio

Simulazione quando $n = 4$ e $T = 2$, se si parte proprio dal minimo

	t	s_3	s_2	s_1	s_0	i
	0	0	0	0	0	0
	1	0	0	0	1	1
	2	0	0	1	1	3
$(T + 1)$	3	0	1	1	1	7
	4	0	1	1	0	6
	5	0	1	0	0	4
$2(T + 1)$	6	0	0	0	0	0

Altri metodi di ricerca locale

Tabu search: proibizione fissa per MINIMA BISEZIONE

PROBLEMA MINIMA BISEZIONE (MIN BISECTION)

DESCRIZIONE: Un grafo $G = (V, E)$ con $|V| = n$.

QUESITO: Determinare una partizione dei nodi (N_0, N_1) tale che
 $|N_0| = |N_1| = n/2$.

MISURA: $\min_{N_0, N_1} |\{(x, y) \mid (x, y) \in E' \wedge x \in N_0 \wedge y \in N_1\}|$.

Nota!

MINIMA BISEZIONE compare in divesi contesti applicativi. Uno è il campo del calcolo parallelo: il problema descrive in astratto la comunicazione tra due processori paralleli dove i nodi sono i task da svolgere e gli archi le precedenze da rispettare.

Altri metodi di ricerca locale

Tabu search: proibizione fissa per MINIMA BISEZIONE

Semplice algoritmo risolutore: **Min-Max-Greedy**

- Sia $|V| = n$ pari;
- Si selezionino casualmente due nodi e si assegnino, rispettivamente a N_0 e N_1 .
- Si prosegue aggiungendo alternativamente un nodo a N_0 e uno a N_1 nel seguente modo:
 - si sceglie il nodo i da aggiungere a N_k in modo che sia **minimo** # archi (i, j) con $j \in N_{1-k}$;
 - Se esistono più nodi che soddisfano il criterio, si sceglie quello con **massimo** # di archi (i, j) con $j \in N_k$;
 - Il buon comportamento è garantito dal secondo criterio;
 - Complessità $O(n^2)$ se si usa particolare struttura dati, detta a "cestini";
 - Prove sperimentali dimostrano una qualità delle soluzioni buona;
 - Se si vuole migliorare, la ricerca locale classica non aiuta.

Altri metodi di ricerca locale

Tabu search: proibizione fissa per MINIMA BISEZIONE

Consideriamo una soluzione tabu search search per migliorare l'algoritmo greedy:

- S = sequenza di n bit dove $s_i = 0$ indica che $v_i \in N_0$;
- μ_i complementa il i -esimo bit $\implies \mu_i^{-1} = \mu_i$;
- Data una soluzione S , non è detto che la soluzione successiva sia ammissibile: spostare un nodo altera la cardinalità degli insiemi!
- T_f è l'indice di proibizione fissa ($0 < T < 1$).

Altri metodi di ricerca locale

Tabu search: proibizione fissa per MINIMA BISEZIONE

Funzione TSF-BISEZIONE(G, T_f)

// $U[]$ è globale!

```

1:  $n = |V|$ ;  $MAX\_TRIES = 100n$ ;  $T = \lfloor nT_f \rfloor$ ;
2:  $(N_0, N_1) = \text{Min-Max-Greedy}(G)$ ;  $n_0 = |N_0|$ ;  $n_1 = |N_1|$ ;
3:  $N_0^* = N_0$ ;  $N_1^* = N_1$ ;  $m_{\min} = m(G, (N_0, N_1))$ ; // Soluzione e valore ottimo
4: for ( $j = 1$ ;  $j \leq MAX\_TRIES$ ;  $j++$ ) do
5:    $k = (n_0 > n/2) ? 1 : 0$ ;
6:    $i = \text{MossaMigliore}(1 - k, j, UU, T)$ ;
7:    $N_k = N_k \cup \{i\}$ ;
8:    $N_{1-k} = N_{1-k} \setminus \{i\}$ ;
9:    $UU[j] = j$ ;
10:  if ( $n_0 == n_1 \&\& m(G, (N_0, N_1)) < m_{\min}$ ) then
     $N_0^* = N_0$ ;  $N_1^* = N_1$ ;  $m_{\min} = m(G, (N_0, N_1))$ ;
11: endfor
12: return ( $N_0^*, N_1^*$ );
```

Tabu search: proibizione fissa per MINIMA BISEZIONE

- 42 / 47

Altri metodi di ricerca locale

Tabu search: proibizione randomizzata

Anziché determinare valori buoni per T , si può tentare un approccio casuale:

- ogni n iterazioni, si sceglie casualmente un nuovo T_f : in questo modo un cattivo valore influenza un numero finito di iterazioni.
- per rendere efficace l'approccio, l'algoritmo viene eseguito più volte, partendo sempre dalla soluzione greedy. La soluzione finale è data dalla soluzione migliore trovata durante tutte le esecuzioni.
- Prove sperimentali hanno mostrato che eseguendo $100n$ volte il programma e, per ciascuna volta, eseguendo $10n$ iterazioni, le soluzioni trovate sono paragonabili a quelle trovate dalla “ricerca tabù fissa” con il miglior T_f .

Esercizi

Scheduling

Esercizio 1

Dati n programmi con tempi di esecuzione t_1, \dots, t_n , si vuole eseguirli su un processore in modo da minimizzare il **tempo medio** di completamento $\frac{1}{n} \sum_i c_i$ dove c_i è il tempo di completamento del programma i , ovvero la durata di i più la durata di tutti i programmi che lo precedono nello scheduling finale.

Si progetti un algoritmo di ricerca locale per risolvere il problema.

Esercizi

Min-Max-Geedy

Esercizio 2 (Bertossi 24.4)

Le operazioni richieste dall'euristica Min-Max-Geedy possono essere realizzate usando una struttura dati detta “a cestino”. Vengono usate due matrici $cestino[0 \dots 1, 0 \dots n]$ e $dimensione[0 \dots 1, 0 \dots n]$.

In generale:

- $cestino[k, h]$ contiene la lista dei nodi del tipo i tale che il # archi (i, j) con $j \in N_k$ è uguale ad h ;
- $dimensione[k, h]$ è la lunghezza della lista $cestino[k, h]$.

continua...

Esercizi

Min-Max-Geedy

continua Bertossi 24.4

Inoltre sono mantenuti i valori $\min[0]$, $\min[1]$, $\max[0]$ e $\max[1]$ che rappresentano gli indici dei cestini con il min/max numero di nodi per i due sottoinsiemi.

Le operazioni sono:

- $\text{Inserisci}(i, \text{cestino})$: inserisce il nodo i in cestino ;
- $\text{Cancella}(i, \text{cestino})$: cancella il nodo i da cestino ;
- $\text{Incrementa}(k, i, \text{cestino})$: aumenta di 1 il numero di archi (i, j) con $j \in N_k$.

Si implementino in modo efficiente tale struttura dati e si dimostri come realizzare efficientemente la scelta del nodo i da aggiungere alla partizione.