Algoritmi di miglioramento iterativo

In molti problemi di ottimizzazione, *il cammino* è irrilevante;
la soluzione è costituita dallo stato goal stesso

Quindi lo spazio degli stati è dato dall'insieme delle configurazioni "complete";

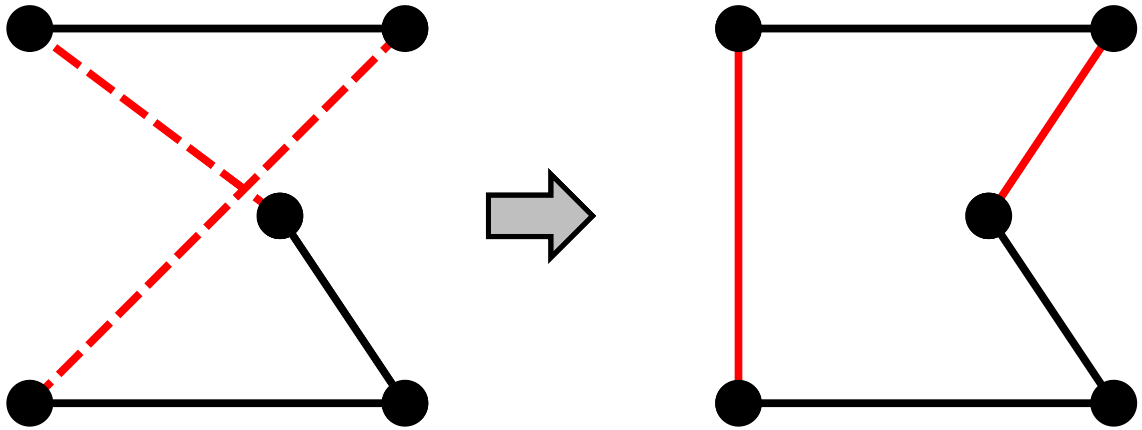
trovare una configurazione *ottima*, es.: TSP
o, trovare una configurazione che soddisfi dei vincoli, es.: orario

In tali casi, si possono usare gli algoritmi di *miglioramento iterativo*;
Mantengono un singolo stato "corrente", e tentano di migliorarlo

Impiegano spazio costante, e sono quindi adatti sia per ricerca online che per ricerca offline

Esempio: Problema del commesso viaggiatore

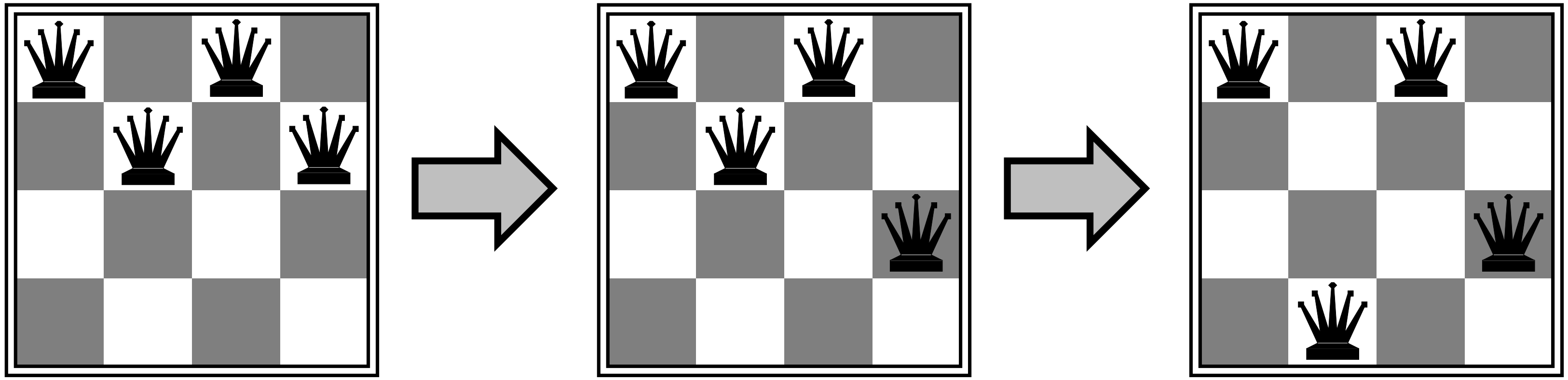
Si parte con un percorso qualsiasi, e si eseguono scambi a coppie



Esempio: n-regine

Disporre n regine su una scacchiera $n \times n$ senza che si minaccino
(non ci devono essere due regine sulla stessa riga, colonna, o diagonale)

Muovere una regina in modo da minimizzare il numero di minacce



Hill-climbing (o discesa/ascesa di gradiente)

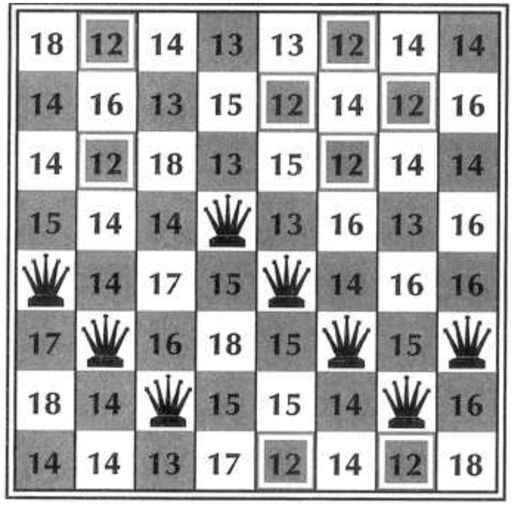
```
function HILL-CLIMBING(problema) returns uno stato che è un massimo locale
    inputs: problema, un problema
    variabili locali: nodo_corrente, un nodo
                      vicino, un nodo

    nodo_corrente ← CREA-NODO(STATO-INIZIALE[problema])
    loop do
        vicino ← il successore di nodo_corrente di valore più alto
        if VALORE[vicino] ≤ VALORE[nodo_corrente] then return STATO[nodo_corrente]
        nodo_corrente ← vicino
```

Hill-climbing: esempio delle 8 regine

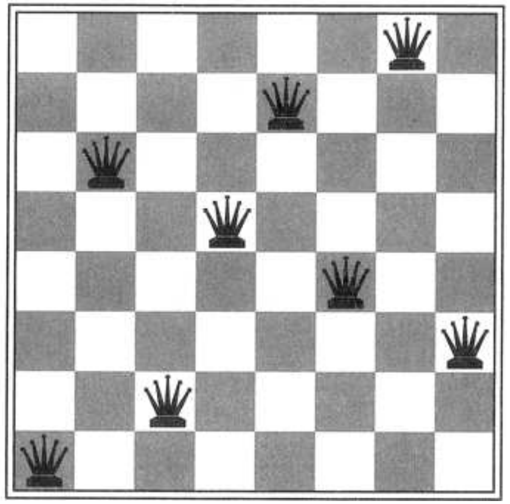
$h(s) =$ *numero di coppie di regine che si attaccano a vicenda*

Quanti successori per ogni stato ? 8×7

($\#$ regine $\times$ caselle libere su colonna/riga [deve esserci 1 regina per colonna/riga])



Uno stato con $h = 17$ e valori di h per ogni successore indicati

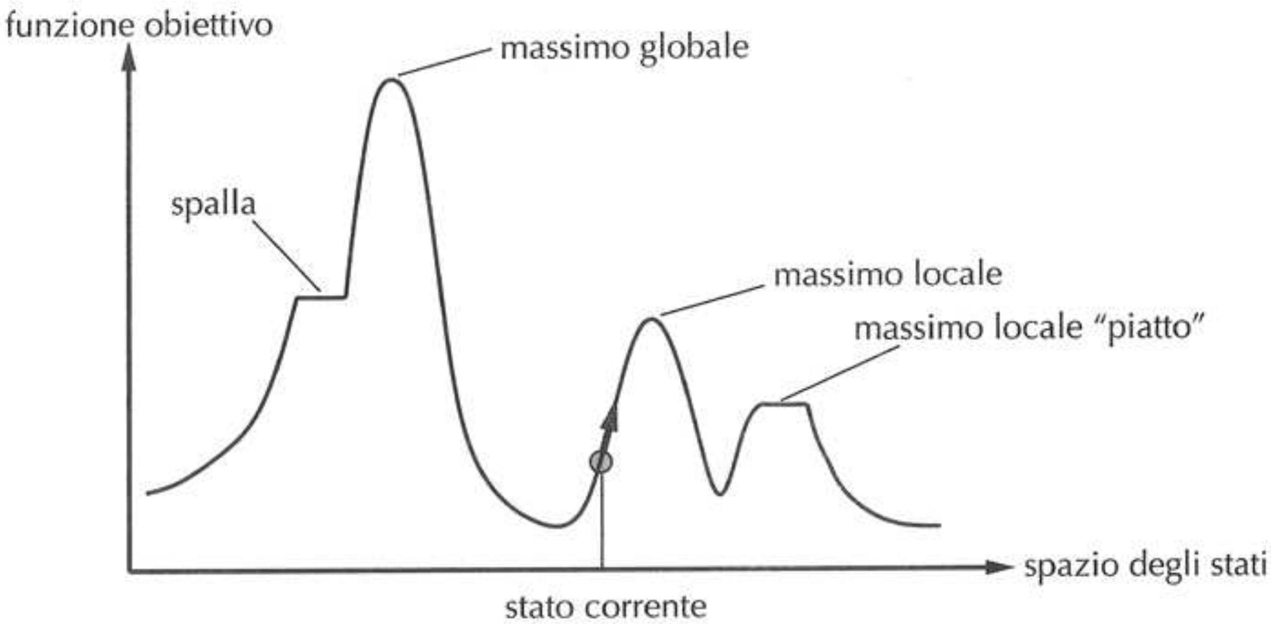


Un minimo locale con $h = 1$. Ogni successore di questo stato ha $h > 1$.

Lo stato a destra "dista" solo 5 passi da quello a sinistra

Hill-climbing: massimi locali ed altri problemi

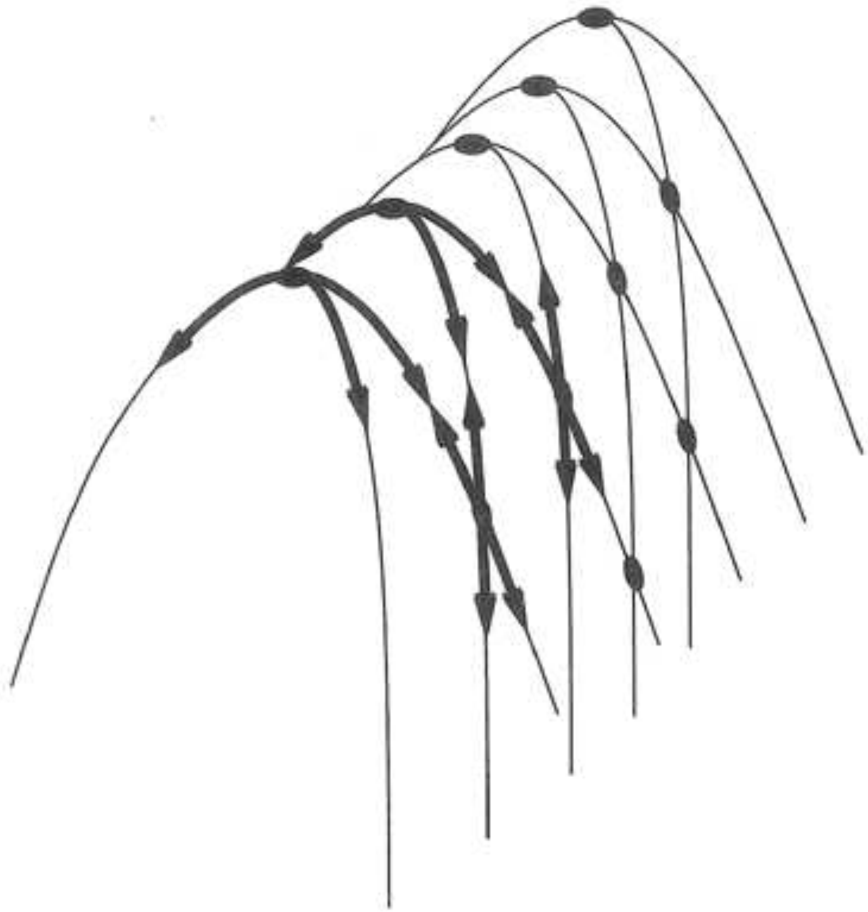
Problemi: a seconda dello stato iniziale, può fermarsi su dei massimi locali e/o "spalle"



Nel caso di spazi continui è difficile scegliere la dimensione del passo di progressione ed inoltre si può avere una convergenza molto lenta

Hill-climbing: cresta (*ridge*)

Sequenza di massimi locali (a valore crescente muovendosi verso l”’interno” della figura) molto difficili da esplorare per Hill-climbing



Hill-climbing: alcune soluzioni

Possibili soluzioni:

- plateau (h piatta): "mossa laterale", cioè ci si sposta in uno stato con identico valore di h
  - bisogna stare attenti ad evitare cicli, specialmente nel caso di massimi (minimi) locali piatti
  - tipica soluzione: porre un limite massimo al numero consecutivo di mosse laterali
- massimi (minimi) locali: eseguire scelte stocastiche e/o più ricerche da stati iniziali diversi
  - Hill-climbing stocastico: scegliere a caso fra tutte le mosse che migliorano h, eventualmente usando una probabilità di selezione che è proporzionale al miglioramento (convergenza + lenta, ma spesso soluzioni migliori)
  - Hill-climbing con riavvio casuale: esegue più ricerche a partire da stati iniziali diversi (scelti a caso). Se p è la probabilità di trovare una soluzione ottima per una singola ricerca, il numero di ricerche atteso prima di trovare una soluzione ottima è $1/p$

Hill-climbing e le 8 regine

Numero stati: 8^8 (circa 17 milioni)

- Hill-climbing standard
  - soluzione (ottima) trovata il 14% delle volte
  - in media circa 4 passi per trovare una soluzione, altrimenti circa 3 passi in caso di soluzione subottima
- Hill-climbing con mosse lateriali (non più di 100 consecutive)
  - soluzione (ottima) trovata il 94% delle volte
  - in media circa 21 passi per trovare una soluzione, altrimenti circa 64 passi in caso di soluzione subottima
- Hill-climbing con riavvio casuale
  - soluzione (ottima) trovata con probabilità $p = 0,14$, quindi circa 7 ricerche per trovare una soluzione ottima ($(1-p)/p = 6,14$ fallimenti $+ 1$ successo). Numero di passi complessivo atteso: $3(1-p)/p + 4 = 22,43$
  - con mosse laterali: soluzione (ottima) trovata con probabilità $p = 0,94$, quindi circa $1,06$ ricerche per trovare una soluzione ottima ($(1-p)/p = 0,06$ fallimenti $+ 1$ successo). Numero di passi complessivo atteso: $0,06(1-p)/p + 21 = 25,08$

Simulated annealing

Idea: evitare i massimi locali permettendo delle mosse "cattive"
ma gradualmente decrementare la loro grandezza e frequenza

```
function SIMULATED-ANNEALING(problema, raffreddamento) returns uno stato soluzione
    inputs: problema, un problema
            velocità_raffreddamento, una corrispondenza dal tempo alla "temperatura"
    variabili locali: nodo_corrente, un nodo
                      successivo, un nodo
                      T, una "temperatura" che controlla la probabilità
                          di compiere passi verso il basso

    nodo_corrente ← CREA-NODO(STATO-INIZIALE[problema])
    for t ← 1 to ∞ do
        T ← velocità_raffreddamento[t]
        if T = 0 then return nodo_corrente
        successivo ← un successore scelto a caso di nodo_corrente
        ΔE ← VALORE[successivo] – VALORE[nodo_corrente]
        if ΔE > 0 then nodo_corrente ← successivo
        else nodo_corrente ← successivo solo con probabilità e^(ΔE/T)
```

Proprietà del simulated annealing

A "temperatura" fissata T, la probabilità di occupazione degli stati raggiunge la distribuzione di Boltzmann

$$p(x) = \alpha e^{\frac{E(x)}{kT}}$$

T diminuito abbastanza lentamente $\Longrightarrow$ si raggiunge sempre lo stato migliore (Metropolis et al., 1953, per problemi di modellazione di processi fisici)

Ampiamente usato in applicazioni pratiche, come la progettazione di circuiti VLSI e la definizione degli orari dei voli delle linee aeree