UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Úvod do Umelej Inteligencie
Cvičenie 3 - Lokálne prehľadávanie: hill-climbing

Október 5, 2022

OPTIMALIZAČNÝ PROBLÉM

Typ úlohy, ktorý dnes budeme riešiť, sa nazýva "optimalizačný problém". Máme nájsť maximum/minimum funkcie, ktorú však nepoznáme, a vieme ju používať iba ako black-box - dáme jej vstup a ona vráti výsledok (číslo). Vstup môže byť ľubovoľná sada parametrov, výstupom je vždy jedno číslo. Našou úlohou je nájsť taký vstup, ktorý vráti čo najväčšie (maximalizácia) alebo najmenšie (minimalizácia) číslo.
Pokiaľ chceme nejakú konkrétnu úlohu (napr. "8 kráľovien") formulovať ako optimalizačný problém, potrebujeme pre ňu definovať tri základné funkcie:

- *random_state()*: vygeneruje náhodný stav (hodnoty vstupných parametrov), v ktorom môžeme začať algoritmus.
- *neighbors(x)*: vráti "susedné" stavy k stavu x - také, ktoré sa od neho veľmi nelíšia.
- *fitness(x)*: ohodnotí stav x - toto je tá funkcia, ktorú reálne optimalizujeme (maximalizujeme/minimalizujeme).

HILL-CLIMBING

Najjednoduchší optimalizačný algoritmus - nájdem najlepší susedný stav a pohnem sa tam, pričom ak žiadny sused nie je lepší ako aktuálny stav, tak som na vrchole a skončím:

Algorithm 1 - hill-climbing algoritmus

```
1: procedure HILL_CLIMB()
2:     x = random_state()
3:     while True do
4:         best_neighbor = best neighbouring state of x
5:         if fitness(best_neighbor) ≤ fitness(x) then
6:             return x
7:         x = best_neighbor
```

Program:

Pripravený program obsahuje tri triedy: *OptimizeMax* a z nej odvodené *MysteryFunction* a *EightQueens*.

Abstraktná trieda *OptimizeMax* je postavená tak, aby riešila ľubovoľný maximalizačný problém. Na riešenie (optimalizáciu) sa môže použiť metóda hillclimb, ktorú budete mať za úlohu doprogramovať:

- *hillclimb(x, max_steps)* - z počiatočného stavu *x* spustíme hill-climbing algoritmus, pričom ho limitujeme na max_steps krokov/iterácií.

V prípade MysteryFunction v algoritme zavolajte *self.plot(x, self.fitness(x))* v každej iterácii (t.j. pre každú novú hodnotu *x*) - na grafe potom uvidíte vývoj optimalizácie.
Všetky ďalšie funkcie sa neimplementujú v tejto (abstraktnej) triede *OptimizeMax*, ale až v odvodených. Každá odvodená trieda reprezentuje jeden konkrétny maximalizačný problém, sú pre ňu definované funkcie:

- *fitness(x)* - ohodnotenie stavu *x* - čím väčšia fitness, tým lepší stav.
- *neighbors(x)* - vráti zoznam (list) susedných stavov pre stav *x*.
- *random_state()* - vráti náhodný stav, v ktorom môžeme začať lokálne prehľadávanie.

Budeme riešiť dva optimalizačné problémy:

- Hľadanie maxima funkcie, ktorej predpis (akože) nepoznáme (trieda *MysteryFunction*). Táto funkcia má globálne maximum v nule, no obsahuje aj viacero lokálnych maxím. Pri hill-climbingu zrejme skončíme väčšinou v lokálnom maxime (závisí od náhodnej inicializácie).
- Problém ôsmich dám na šachovnici (trieda *EightQueens*). Úlohou je umiestniť 8 dám na šachovnicu tak, aby sa navzájom neohrozovali. Pri tomto probléme je potrebné zvoliť si vhodnú reprezentáciu stavu - napr. list dvojíc (x,y), avšak sú aj pohodlnejšie možnosti. Netriviálnou úlohou je tiež naprogramovať schopnú fitness funkciu. V kóde krátko okomentujte svoju voľbu reprezentácie stavu. Pri tomto probléme sa nevykresľuje nič (čiže nepoužite self.plot(...)), stačí iba vrátiť výsledok po aplikovaní hill-climbingu.

Úloha 1 (0.25b): Doprogramujte funkciu *hillclimb(x, max_steps)* tak, aby vedela nájsť (lokálne) maximum.
Úloha 2 (0.75b): Do triedy *EightQueens* doprogramujte funkcie *fitness(x)*, *neighbors(x)* a *random_state()*, ktoré budú popisovať problém ôsmich kráľovien na šachovnici. Vaša implementácia hill-climb by mala vedieť nájsť riešenie (nanajvýš po cca 10 spusteniach, pri takejto jednoduchej metóde ako je hillclimb sa to často môže zaseknúť v lokálnom extréme). **V kóde jednoznačte napíšte, aká hodnota fitness zodpovedá vyriešenej úlohe 8-queens**.