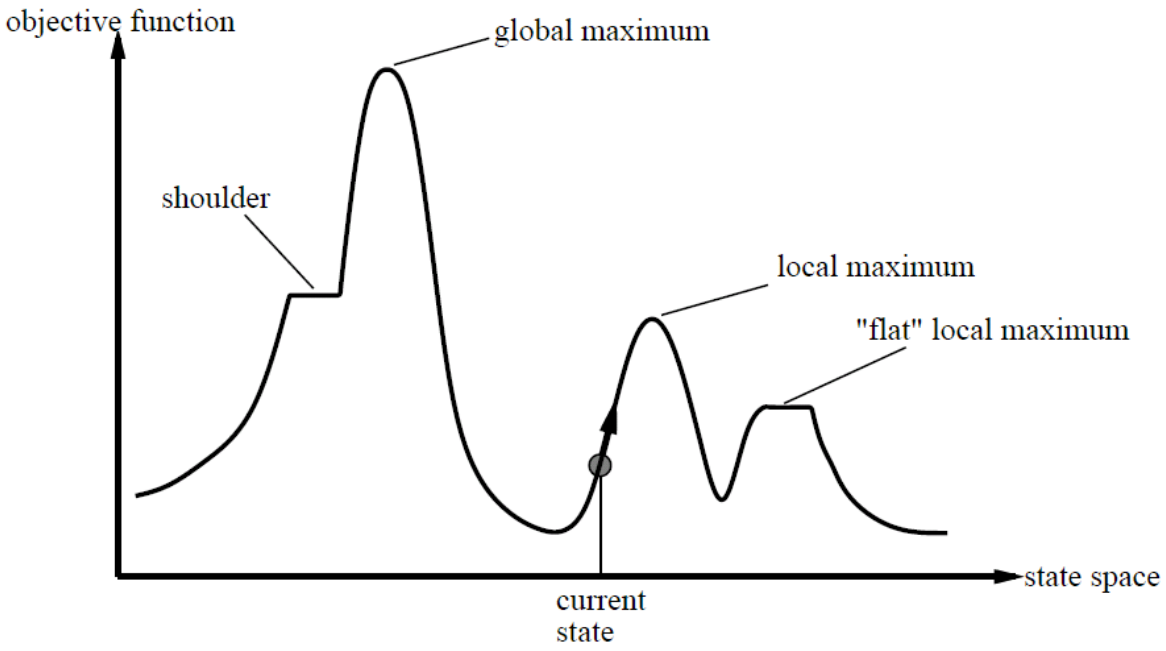
Algoritmi di ricerca locale

- Utilizzati in problemi di ottimizzazione
- Tengono traccia solo dello stato corrente e si spostano su stati adiacenti
- Necessario il concetto di vicinato di uno stato
- Non si tiene traccia dei cammini
- Ad ogni stato è associata una funzione obiettivo
- Nessuna garanzia di raggiungere l'ottimo globale ma prestazioni efficienti
- Garanzia di raggiungere un ottimo locale

Panorama dello spazio degli stati



- Uno stato ha una posizione sulla superficie e una altezza che corrisponde al valore della funzione obiettivo
- Un algoritmo di ottimizzazione provoca movimento sulla superficie
- Trovare l'avvallamento più basso o il picco più alto

Ricerca in salita (*Hill climbing*)

Dato lo stato attuale *i* e la funzione obiettivo *f*:

- Considerato l'insieme *N*(i) degli stati “vicini” di *i*, occorre definire un criterio per scegliere il successivo stato:
  i. il migliore, cioè quello che migliora di più la *f* (Hill climbing a salita rapida)
  ii. uno a caso tra quelli che migliorano, Hill climbing stocastico
  iii. il primo, Hill climbing con prima scelta

(*Hill climbing*)
Criteri di arresto

Per evitare che l'algoritmo impieghi troppo tempo in caso di spazi ad alta dimensione, occorre definire dei criteri di arresto:

- Numero massimo di iterazioni
- Miglioramenti esigui
  - Terminiamo quando nessuno degli stati vicini migliora la *f* di una quantità superiore ad un fissato valore ε
- Soluzione ottima localmente
  - Nessuna soluzione nel vicinato migliora quella attuale (massimo locale)

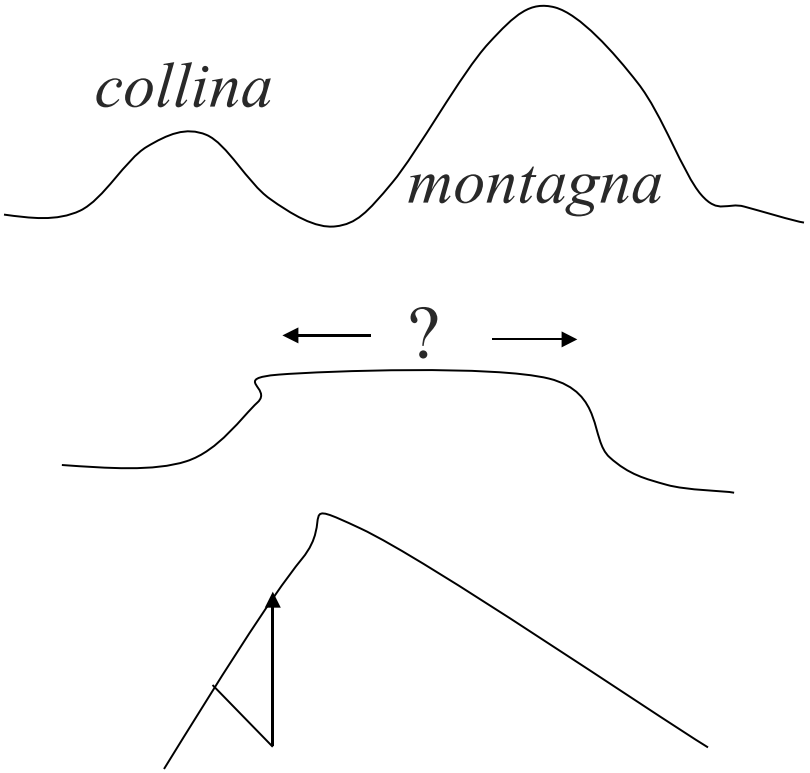
L'algoritmo Hill climbing

function Hill-climbing (*S*, *N*, *f)*
// S spazio degli stati
// N : S → P(S) funzione di vicinato
// f : S → R funzione obiettivo
    s ← *x in S*   // *s* Stato iniziale.
    loop do
        vicino ← *x in N*(*S*) // *x scelto con criterio (i)*
        if *f*(*vicino)* ≤ *f*(*s*) **then**
                return *s* // interrompe la ricerca. s è massimo locale
        s = vicino

Problemi con Hill-climbing

Se la f. è da massimizzare, si cercano i picchi (massimi locali o globali)

- Massimi locali
- Pianori o spalle
- Crinali (o creste)



Miglioramenti

1. Hill-climbing stocastico: si sceglie a caso tra le mosse in salita (che migliorano *f*)
   - converge più lentamente ma a volte trova soluzioni migliori
2. Hill-climbing con scelta casuale
   - generare le mosse a caso ed essere più efficace quando i successori sono molti
   - Convergenza lenta ma può evitare massimi locali non buoni

Miglioramenti (cont.)

1. Hill-Climbing con *riavvio casuale* (*random restart*): ripartire da un punto scelto a caso
   - Se la probabilità di successo è *p* saranno necessarie in media 1/p ripartenze per trovare la soluzione
   - Se funziona o no dipende molto dalla forma del panorama degli stati

Esercizio 1

- Sia G=(V,E, w) un grafo non diretto pesato, con funzione peso w: E → [-1,1]. V = {1, 2,...., n}
  - Nota che, essendo il grafo non diretto, w può essere rappresentata mediante una matrice simmetrica.
  - Si assume inoltre che w(i,i) = 0 per ogni i
- E' data inoltre anche una funzione L: $V \rightarrow \{-1,1\}$ che assegna ad ogni nodo una etichetta binaria, -1 oppure 1.
- Uno stato della rete è dato dalla sequenza di etichette x=($L(1)$,..., $L(n)$). La **funzione obiettivo** è $E(x) = x \cdot W \cdot x^T$.
- Determinare un ottimo locale di E mediante l'algoritmo di Hill Climbing, partendo da uno stato inziale a caso x_0 in $\{-1,1\}^n$
- $N(x)$ è l'insieme degli stati differenti da x in una sola componente
- Provare ad applicare diversi criteri di selezione e/o di miglioramento e a confrontare le soluzioni calcolate

Il simulated annealing (SA)

Analogia con il processo di solidificazione di un metallo fuso (Physical Annelaling)

- A partire dal metallo fuso, la temperatura viene abbassata lentamente e il sistema transita da uno stato energetico al successivo fino a quando il metallo solidifica nello stato di minima energia

10

Passi salienti del processo fisico

- Il sistema viene portato alla temperatura di fusione

- il sistema si trova in uno stato *i* e può transitare in uno stato *j* ottenuto perturbando lo stato *i*
- la transizione avviene seguendo il *criterio di Metropolis*

$$P = \exp\left(\frac{E_i - E_j}{k_B T} \right)$$

- **la temperatura deve essere abbassata lentamente in modo che il solido raggiunga l'equilibrio termico ad ogni temperatura**

11

Modello matematico

x	**stato del sistema**
f(x)	energia dello stato
parametro di controllo T	**temperatura**
funzione di T decrescente	**raffreddamento**
generatore di configurazioni x	**perturbazione**
$P_T\{j\} = \begin{cases} 1 & \text{se } f(j) \le f(i) \\ \exp\left(\frac{f(i) - f(j)}{T}\right) & \text{se } f(j) > f(i) \end{cases}$	**criterio di Metropolis**
ottimo globale	**minima energia**

12

Teoria matematica

- Descrizione mediante catene di Markov
- Convergenza alla soluzione ottima con probabilità 1 per un numero infinito di transizioni

13

Realizzazione dell'algoritmo: ricerca di massimo locale

1. Descrizione delle configurazioni e generatore casuale di configurazioni

2. Funzione obiettivo F e criterio di accettazione

3. Parametro di controllo T e schema di raffreddamento

14

Generazione casuale

Assumendo che lo stato $x \in \{-1, 1\}^n$

1. Dato *x(t)*, stato al tempo *t*, scegliamo casualmente $i \in \{1, 2, \ldots, n\}$ e settiamo $x_i = -x_i$ (flip), ottenendo *x(t+1)*

2.Una scelta alternativa può essere di scegliere in un ordine casuale, tutti gli $i \in \{1, 2, \ldots, n\}$ in *n* scelte consecutive, cioè evitiamo di scegliere di aggiornare due volte la stessa componente prima di aver aggiornato una volta tutte le altre componenti

15

Criterio di Metropolis: massimizzazione

Calcolo della funzione obiettivo della nuovo stato *x(t+1)* e confronto il precedente stato x(t) per il calcolo della probabilità di accettazione P_T {x(t+1)} di *x(t+1)*:

$$1 \text{ se } E(x(t+1)) \geq E(x(t))$$

$$\exp\left(\frac{E(x(t+1)) - E(x(t))}{T}\right) \quad \text{Se } E(x(t+1)) < E(x(t))$$

Generazione di un numero random r ∈]0,1[e confronto con P_T{t+1}: se P_T{x(t+1)} < r la nuova configurazione viene scartata.

16

Criterio di Metropolis

- Ad ogni passo t+1 si sceglie un nuovo stato a caso:
  - se migliora lo stato corrente x(t) viene espanso
  - se no (caso in cui E(x(t+1)) – E(x(t)) < 0), lo stato x(t+1) viene scelto comunque con probabilità

$$\exp\left(\frac{E(x(t+1))-E(x(t))}{T}\right)$$

- *T* descresce col progredire di *t* (quindi anche la probabilità di accettare peggioramenti) secondo un piano definito (valore iniziale e decremento sono parametri).
- Accettare peggioramenti all'inizio permette di esplorare meglio lo spazio dello soluzioni

17

Schema di raffreddamento

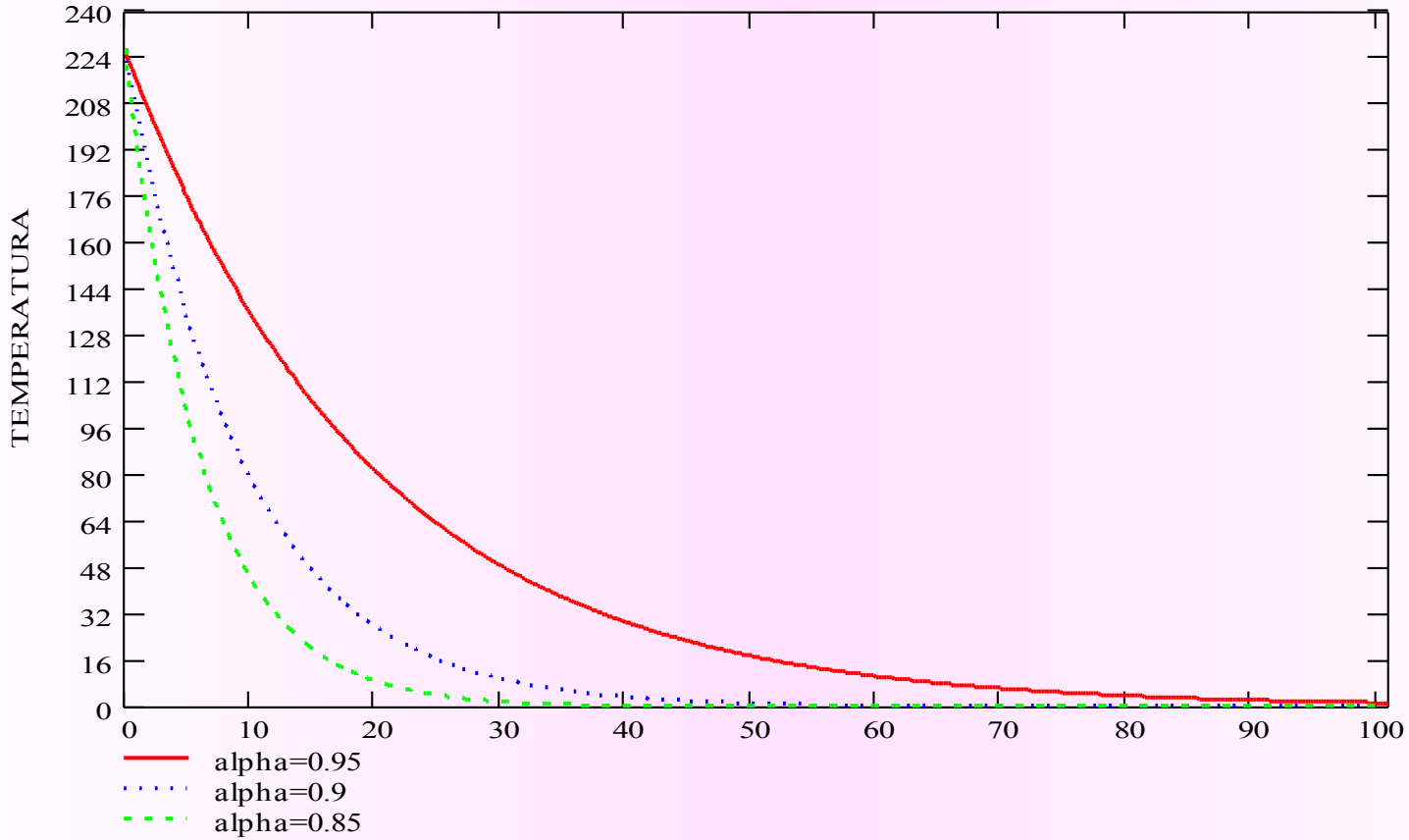
- Temperatura iniziale
- Numero di passi a temperatura costante
- Funzione di decremento
- Criterio di arresto

Il concetto fondamentale che guida la scelta di questi parametri è quello di *quasi equilibrio*

18

Schemi di raffreddamento

$$T_{k+1} = \alpha T_k \qquad 0.8 \leq \alpha \leq 0.99$$

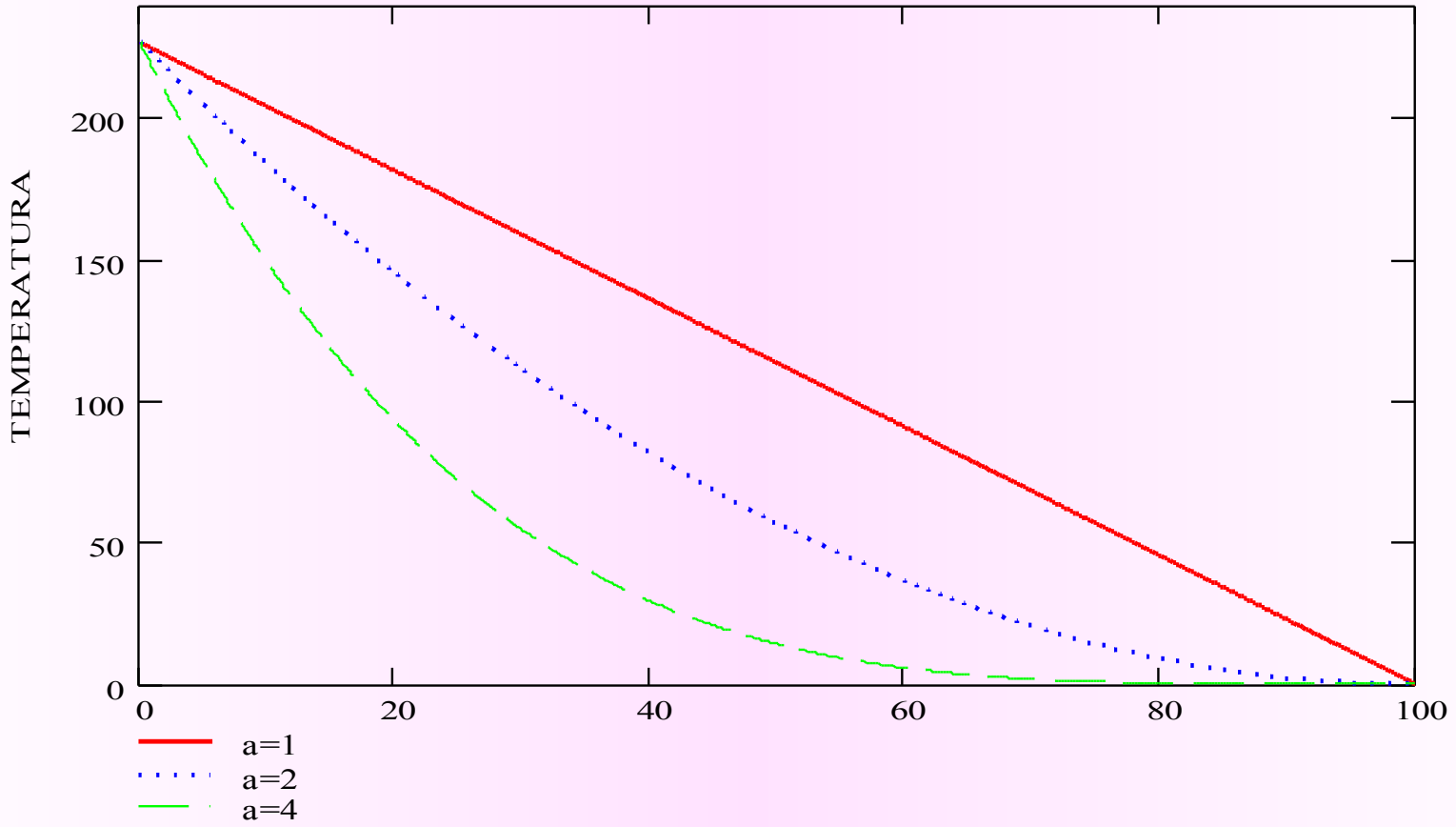


19

Schemi di raffreddamento

$$T_k = T_0\left(1 - k/K\right)^a \qquad a = 1,2\ o\ 4$$

$$K = numero\ \max\ decrementi$$



20

Criterio di arresto

- Al raggiungimento del valore ottimo (se noto)
- Numero massimo di iterazioni, o temperatura sotto un certo valore
- Miglioramento di energia al di sotto di un valore prefissato
- Combinazione di criteri

21

Esercizio

• Scrivere un programma Matlab che implementi l'algoritmo di Simulated Annealing su spazio delle soluzioni discreto

• Nelle stesse condizioni dell'esercizio 1, applicare l'algoritmo SA per trovare un ottimo della funzione energia *E* confrontando diversi criteri di arresto e di decremento della temperature, riportandone anche i tempi di esecuzione.

•Confrontare il massimo trovato con quello dell'algoritmo Hill Climbing

22