Contents

1 Beam search

ricerca a fascio[1] hai una frontiera limitata, la coda con priorità invece di creschere all'infinito, ha un limite massimo di dimensione, ti tieni solo i 100 più fighi nella coda, gli altri si fottano si perde l'ottimalità, ci sono anche casi patologici in cui perdi la completezza, ma nelle applicazioni pratiche di solito non succede

2 Ricerca locale

è piuttosto diversa, anche come formulazione, dal cazzo che me frega

2.1 Problema delle 8 regine

abeums scacchiera, dobbiamo metterci 8 regine, nessuna regina deve poter attaccare un'altra regina, non in orizzontale, non in verticale, non in diagonale

questo sarebbe formulabile come problema di ricerca, ma come cazzo me lo faccio con una sequenza di azioni, degli stati in mezzo, come cazzo mi definisco le azioni... che cazzo di senso ha? nessuno

(le applicazioni effettive di sti problemi è ad esempio, la disposizione di componenti in un circuito, quelli per le consegne di amazon, problemi di logistica, et al..., problemi grossi, dove piccoli miglioramenti algorigmici vogliono dire SOLDI)

[1] si ricerca e si produce, per la figa e per il duce

2.2 Algoritmi di local search

2.2.1 Hill climbing

memoria $O(1)$, si chiama locale perchè sta in uno stato solo, non ha una frontiera, si guarda intorno, vede i suoi vicini, vede quale vicino è meglio, e ci si va

> nota di **gradient ascent** vado avanti prendendo ogni volta come passo il gradiente (la derivata, magari a più dimensioni)

```
def hill_climbing(problem):
    current = problem.initial
    while True:
        best = best_successor(current, value_function=value)
        if value(best) <= value(current):
            # se siamo in un massimo locale
            # non ci sono miglioramenti possibili
            return current
        current = best

def best_succesor(current, value_function):
    best = None
    best_value = 0
    for n in successors:
        if n.value >= best_value:
            best = n
            best_value = n.value
    return best
```

questo si pianta negli ottimi locali o negli altipiani (plateau) se non si trova soluzione, o se ci si becca a cazzo in un massimo locale stupido, si ricomincia da uno stato alla cazzo, noto anche come **random restarts**

2.2.2 Local beam search

fasci locali, AUTARCHIA

questo e beam search sono due algoritmi parecchi parecchio diversi, local beam search alloca un pool di ricercatori in posizione diverse questo è un primordio di cosiddetti **population methods**, metodi basati su poplazioni di roba, si veda ad esempio gli algoritmo genetici

la differenza tra 4 agenti e 4 restart è che li agenti possono parlare tra di loro si impara roba sulla struttura del problema da stati passati

si fanno gli argmax dell'unione di tutti i vicinati, parlano tra di loro perchè **we are the borg** quello con i vicini più brutti si ammazza, quelli con i vicini più fighi vanno avanti, Darwin approva.

in questi problemi in genere l'ottimalità non si garantisce quasi mai
se finite su un'ottimo locale

2.2.3 Simulated annealing

(tranquilli se non lo capite tanto non lo caga nessuno)

si fa raffreddare piano piano piano piano piano piano... si vuole uno stato della materia il più ordinato possibile

si cambia il codice

```
import math.exp

def simulated_annealing(problem):
    current = problem.initial
    for i in range(1, ):
        temp = schedule(i)
        if temp <= qualche_soglia:
            return
        proposta = genera_alla_cazzo_di_cane(current)
        delta = proposta.value() - current.value()
        if delta > 0: # if proposta.value() > current.value()
            current = proposta
        else:
            # ricorda che qui delta <= 0
            p = exp(delta/temp)
            # a basse temperature non si accetta niente, e^(-roba/0) =
            ↪  e^(-infinito) = 0
            # ad alte temperature si accetta un po' di più,
            ↪  e^(-roba/tanto) = e^(-0) = 1
            # per ammazzare matematici
            if rand(0,1) <= p:
                current = proposta

def schedule(ind):
    boh()
    qualcosa_di_esponenziale()
    bla_bla_bla(boltzmann)
```

inizia andando a cazzo, random walk, poi gli passa la sbronza e cammina in maniera più normale verso un goal