
Úvod do Umelej Inteligencie

Cvičenie 4 - Genetický Algoritmus: výber rodičov, kríženie a mutácia

Október 12, 2022

GENETICKÝ ALGORITMUS

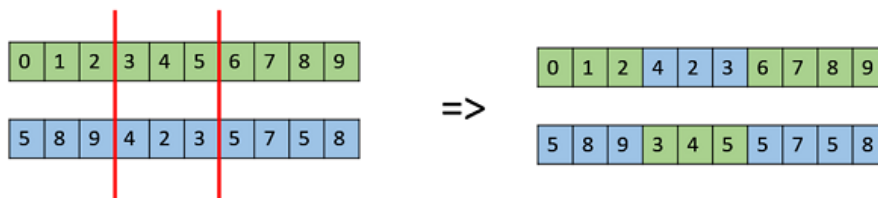
HLAVNÉ KOMPONENTY GENETICKÉHO ALGORITMU:

GA je iteračný algoritmus lokálneho prehľadávania, t.j. každú iteráciu = generáciu sa posunie bližšie k riešeniu. Počas každej generácie sa dejú nasledovné hlavné kroky:

- Ohodnotenie každého jedinca z populácie fitness funkciou.
- Výber rodičov - dvojíc, ktoré sa budú krížiť. Každý pár rodičov krížením vygeneruje (zvyčajne dve) deti.
- Mutácia - vybraní jedinci sa s malou pravdepodobnosťou náhodne zmenia.
- Výber kto postúpi do ďalšej generácie: kombinácia pôvodnej populácie (rodičov) a detí.

Každý z týchto krokov sa dá realizovať veľa rôznymi spôsobmi. My budeme implementovať nasledovné:

- Ohodnotíme si jedincov.
- Skupinu rodičov vyberieme ako najlepších $N/2$ jedincov. Zo skupiny rodičov môžeme dvojice na kríženie vyberať náhodne, alebo v určitom poradí.
- Z každej dvojice rodičov vznikne dvojica detí pomocou viacnásobného kríženia (k -point crossover) - sekvencie dvoch rodičov sa prekrížia na **práve** k náhodných miestach, čím vzniknú dve deti. Pre $k = 2$:



Deti prejdú mutáciou - každá zložka daného jedinca sa zmení s malou pravdepodobnosťou p . Implementujte dva druhy mutácie:

- Bitová mutácia: ak je jedinec reťazec bitov (napr. [0, 1, 1, 0, 1]), tak mutácia znamená obrátenie bitu z 0 na 1 alebo naopak.

- Číselná mutácia: ak je jedinec reťazec čísel (napr. [0.3, 1.0, 0.0, 0.8, 0.6]), tak mutácia znamená pripočítanie náhodného čísla z $\mathcal{N}(0, \sigma^2)$ (normálneho rozdelenia).

Nová generácia vznikne z:

- Najlepších $N/2$ jedincov z pôvodnej populácie (t.j. rodičov).
- $N/2$ detí, ktoré prešli mutáciou.

Program:

V *problems.py* máte zadefinované tri úlohy (problémy), ktoré budeme riešiť GA: reťazec jednotiek, kreslenie smajlíka a kreslenie umeleckého diela. Tieto tri triedy dedia od *GenAlgProblem*, a majú implementované funkcie *fitness(x)* a *mutation(x)*.

V *genetic.py* máte hlavnú triedu *GenAlgProblem*, do ktorej dorobte:

- *crossover(x, y, k)* (0.2b) - zoberie dvoch rodičov x a y , vykoná k -point crossover, a vráti výsledné dva reťazce - deti.
- *boolean_mutation(x, prob)* (0.15b) - každý bit jedince x zmení s pravdepodobnosťou $prob$.
- *number_mutation(x, prob)* (0.15b) - každé číslo jedince x s pravdepodobnosťou $prob$ zmení tak, že k nemu pripočíta náhodné číslo.
- *solve(max_generations, goal_fitness)* (0.5b) - spustí samotný genetický algoritmus, využijúc vyššie uvedené funkcie.

Algoritmus sa zastaví, keď sa presiahne počet *max_generations* generácií, alebo keď získame jedince s fitness rovnou aspoň *goal_fitness*.

Pri implementovaní GA môžete predpokladať, že veľkosť populácie je deliteľná 4.

Zopár detailov:

1. Populácia *self.population* je list jedincov, každý jedinec je list čísel.
2. V triede *GenAlgProblem* máte premenné *self.n_crossover* a *self.mutation_prob*, ktoré v *solve()* použijete namiesto konštánt.
3. Funkcia *solve()* má vrátiť najlepšieho nájdeneho jedince.
4. Generovanie iniciálnej populácie máte pripravené.
5. Na testovanie funkcie *crossover(x, y, k)* máte pripravený kúsok kódu na konci *genetic.py*.

Úloha (1b): Implementujte spomínané funkcie tak, aby riešili všetky tri problémy. Pri treťom probléme (kreslenie umeleckého diela) výsledný fitness by mal byť aspoň okolo 0.75. Počas debugovania/testovania sa môžete hrať s parametrami a pozorovať, ako vplyvajú na úspešnosť algoritmu (veľkosť populácie, počet prekrížení pri k -point crossover, pravdepodobnosť mutácie, sigma pri číselnej mutácii).

Bonus (0.5b): Implementujte dve pokročilejšie metódy na výber rodičov: ruleta a turnaj (dá sa o tom nájsť veľa literatúry, napríklad TU). Do kódu alebo *readme.txt* stručne opíšte, čo si o všetkých implementovaných metódach myslíte, ktorá je najlepšia, prípadne kedy sa ktorú oplatí použiť.