Intelligenza Artificiale

Anno Accademico 2022 - 2023

Local Search:

Hill Climbing, Simulated Annealing

Norme di utilizzo dei materiali didattici

La visione e l'utilizzo del presente materiale didattico è riservato agli utenti iscritti al corso al solo fine di studio e approfondimento didattico.

L'utente che accede alla sezione e-learning e che ne consulta i contenuti è tenuto a rispettare le disposizioni legislative che tutelano il diritto d'autore e pertanto è fatto espresso divieto di riprodurre, pubblicare o distribuire i materiali tratti dal presente sito, anche in forma parziale, fatta salva la possibilità di realizzare un'unica copia del materiale, in formato cartaceo e/o digitale, all'esclusivo fine di studio.

L'utente è responsabile per l'uso improprio o non autorizzato del materiale pubblicato effettuato in violazione delle suddette disposizioni.

Sommario

- Introduzione agli Algoritmi di Ricerca Locale
- Hill-Climbing (Steepest Ascent)
- Random-restart Hill-Climbing
- Stochastic Hill-Climbing
- Simulated Annealing

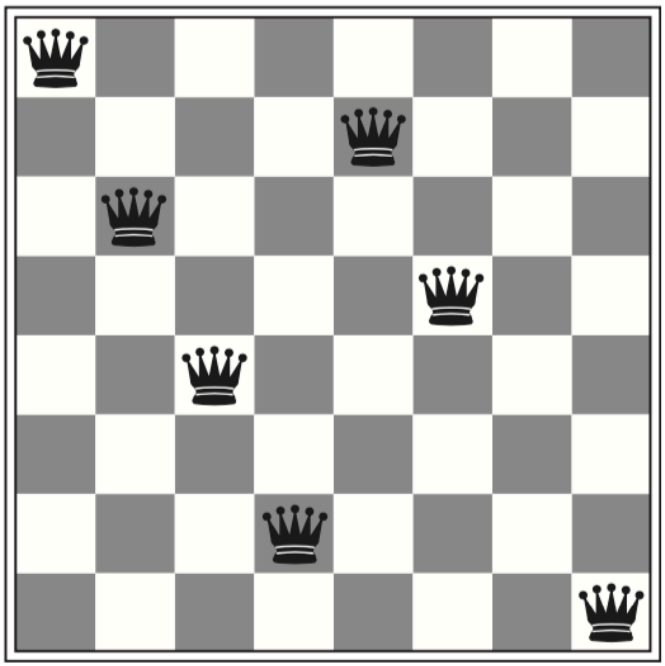
Algoritmi di Ricerca Locale

- Negli problemi visti fino ad ora, quando l'algoritmo risolutivo raggiunge uno stato obiettivo, il cammino verso quello stato costituisce una soluzione al problema.

- Ci sono però problemi in cui lo stato obiettivo contiene tutte le informazioni rilevanti per la soluzione. In tali casi il cammino verso l'obiettivo è irrilevante.

- Ad esempio, nel problema delle 8 regine, ciò che conta è la configurazione finale delle regine e non l'ordine con cui sono state aggiunte.

Esempio:
Problema delle Otto Regine



Esempio:
Problema delle Otto Regine

- Gli algoritmi di ricerca locale usano in genere una formulazione a stato completo:
  - ogni stato ha 8 regine sulla scacchiera
  - una regina per colonna
- La funzione euristica h è il numero di coppie di regine che si attaccano a vicenda, direttamente o indirettamente
- Il minimo globale è zero

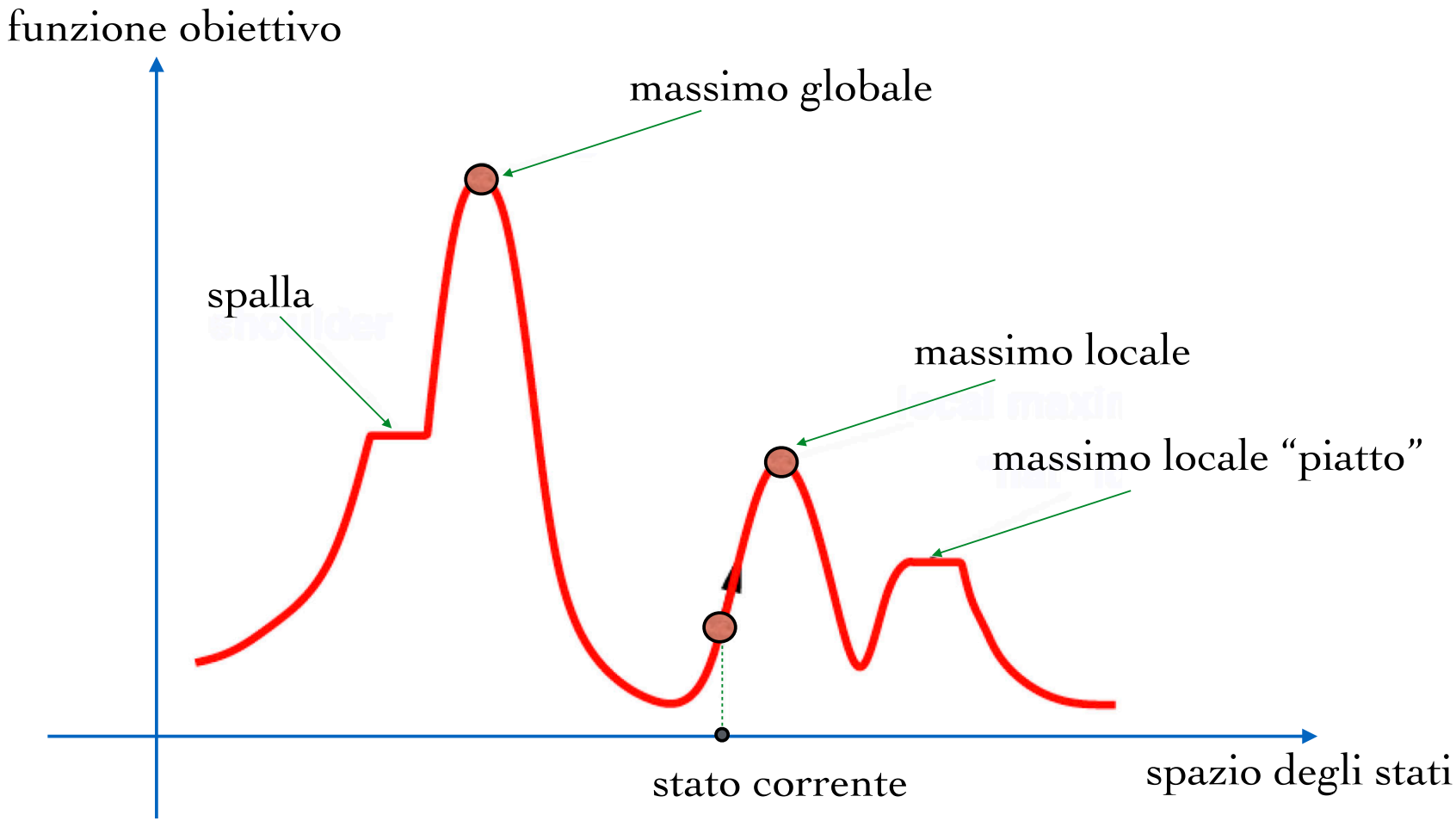
Algoritmi di Ricerca Locale

- Tale classe di problemi è relativa a molte importanti applicazioni, come ad esempio:
  - Progettazione di circuiti integrati.
  - Job-shop scheduling.
  - Ottimizzazione di reti di telecomunicazioni.
  - Instradamento di veicoli.
  - Gestione di portafogli di azioni.

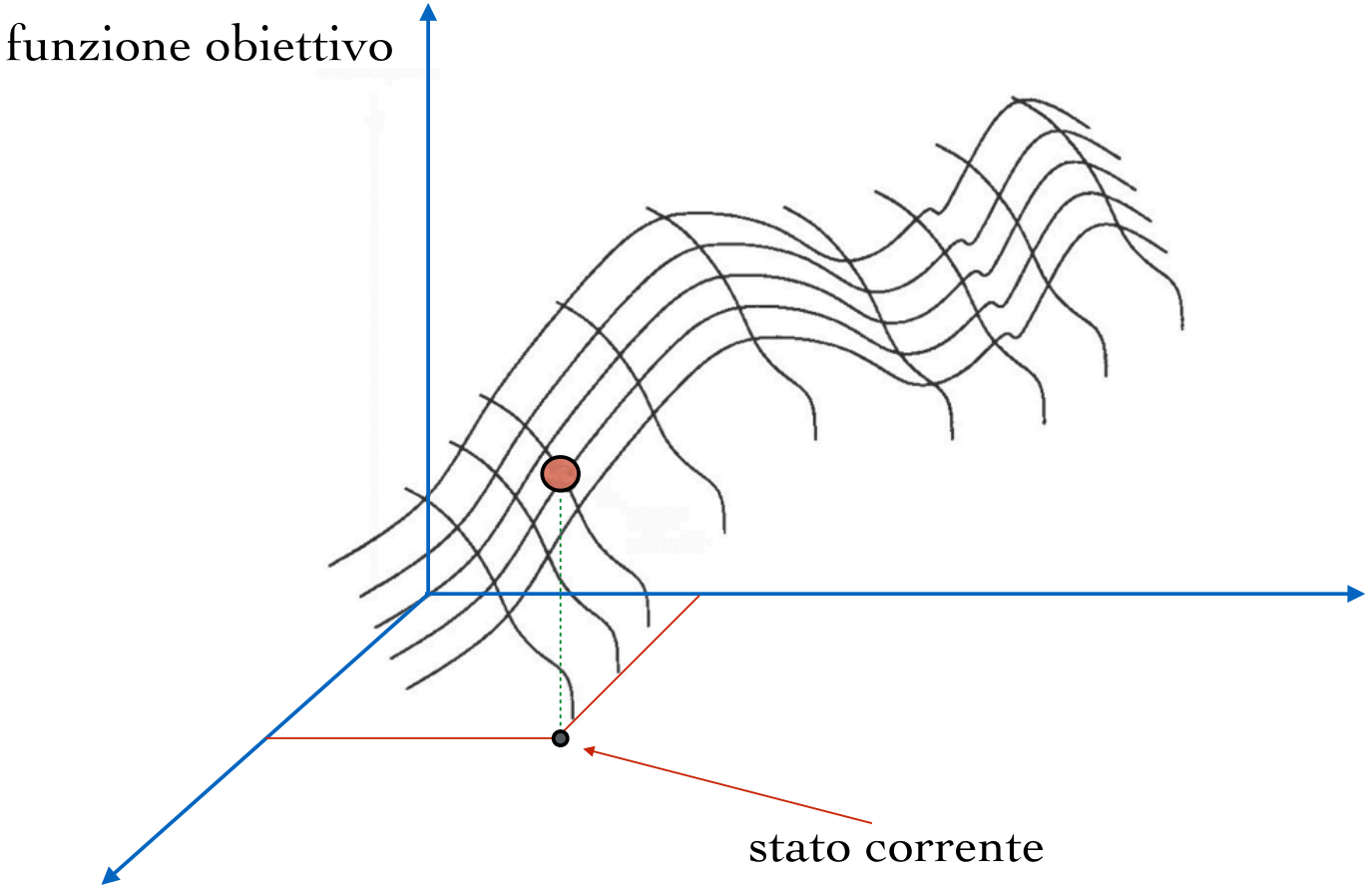
Algoritmi di Ricerca Locale

- Idea intuitiva: immaginare tutti gli stati collocati sulla superficie di un territorio (**panorama dello spazio degli stati** - *state space landscape*).
- L'altezza di un punto nel territorio corrisponde al valore della funzione di valutazione dello stato in quel punto.
- Gli algoritmi di **ricerca locale** (o di **miglioramento iterativo**) si muovono sulla superficie cercando i picchi più alti (le soluzioni ottime).
- Si tiene traccia solo dello stato corrente e si considerano solo gli stati "vicini" (ossia quelli immediatamente adiacenti).

Esempio
(Panorama Spazio degli Stati Monodimensionale)

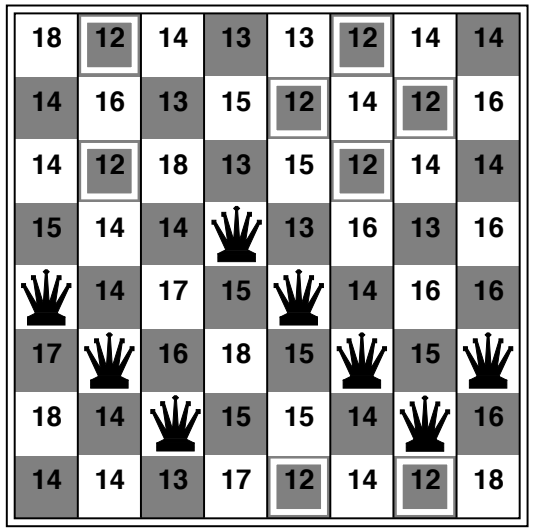


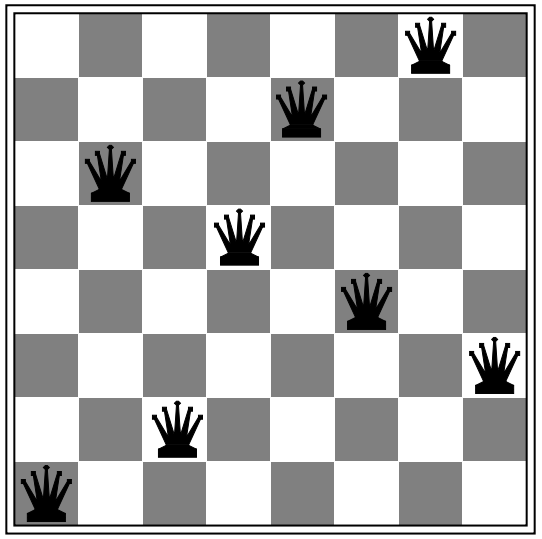
Esempio
(Panorama Spazio degli Stati a due dimensioni)



Esempio:
Problema delle Otto Regine

- Valori della funzione di costo (sinistra) ed esempio di minimo locale (destra)





Algoritmo Hill-Climbing

- Il termine **Hill Climbing**, che letteralmente significa "scalatore di colline", deriva dal fatto che segue sempre le salite più ripide.

- Si tratta di un semplice ciclo che si muove continuamente verso l'alto, cioè nella direzione dei valori crescenti, e termina quando raggiunge un picco che non ha vicini di valore più alto.

Algoritmo Hill-Climbing

Ciclo: spostamento verso stati con valori migliori

Non viene memorizzato l'albero di ricerca

`NODO = <STATO, VALORE>`

Scelta casuale tra i successori con valore massimo.

Algoritmo Hill-Climbing

- **Steepest Ascent** Hill Climbing
- **First-choice** Hill Climbing
- **Random-Restart** Hill Climbing (Iterated Hill Climbing)
- **Stochastic** Hill Climbing

Algoritmo Hill-Climbing
Steepest Ascent

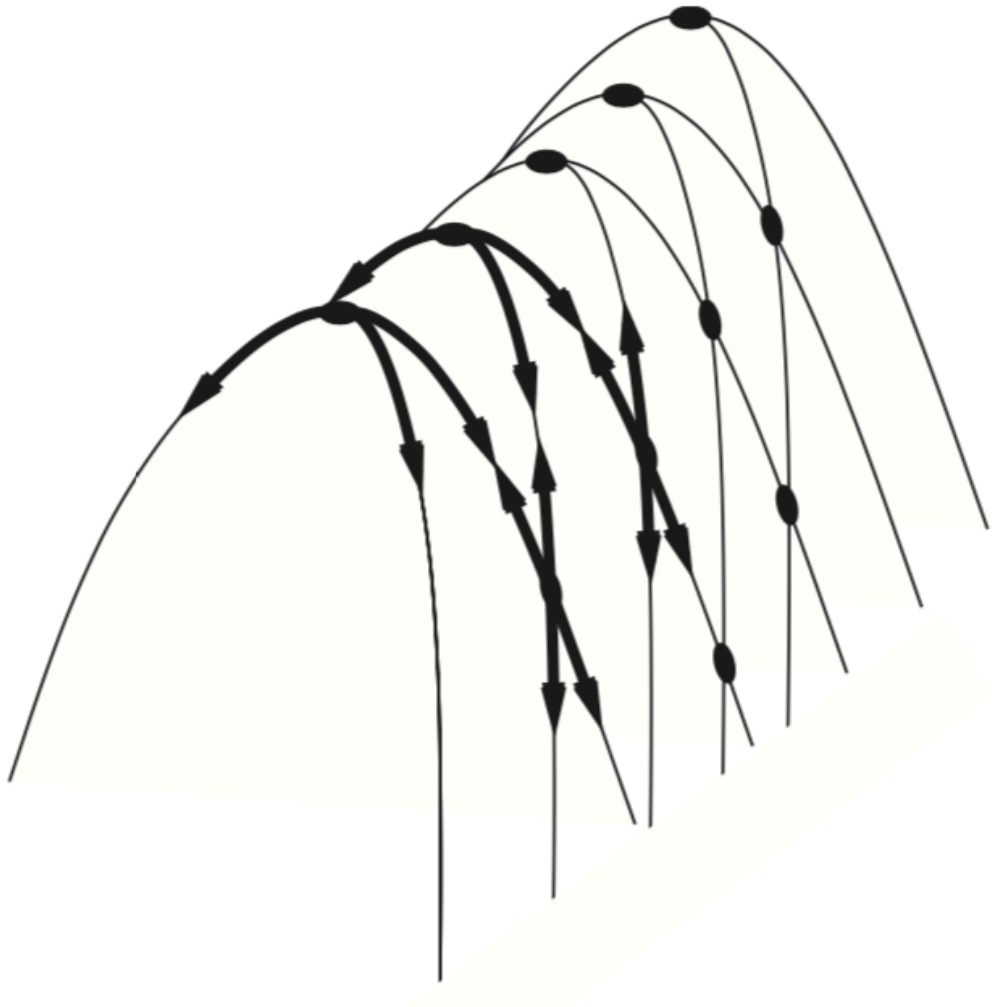
```
function HILL-CLIMBING(problem) returns uno stato che è un
massimo locale
inputs: problem, un problema
local variables: current, un nodo
                 next, un nodo

current ← MAKE-NODE(INITIAL-STATE(problem))
loop do
    next ← successore di current di valore più alto
    if VALUE(next) < VALUE(current) then return STATE(current)
    current ← next
end
```

Algoritmo Hill-Climbing

Inconvenienti:

- Massimi locali $\Rightarrow$ Soluzioni non ottimali
- Plateaux (zone tendenzialmente piatte) $\Rightarrow$ Percorso casuale
- Creste $\Rightarrow$ salita ripida, poi progresso lento

Esempio di creste



Algoritmo Hill-Climbing (1)
Random Restart

- L'algoritmo è in un Hill-Climbing con **riavvio casuale**.
- Esso conduce una serie di ricerche Hill-Climbing partendo da stati iniziali generati casualmente.
- L'algoritmo è completo, con probabilità tendente a 1, per la banale ragione che prima o poi dovrà generare, come stato iniziale, proprio un obiettivo.

Algoritmo Hill-Climbing (2)

Random Restart

```
function RANDOM-RESTART-HILL-CLIMBING(problem) returns uno stato
soluzione
inputs: problem, un problema
local variables:  current, next, best: nodi; local: booleana
t ← 0
Inizializza best
repeat
     local ← false
     seleziona un punto iniziale current random
     repeat
       next ← successore di current con VALUE più alto
       if VALUE(next) > VALUE(current)
         then current ← next
         else local ← true
     until local
     t ← t + 1
     if VALUE(current) > VALUE(best) then best ← current
until t = MAX
return STATE(best)
end
```

Algoritmo Hill-Climbing (3)
Random Restart

Dall'analisi dell'algoritmo notiamo che:

- Il ciclo interno della procedura restituisce sempre un ottimo locale.
- La procedura sfugge dagli ottimi locali solo avviando una nuova ricerca (ciclo esterno) partendo da un nuovo punto scelto casualmente.

Stochastic Hill-Climbing (1)

L'algoritmo Stochastic Hill-Climbing si ottiene modificando la procedura precedente come segue:

- Anziché valutare tutti i punti vicini (neighbourhood) a **current** e selezionare il migliore, selezioniamo casualmente solo un punto **next**.

- Accettiamo **next** con una probabilità che dipende dalla differenza di valutazione tra i due punti:

$$\Delta E = \text{VALUE}(\textbf{current}) - \text{VALUE}(\textbf{next})$$

Stochastic Hill-Climbing (2)

```
function STOCHASTIC-HILL-CLIMBING(problem) returns uno stato soluzione
inputs: problem, un problema
local variables:  current, next, best: nodi
t ← 0
seleziona un punto iniziale current random
Inizializza best
repeat
  next ← seleziona successore di current random
  seleziona next come nodo corrente con probabilità p = 1/(1 + e^(ΔE/T))
  se selezionato next come nuovo current:
    if VALUE(current) > VALUE(best) then best ← current
  t ← t + 1
until t = MAX
return STATE(best)
end
```

Stochastic Hill-Climbing (3)

Osservazioni:

- L'algoritmo ha un solo ciclo.
- Il nuovo punto selezionato è accettato con probabilità p.
- Questo significa che può essere accettato anche un nuovo punto "peggiore" di quello corrente.
- La probabilità dipende dalla differenza tra le valutazioni dei due punti e dal parametro T.
- Il parametro T rimane costante durante l'esecuzione dell'algoritmo.

Stochastic Hill-Climbing (4)

Analizziamo il ruolo del parametro T per il calcolo della probabilità di accettazione del nuovo punto.

Assumiamo ad esempio i seguenti valori:

VALUE(**current**) = 107
VALUE(**next**) = 120

La differenza tra le valutazioni è:

ΔE = VALUE(**current**) - VALUE(**next**) = -13

Probabilità di Accettazione (1)

T	$e^{-13/T}$	p
1	0,000002	1
5	0,0743	0,93
10	0,2725	0,78
20	0,52	0,66
50	0,77	0,56
10^{10}	0,9999..	0,5..

Probabilità di Accettazione (2)

- Dall'analisi della tabella risulta chiaro che al crescere di T diventa sempre meno importante la differenza delle valutazioni dei due punti ai fini dell'accettazione del nuovo punto.

- Se invece T è molto piccolo (e.g., T = 1) la procedura diventa simile ad un semplice hill-climber.

- Vediamo ora un esempio del come possa variare la probabilità di accettazione al variare di VALUE(**next**) supponendo T=10 per una certa esecuzione dell'algoritmo (supponiamo ancora che VALUE(**current**)=107).

Probabilità di Accettazione (3)

VALUE(next)	ΔE	$e^{\Delta E/10}$	p
80	27	14,88	0,06
100	7	2,01	0,33
107	0	1	0,5
120	-13	0,27	0,78
150	-43	0,01	0,99

Probabilità di Accettazione (4)

- Se il nuovo punto ha la stessa valutazione del punto corrente, la probabilità di accettazione è del 50%.
- Se il nuovo punto ha una valutazione migliore del punto corrente, la probabilità di accettazione è maggiore del 50%.
- La probabilità di accettazione cresce con la differenza (negativa) tra le due valutazioni. Se il nuovo punto ha una valutazione nettamente migliore del punto corrente (VALUE(**next**) = 150), la probabilità si avvicina al 100%.

Link Function (1)

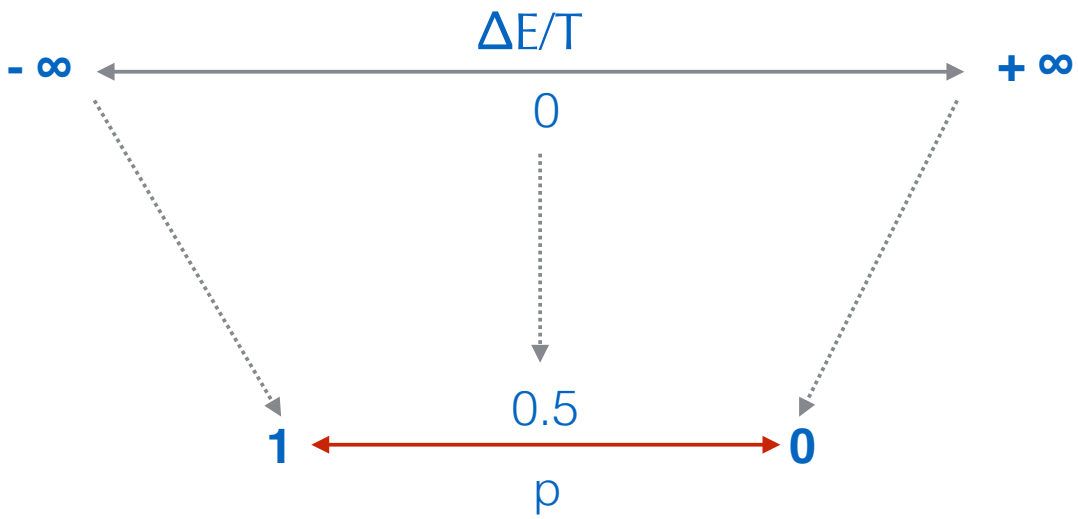
- Il problema risolto nell'algoritmo è relativo al passaggio dai valori di ΔE/T a quelli delle probabilità.

- La funzione ΔE/T ha un range che va da -∞ a +∞:

- La probabilità, come sappiamo, può variare da 0 a 1:

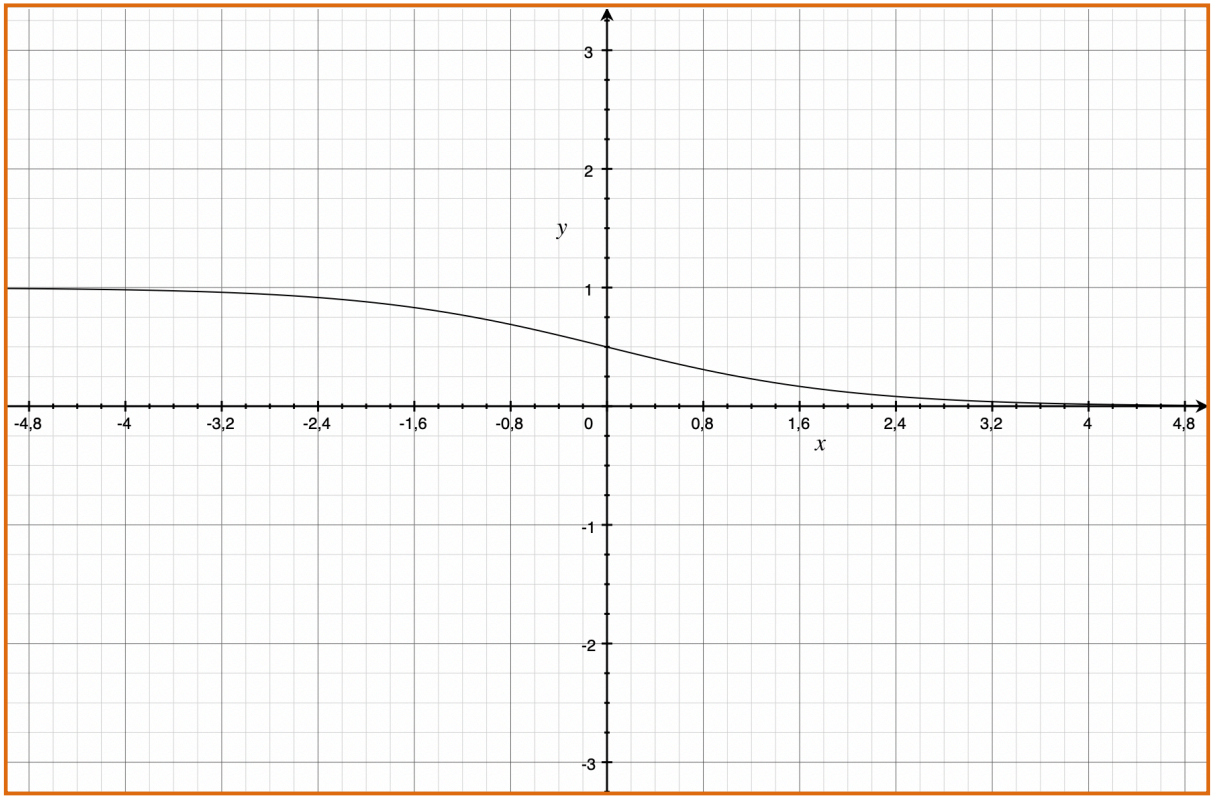
Link Function (2)

- L'espressione usata nell'algoritmo per la probabilità è in buona sostanza una "link function" che realizza un mapping tra i due intervalli:



Link Function (2)

- La figura mostra la funzione $y=\frac{1}{1+e^x}$ che, come si vede, ha l'insieme di definizione costituito dall'intervallo $(-\infty, +\infty)$ e come codominio l'intervallo [0, 1].



Simulated Annealing (1)

- L'algoritmo di Simulated Annealing (dall'inglese Tempra Simulata) prende il nome dall'analogia con il processo di ricottura in metallurgia che consiste nel ridurre gradualmente la temperatura di un materiale in modo da permettere alla sua struttura cristallina di raggiungere uno stato ad energia minima.

- E' un algoritmo con miglioramenti iterativi che consente di sfuggire da ottimi locali durante la fase di search.

Simulated Annealing (2)

- La differenza principale tra lo stochastic hill-climber e il simulated annealing è che quest'ultimo modifica il parametro T durante l'esecuzione.

- Il simulated annealing comincia con valori alti di T comportandosi in modo simile ad una pura random search. Poi decrementa gradualmente il valore di T. Verso la fine dell'esecuzione i valori di T sono molto piccoli e perciò l'algoritmo si comporta in modo simile ad un normale hill-climber.

- Inoltre l'algoritmo accetta sempre nuovi punti se essi hanno una valutazione migliore del punto corrente.

Simulated Annealing (3)

```
function SIMULATED-ANNEALING(problem) returns uno stato soluzione
inputs: problem, un problema
local variables: current, next: nodi;
t ← 0
Inizializza T
seleziona un punto iniziale current random
Inizializza best
repeat
     repeat
       next ← successore di current scelto random
       if VALUE(next) > VALUE(current)
         then current ← next
              if VALUE(current) > VALUE(best) then best ← current
         else if random[0, 1) < e^(-ΔE/T) then current ← next
     until (termination-condition)
     T = g(T, t)
     t ← t + 1
until (halting-criterion)
return STATE(best)
end
```

Simulated Annealing (4)

Sistema Fisico	Problema di ottimizzazione
Stato	Soluzione accettabile
Energia	Funzione di valutazione
Stato “ground”	Soluzione ottimale
Raffreddamento rapido	Local search
Temperatura	Parametro di controllo T
“careful annealing”	Simulated annealing

Simulated Annealing (5)

Come per gli altri algoritmi di ricerca di questo tipo, il simulated annealing richiede che si determinino le risposte ai seguenti quesiti relativi allo specifico problema:

- Come definiamo una soluzione al problema?
- Quali sono i "vicini immediati" di una soluzione?
- Qual è il costo della soluzione?
- Come determiniamo la "soluzione iniziale"?

Simulated Annealing (6)

- Le risposte alle precedenti domande consentono di produrre la struttura dello spazio di ricerca, di definire il "vicinato" (neighborhood), la funzione di valutazione, il punto di partenza.

- Inoltre, il simulated annealing richiede che si determinino i seguenti parametri addizionali:

  - la "temperatura" iniziale T
  - il rapporto di raffreddamento g(T,t)
  - la condizione di terminazione
  - il criterio di arresto (halting-criterion)

Simulated Annealing (7)

Molte implementazioni dell'algoritmo seguono la semplice sequenza di passi:

STEP 1: $T \leftarrow T_{max}$
select **current** at random

STEP 2: pick a point **next** from the neighborhood of **current**
if VALUE(**next**) is better than VALUE(**current**) then select it
(**current** ← **next**)
else select it with probability $e^{-\Delta E/T}$
repeat this step k_T times

STEP 3: set $T \leftarrow r\ T$
if $T \geq T_{min}$ then goto **STEP 2**
else goto **STEP 1**

Simulated Annealing (8)

Aree di applicazione dell'algoritmo:

- Computer-aided design
- integrated circuits
- Image processing
- Code design
- Neural network theory
- Numerical analysis
- Biology
- Magnetics
- Materials science
- Game theory
- Wheel balancing
- ecc.

Sintesi degli Argomenti Trattati nella Lezione

- Gli algoritmi di **Ricerca Locale** (o con **miglioramenti iterativi**) lavorano su formulazioni del problema a stato completo. Si applicano per problemi in cui lo stato obiettivo contiene tutte le informazioni rilevanti per la soluzione, dove quindi il cammino verso l'obiettivo è irrilevante. Essi tengono in memoria solo un singolo stato, ma possono rimanere bloccati su massimi locali.

- L'algoritmo **Hill-Climbing** si muove continuamente verso l'alto, ossia verso valori crescenti, e termina quando raggiunge un picco che non ha vicini di valore più alto.

- L'algoritmo **Iterative Hill-Climbing** è un Hill-Climbing con riavvio casuale, ossia effettua una serie di ricerche hill-climbing partendo da stati iniziali generati casualmente. E' completo con probabilità tendente a 1 perché prima o poi dovrà generare, come stato iniziale, proprio un obiettivo. Spesso è possibile trovare un massimo locale ragionevolmente buono con un numero abbastanza piccolo di riavvii.

- Lo **Stochastic Hill-Climbing** sceglie la mossa da effettuare con una probabilità che dipende dalla "pendenza" (ΔE) delle mosse, oltre che da un parametro fisso T.

- L'algoritmo **Simulated Annealing** fornisce un modo per evitare ottimi locali scegliendo casualmente una mossa, accettando subito il nuovo stato se è migliore dello stato corrente, oppure accettandolo con probabilità p se è peggiore. La probabilità dipende da ΔE e dal parametro T. Il parametro T varia nel tempo. L'algoritmo fornisce soluzioni ottime quando viene impostata una "velocità di raffreddamento" appropriata. Le aree di applicazione del Simulated Annealing sono numerosissime.

Riferimenti

Russell, S., Norvig, P. *Artificial Intelligence - a Modern Approach*, fourth Edition, Pearson, 2021.

Michalewicz, Z. E Fogel, D.B. *How to Solve It: Modern Heuristics*, Springer, 2010.

Kirkpatrick, S., Gelatt Jr., C.D. e Vecchi, M.P. , N. "Optimization by Simulated Annealing", *Science* **220**, 1983, pp. 671-680.

Cerny, V. "Thermodinamical Approach to the Traveling Salesman Problem", *J. Opt. Theory Appl.* **45**, 1985, pp. 41-51.

Van Laarhoven, P.J.M. e Aarts, E.H.L. *Simulated Annealing: Theory and Applications*, Kluwer, 1987.