14/11/2021

Strojové učenie sa

Umelá inteligencia – zadanie 3

Samuel Hetteš, ID: 110968
STU FIIT 2021/2022

OBSAH

ZADANIE Č. 3C

Úloha

Obchodný cestujúci má navštíviť viacero miest. V jeho záujme je minimalizovať cestovné náklady a cena prepravy je úmerná dĺžke cesty, preto snaží sa nájsť najkratšiu možnú cestu tak, aby každé mesto navštívil práve raz. Keďže sa nakoniec musí vrátiť do mesta z ktorého vychádza, jeho cesta je uzavretá krivka.

Zadanie

Je daných aspoň 20 miest (20 – 40) a každé má určené súradnice ako celé čísla X a Y. Tieto súradnice sú náhodne vygenerované. (Rozmer mapy môže byť napríklad 200 * 200 km.) Cena cesty medzi dvoma mestami zodpovedá Euklidovej vzdialenosti – vypočíta sa pomocou Pytagorovej vety. Celková dĺžka trasy je daná nejakou permutáciou (poradím) miest. Cieľom je nájsť takú permutáciu (poradie), ktorá bude mať celkovú vzdialenosť čo najmenšiu.

Výstupom je poradie miest a dĺžka zodpovedajúcej cesty.

Riešenie – Simulované žíhanie

Simulované žíhanie patrí do skupiny algoritmov, ktoré využívajú na hľadanie riešenia v priestore možných stavov lokálne vylepšovanie. Zároveň sa algoritmus snaží zabrániť uviaznutiu v lokálnom extréme. Z aktuálneho uzla si algoritmus klasicky vytvorí nasledovníkov. Potom si jedného vyberie. Ak má zvolený nasledovník lepšie ohodnotenie, tak doň na 100% prejde. Ak má nasledovník horšie ohodnotenie, môže doň prejsť, ale len s pravdepodobnosťou menšou ako 100%. Ak ho odmietne, tak skúša ďalšieho nasledovníka. Ak sa mu nepodarí prejsť do žiadneho z nich, algoritmus končí a aktuálny uzol je riešením. Pre nájdenie globálneho extrému je dôležitý správny rozvrh zmeny pravdepodobnosti výberu horšieho nasledovníka. Pravdepodobnosť je spočiatku relatívne vysoká a postupne klesá k nule.

Problém je opäť reprezentovaný vektorom, ktorý obsahuje index každého mesta v nejakom poradí (nejaká permutácia miest). Nasledovníci sú vektory, v ktorých je vymenené poradie niektorej dvojice susedných uzlov.

Dôležitým parametrom tohto algoritmu je rozvrh zmeny pravdepodobnosti výberu horšieho nasledovníka. Príliš krátky (rýchly) rozvrh spôsobí, že algoritmus nestihne obísť lokálne extrémy, príliš dlhý rozvrh natiahne čas riešenia, lebo bude obiehať okolo optimálneho riešenia. Je potrebné nájsť vhodný rozvrh.

Dokumentácia musí obsahovať opis konkrétne použitého algoritmu a reprezentácie údajov. Dôležitou časťou dokumentácie je zhodnotenie vlastností vytvoreného systému a opis závislosti jeho vlastností na rozvrhu. Použite aspoň dva rôzne počty miest (napr. 20 a 30).

OPIS RIEŠENIA

Daný problém som sa rozhodol riešiť v programovacom jazyku Python. To najmä z toho dôvodu, že sa nemusím starať o pamäť a taktiež poskytuje jednoduchú syntax.

Samotná stavba programu pozostáva z viacerých tried, ktoré bližšie opíšem. Získanie vstupných informácií a výpis výstupných informácií je zabezpečený mimo týchto tried. Začnem vysvetlením vedľajších tried a potom sa vrátim k hlavnej triede, ktorá predstavuje logiku algoritmu.

Trieda City

Trieda City predstavuje jedno mesto. Pri inicializácii nového mesta sa vždy ukladá informácia o jeho súradniciach (x, y) a mene. Táto trieda obsahuje metódy:

- *generate_cities(height, width, count, random_bool)* – statická metóda, ktorá zabezpečí buď náhodné vygenerovanie súradníc miest alebo vygenerovanie miest na základe súradníc zadávaných od používateľa.
- *plot_cities(cities)* – statická metóda, ktorá vykreslí mestá na mape
- *coords(self)* – metóda, ktorá vráti tuple dvojicu súradníc mesta – (x, y).

Trieda Route

Trieda Route predstavuje jednu z možných ciest. Pri inicializácii tejto triedy sa ukladá poradie miest v akom majú byť navštevované a celková vzdialenosť.

Trieda obsahuje statickú metódu *generate_init_route(cities)*, ktorá vráti náhodne vygenerovanú cestu a statickú metódu *euclid_distance(coords_from, coords_to)*, ktorá vypočíta euklidovu vzdialenosť medzi dvoma mestami.

Trieda ďalej obsahuje metódy:

- *generate_neighbor_route(self, index_from, index_to)* – táto metóda vytvorí a vráti pre danú cestu potomka (susednú cestu) a to tak, že zamení dve mestá v poradí miest pôvodnej cesty.
- *get_distance(self)* – táto metóda vypočíta a vráti celkovú vzdialenosť danej cesty.
- *plot_route(self)* – táto metóda vykreslí danú cestu.

Teraz už poznáte vedľajšie triedy a môžeme pokračovať opisom získania vstupných informácií a hlavnou časťou algoritmu.

Vstupné informácie

Pri zapnutí programu si program od používateľa vypýta rozmery mapy - výšku a šírku, počet miest a možnosť zapnúť náhodné vygenerovanie miest alebo manuálne zadávanie súradníc. Ak je zapnuté náhodné generovanie miest používateľovi sa ukáže mapa s vygenerovanými mestami, ak sa mu nepáči môže generovanie opakovať.

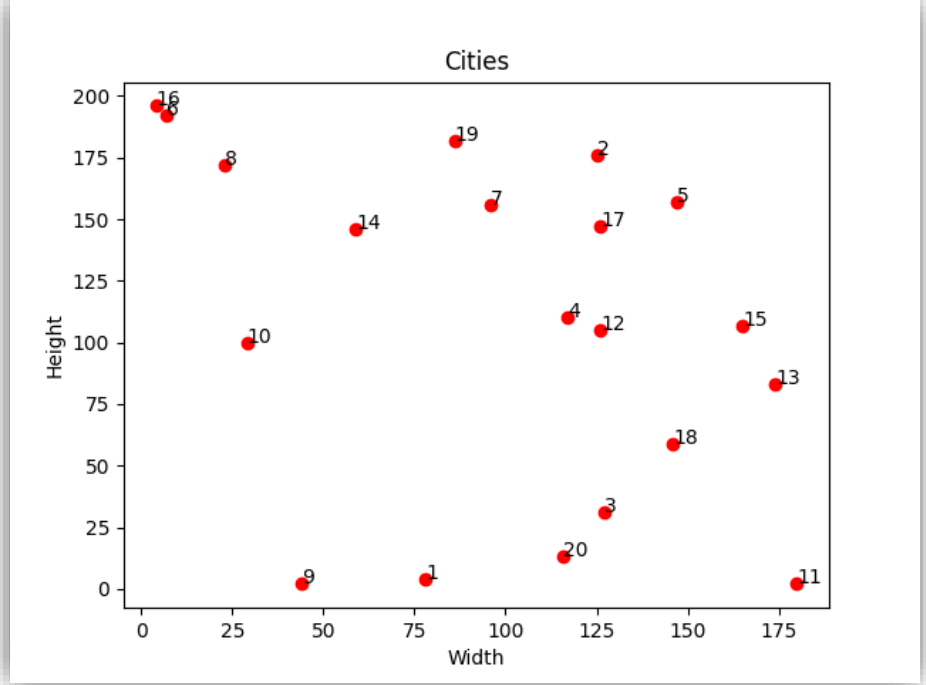
Ďalej sa používateľovi poskytne možnosť nastaviť parametre algoritmu simulovaného žíhania – počiatočná teplota, minimálna teplota, faktor chladenia a možnosť zapnúť animáciu vyhľadávania ciest.

Ďalej nasleduje inicializácia triedy Solver a spustenie samotného riešenia. Po skončení riešenia má používateľ možnosť daný problém riešiť znova s inými parametrami algoritmu alebo ukončiť program.

Príklad zadávania celého vstupu:

```
--- Map configuration ---
Enter the map height: 200
Enter the map width: 200
Enter the number of cities: 20
Enter '1' to generate random cities or '0' to enter coordinates: 1
```

Vygenerované mestá:



Pokračovanie zadávania vstupu:

```
Enter '1' to generate again or '0' to continue: 0

--- Simulated annealing algorithm configuration ---
Enter the initial temperature: 30
Enter the minimal temperature: 1
Enter the cooling factor: 0.99
Enter '1' for live animation or '0' for no animation: 1
```

Trieda Solver

Táto trieda obsahuje základ algoritmu, logiku vykonávania. Pri inicializácii tejto triedy sa uloží informácia o náhodne vygenerovanej počiatočnej ceste, počiatočnej a minimálnej teplote, faktore chladenia a počte generovaných susedov pre každú etapu podľa vzorca: n*(n-1)/2, kde n je počet miest, pretože v každej etape generujem susedov tak, že skúšam zameniť všetky možné dvojice miest.

Samotný algoritmus obsahuje funkcia *solve(self, animation)*. Algoritmus pozostáva z dvoch cyklov. Vo vonkajšom cykle sa mení teplota – počiatočná teplota sa každým cyklom ochladzuje na základe faktora chladenia až pokým neprekročí hranicu minimálnej teploty. Vo vnútornom cykle sa na základe počtu generovaných susedov (dĺžky jednej etapy) pre túto cestu generujú susedné cesty (zamieňaním všetkých možných dvojíc miest). Pre každú vygenerovanú cestu sa zavolá statická funkcia *accept(current_route, neighbor_route, temp)*, ktorá vráti True, ak je táto cesta akceptovaná, inak False. Akceptovanie cesty znamená, že celková dĺžka vygenerovanej cesty je lepšia alebo náhodne vygenerované číslo spadá do rozmedzia pravdepodobnosti akceptovania, ktorá je vypočítaná Boltzmannovou distribučnou rovnicou: probability = math.exp((current_distance – new_distance) / teplota). Ak je nová cesta akceptovaná, uloží sa ako aktuálna cesta a skontroluje sa či nie je najlepšia celkovo, ak hej uloží sa ako celkovo najlepšia cesta.

Toto bola opísaná logika algoritmu. Okrem nej algoritmus taktiež meria celkový čas vykonávania, zapisuje celkové dĺžky vygenerovaných ciest po každej etape a po každej etape taktiež plotuje aktuálnu cestu, ak je zapnutý parameter *animation*.

Algoritmus vracia najlepšiu nájdenú cestu, dĺžky ciest po každej etape a celkový čas vykonávania.

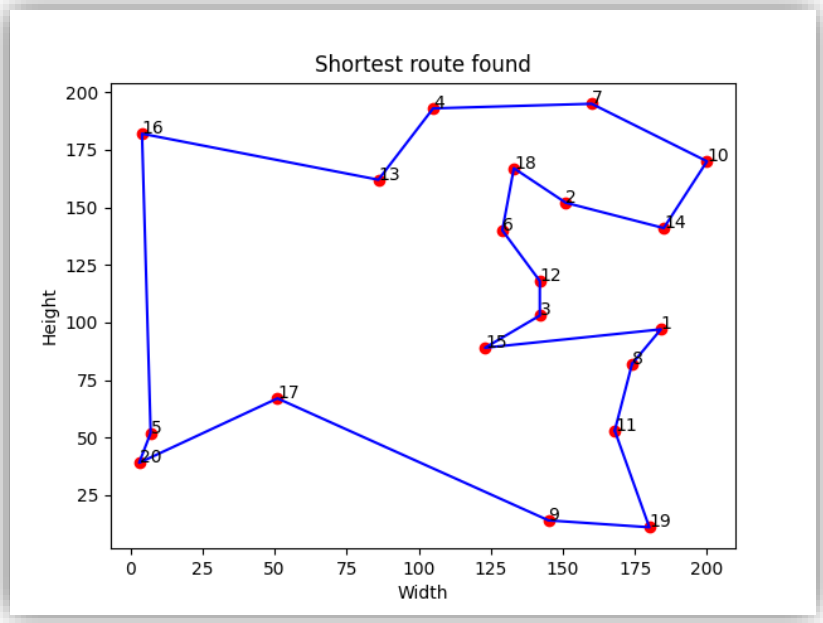
Výstupné informácie

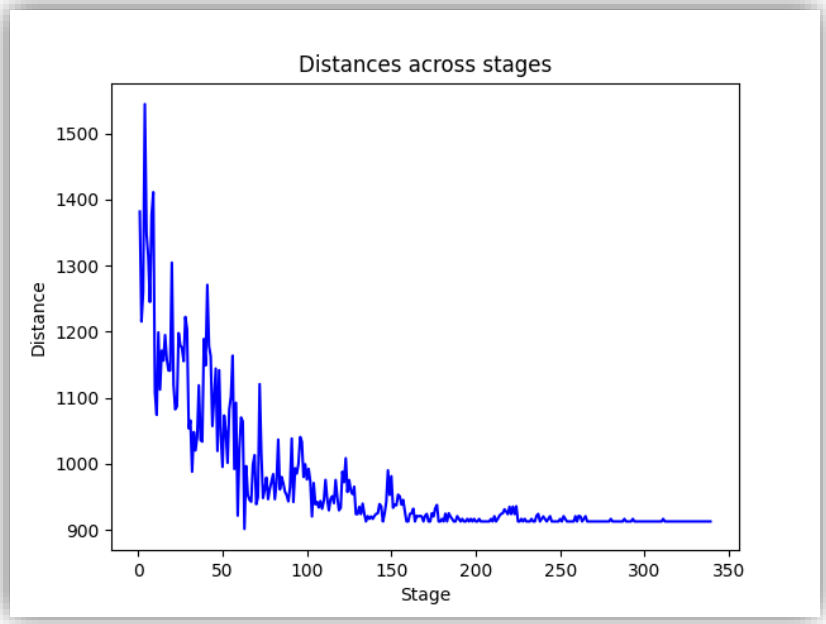
Po skončení hľadania riešenia sa vykreslí najlepšia nájdená cesta, jej vzdialenosť a celkový čas vykonávania. Na záver sa v grafe vykreslia

vzdialenosti ciest po každej etape, kde môžeme sledovať proces vývoju hľadania.

Príklad výstupu riešenia:

```
--- Summary ---
Shortest route: 11 19 9 17 20 5 16 13 4 7 10 14 2 18 6 12 3 15 1 8
Total distance: 901.33
Time taken: 1.36 sec
```





Modrá krivka predstavuje zmenu priemernej vzdialenosti naprieč etáp.

TESTOVANIE

Testovanie parametrov som vykonal pre mapu 200x200 s počtami miest 20 a 30.

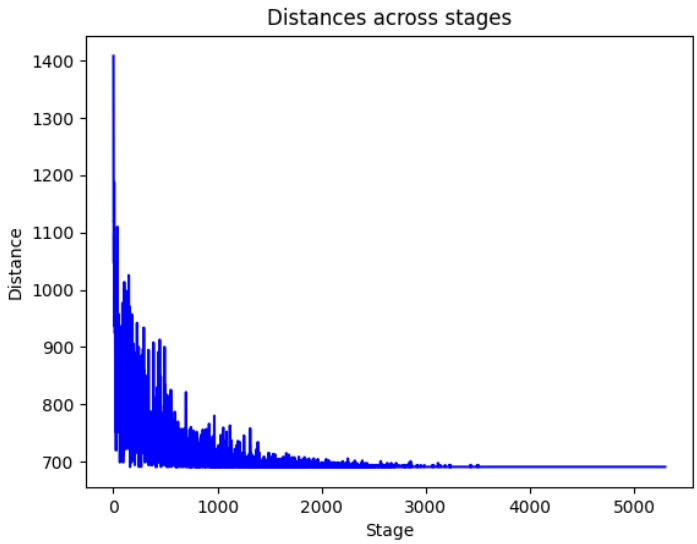
Počet miest – 20

Výsledky:

Nastavenia parametrov			Výsledky	
Počiatočná teplota	Minimálna teplota	Faktor chladenia	Celková vzdialenosť	Čas vykonávania
20	0,1	0,999	691,08	24,47
35	0,1	0,999	691,08	32,98
50	0,1	0,999	691,08	30,03
20	0,1	0,99	691,08	2,24
35	0,1	0,99	691,08	2,47
50	0,1	0,99	691,08	2,64
20	0,1	0,9	859,77	0,22
35	0,1	0,9	691,08	0,23
50	0,1	0,9	691,08	0,25

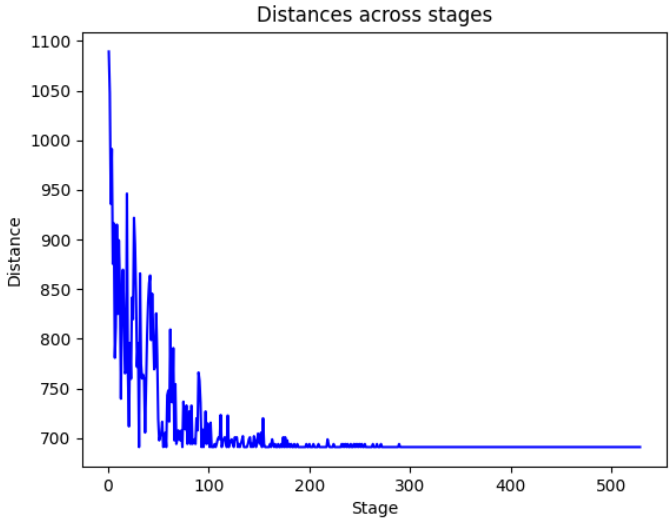
Ako môžeme vidieť, z dát v tabuľke nám vyplýva, že pri počte miest 20 a dostatočne pomalom ochladzovaní dávajú všetky počiatočné teploty rovnaké výsledky. Pri veľmi pomalom ochladzovaní sa však doba riešenia výrazne natiahne. Pri veľmi rýchlom ochladozvaní síce dosiahneme veľmi rýchly čas a v tomto prípade aj dobré výsledky, ale to bude pravdepodobne zapríčinené dobrým vygenerovaním miest a počiatočnej cesty, pretože môžeme vidieť, že pri počiatočnej teplote 20 bol výsledok o moc horší. Skúsil som overiť výsledky pre inú konfiguráciu 20 miest a môj predpoklad sa potvrdil, výsledky boli horšie. Vyššia počiatočná teplota ako 20 sa v tomto prípade javí ako zbytočná a iba naťahuje dobu riešenia. Celkovo sa najvýhodnejšia zdá počiatočná teplota 20 s faktorom ochladzovania 0,99.

Faktor ochladzovania – 0,999, počiatočná teplota – 20



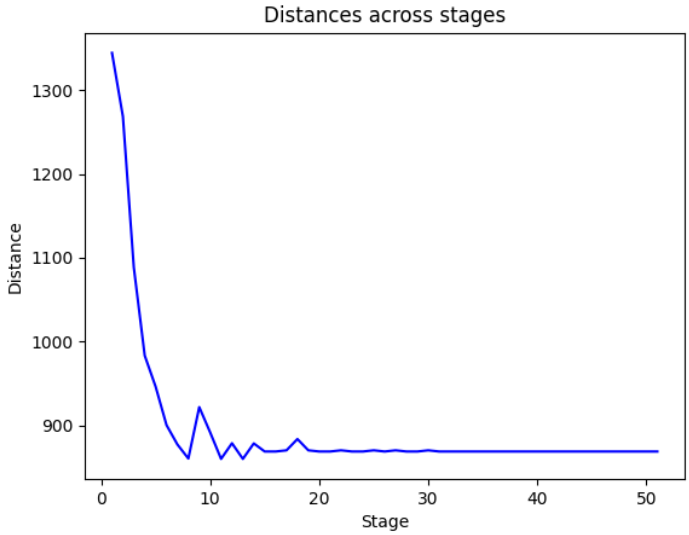
Na tomto grafe môžeme vidieť, že pri veľmi pomalom ochladzovaní riešenie síce prekonáva lokálne extrémy, ale pomerne dlhú dobu sa pohybuje okolo zlého bodu, čo výrazne naťahuje dobu riešenia.

Faktor ochladzovania – 0,99, počiatočná teplota - 20



Na tomto grafe môžeme vidieť takmer ideálne riešenie, ktoré pekne prekonáva lokálne extrémy, rovnako dostávajú priestor aj horšie riešenia, až sa nakoniec nájde globálne minimum. Toto riešenie by sa dalo ešte z časového hľadiska vylepšiť zvolením väčšej minimálnej teploty, pretože vidíme, že od etapy 300 sa žiadne lepšie minimum už nenašlo.

Faktor ochladzovania 0,9, počiatočná teplota – 20



Na tomto grafe môžeme vidieť, že pri veľmi rýchlom ochladzovaní sa prekoná veľmi málo lokálnych extrémov a teda samotné výsledky sú s veľkou pravdepodobnosťou o dosť horšie, nájdené minimum nebude globálne, ale lokálne.

Počet miest – 30

Výsledky:

Nastavenia parametrov			Výsledky	
Počiatočná teplota	Minimálna teplota	Faktor chladenia	Celková vzdialenosť	Čas vykonávania
20	0,1	0,999	979,37	94,03
35	0,1	0,999	997,3	100,32
50	0,1	0,999	1014,83	110,28
20	0,1	0,99	979,37	8,2
35	0,1	0,99	997,3	8,91
50	0,1	0,99	1018,98	9,62
20	0,1	0,9	1092,92	0,72
35	0,1	0,9	1024,27	0,81
50	0,1	0,9	992,69	0,84

Dáta v tejto tabuľke nám potvrdzujú zistenia z predchádzajucej tabuľky. Ideálne riešenie je počiatočná teplota 20 s faktorom ochladzovania 0,99.

Počiatočná teplota 20 však nebude ideálna pre väčší počet miest, so zvyšujúcim sa počtom miest bude nutné pomaly zvyšovať počiatočnú teplotu pre dosiahnutie lepších výsledkov.

ZÁVER

Myslím, že som pokryl všetky veci, ktoré sa od nás v tomto zadaní očakávali. Používateľovi som poskytol jednoduché prostredie a taktiež viacero možností nastavenia daného riešenia. Samotné nastavenia parametrov som pekne odtestoval a zdokumentoval výhody a nevýhody.