SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Zadanie 3 – Zenová záhrada

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Riešený problém – 3a. pomocou HC,TS a SA

HC – Hill climbing (lezenie do kopca)
TS – Tabu search (zakázané prehľadávanie)
SA – Simulated annealing (simulované žíhanie)

Opis zadania:
Zenová záhrada je plocha vysypaná pieskom, kde sa môžu nachádzať aj kamene, ktoré sú nepohyblivé. Mních má za úlohu pohrabať piesok v záhrade pomocou hrablí tak, ako je znázornené na obrázku.

Obrazok 1. Zdroj: http://www2.fiit.stuba.sk/~kapustik/zen.html

Mních môže hrabať iba vodorovne alebo zvislo. Začína vždy na okraji záhrady a hrabe v danom smere až po druhý okraj alebo po prekážku. Mimo záhrady môže chodiť ako chce. Ak však príde ku prekážke (kameňu alebo pohrabanému piesku) musí sa otočiť buď doľava alebo doprava to už si rozhodne mních. Ak má voľný iba jeden smer otočí sa tam a ak sa nemá kam otočiť je koniec hry. Úspešná hra je taká, kde dokázal mních pohrabať celú záhradu.
Moja reprezentácia vyššie uvedenej záhrady:
(0 – nepohrabané, -1 – kameň, 1 až n – iterácia hrabania)

0	0	1	0	0	0	0	0	10	10	8	9
0	0	1	0	0	-1	0	0	10	10	8	9
0	-1	1	0	0	0	0	0	10	10	8	9
0	0	1	1	-1	0	0	0	10	10	8	9
0	0	-1	1	0	0	0	0	10	10	8	9
2	2	2	1	0	0	0	0	10	10	8	9
3	3	2	1	0	0	0	0	-1	-1	8	8
4	3	2	1	0	0	0	0	5	5	5	5
4	3	2	1	0	0	0	11	5	6	6	6
4	3	2	1	0	0	0	11	5	6	7	7

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Základné informácie

Program vyhotovený v jazyku Python vo verzii 3.8.6
Program je spustiteľný ako .exe
Výstup programu je do konzoli
Použité knižnice:

- random
- numpy (inštalácia cez príkaz **pip install numpy**)
- time
- os

Program je rozdelený na 3 hlavné časti:

- Ručné zadávanie dát záhrady
- Načítanie dát záhrady zo súboru (.txt)
- Testovanie algoritmov

Názvy funkcií algoritmov:

- HC – hill_climb()
- TS – tabu_search()
- SA – simulated_annealing()

Nastavenia parametrov

Pre nastavenia sú tieto globálne premenné

```
decisions_size = 8                #urcuje aj pocet susedov s zmenenymi rozhodnutiami (nesmie byt 0)
number_of_monks_random_start = 8  #pocet susedov s rovnakymi rozhodnutiami ale inym startom
tabu_list_size = 5                #velkost tabu listu
annealing_temperature = 30        #teplota simulovaneho zihania
annealing_cooling = 0.5           #o kolko sa teplota ochladi po cykle
iterations_of_algorithms = 500    #maximalny pocet vykonani kazdeho algoritmu
```

Reprezentácia mnícha

```
class Monk:
    def __init__(self, fitness, decisions : list, steps : list, spawns : list):
        self.fitness = fitness  #pocet policok ktore pohrabal
        self.decisions = decisions  #list rozhodnuti [1,-1,1,...]
        self.steps = steps  #list krokov, ktore vykonal, (startovacia pozicia monka steps.[0])
        self.spawns = spawns #list krajnych policok, kde moze vstupit do zahrady
```

Reprezentácia pohybov

```
rotations =np.array([
    (0, -1),    #left
    (-1, 0),     #up
    (0, 1),     #right
    (1, 0)     #down
])
```

Záhrada je reprezentovaná ako 2D list

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Opis riešenia

Po spustení program ponúkne možnosti. Používateľ si vyberie či chce zadať dáta pre záhradu ručne alebo ich načítať zo súboru prípadne či chce odtestovať program pripadne ukončiť program.

1. **Možnosť ručného zadania –** manual_input()
    - program si vyžiada rozmery záhrady následne počet kameňov a ich súradnice. Ďalej vykreslí záhradu a ponúkne algoritmy, z ktorých si užívateľ zvolí jeden po zvolení sa spustí tento algoritmus zo zadanou záhradou a vypíše sa fitness mapy, fitness najlepšieho mnícha, jeho rozhodnutia a pohrabaná záhrada od tohto mnícha.

2. **Možnosť načítania zo súboru –** file_input()
    - Program si vyžiada súbor (.txt), následne sa z tohto súboru načítajú dáta pre záhradu. V súbore musia byť v prvom riadku rozmery záhrady oddelené medzerou (napr. 10 12) prvý rozmer je počet riadkov a druhý počet stĺpcov následne v ďalších riadkoch sú pozície kameňov (napr. 0 1) prvá súradnica je y a druhá x v súradnicovom systéme. Ďalej vykreslí záhradu a ponúkne algoritmy, z ktorých si užívateľ zvolí jeden po zvolení sa spustí tento algoritmus zo zadanou záhradou a vypíše sa fitness mapy, fitness najlepšieho mnícha, jeho rozhodnutia a pohrabaná záhrada od tohto mnícha.

3. **Tester –** tester()
    - Program si vyžiada rozmery záhrady pre akú chceme program odtestovať a s akým počtom kameňov, pozície kameňov sa náhodne vygenerujú. Tieto dáta sa zapíšu do súboru input.txt pre možnosť ďalšej optimalizácie záhrady s týmito dátami alebo porovnanie pre rôzne nastavenia. Vykreslí sa vygenerovaná záhrada a spustí sa testovanie všetkých algoritmov pre všetky možné začiatočné pozície v záhrade a následne program poskytne výsledky pre každý z algoritmov v podobe priemerného času trvania algoritmu nad danou záhradou. Fitness záhrady a fitness najlepšieho mnícha a priemerný fitness mníchov

Opis algoritmov

Do každej funkcie pre daný algoritmus sa posiela nepohrabaná záhrada, štartovacie políčko, rozmery záhrady, a všetky krajné vstupné políčka záhrady, ktoré sú nepohrabané a nie je tam prekážka. Následne sa získa fitness mapy, vytvorí sa prvý mních so štartovacím políčkom a prístupovými (krajnými) políčkami do záhrady pomocou funkcie create_monk() následne sa spustí funkcia rake_garden(), ktorá slúži na hrabanie zadanej záhrady s poskytnutým mníchom tato funkcia je opísaná nižšie v opise rozhodovania.
Ďalej sa prvý vytvorený mních zvolí za najlepšieho globálneho mnícha a najlepšieho lokálneho kandidáta.

1. **HC**
    - Je implementovaný vo funkcii hill_climb(). V cykle na hľadanie najlepšieho mnícha sa končí ak sa našiel mních, ktorý pohrabal celú záhradu alebo sa vykonal presný počet iterácií alebo prestala stúpať fitness mníchov pri iteráciach. V cykle sa najprv vygeneruje list susedných mníchov pomocou funkcie get_neighbors(). Zapnutým zotriedením (sorting = True) docielime že získaný list susedných mníchov bude zotriedený a mních s najlepším fitness bude prvý v liste. Následne sa skontroluje či je lepší ako globálne mních ak je prehlásime ho za nového globálneho a pokračujeme v ďalšej iterácií cyklu, ak však už nie je lepší tak skončíme cyklus a funkcia vráti globálne najlepšie mnícha.

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

2. **TS**
   - Je implementovaný vo funkcii tabu_search() a funguje rovnako ako HC, ale s tým rozdielom, že ma obsahuje list zakazaných stavov tzv. tabu list a neskončí, keď sa nenašiel lepší susedný mních. Ak sa nenašiel lepší susedný mních pridá sa lokálne najlepší do tabu listu a za lokálne najlepšie sa zvolí mních s nižšou fitness a keďže je list susedným mníchov zotriedený od najlepšieho po najhorší lebo je zapnute zotriedenie, môžeme to chápať ako zostupovanie z kopca s lokálnym fitness extrémom. V **TS** sa môžeme zacykliť, preto program obsahuje globálnu premennú s názvom „iterations_of_algorithms“, ktorá obmedzuje počet iterácií algoritmu na číslo v tejto premennej.
3. **SA**
   - je implementovaný vo funkcii simulated_annealing(). Tento algoritmus je HC s tým že tiež nekončíme ak sa nenašiel lepší a je rozšírený o teplotu algoritmu, ktorá reprezentuje percentuálnu šancu na akceptovanie horšieho susedného mnícha ako lokálne najlepšieho a následne jeho zvolenie za najlepšieho lokálneho. Táto teplota sa z každou iteráciou algoritmu znižuje. Teplota je stanovená v globálnej premennej „annealing_temperature“ a tak isto aj ochladzovanie „annealing_cooling“. A pre širší výber mnícha s horším fitness je zotriedenie vypnuté.

Funkcia get_neighbors()

Vracia list susedných mníchov. Počet susedných mníchov odpovedá veľkosti listu rozhodnutí pre mnícha, kde každý susedný mních ma zmenené práve jedno rozhodnutie + zadanému počtu mníchov v globálnej premennej „number_of_monks_random_start“ s náhodne vybraným štartovacím políčkom.
Vo funkcií sa rovno aj ohodnotia mnísi spustením funkcie rake_garden() pre každého mnícha. Ďalej funkcia môže aj list zotriediť podľa fitness mníchov ak pri volaní funkcie argument „sorting“ bude True.

Opis rozhodovania

Funkcia rake_garden()

V tejto funkcií sa deje hrabanie záhrady mníchom, pre začiatočné políčko sa získa smer, ktorým sa má mních vydať pomocou funkcie get_direction(), premenná decision_index ukazuje na rozhodnutie na danom indexe v liste rozhodnutí mnícha. List rozhodnutí mnícha obsahuje postupnosť 1 a -1, kde 1 reprezentuje otočenie vpravo a -1 otočenie vľavo.
Mních hrabe políčka záhrady v danom smere až kým nevyjde zo záhrady, v tomto prípade pokračuje v hrabaní od nového vstupu s novým smerom ak vstupuje do záhrady z krajného políčka tak sa rozhoduje akým smerom sa vyberie podľa rozhodnutia v liste rozhodnutí mnícha na indexe na ktorý ukazuje premenná decision_index . (*toto rozhodovanie pre rohové krajné políčka sa deje vo funkcii get_direction() ak rozhodnutie je 1 vstupuje vodorovne ak -1 vstupuje horizontálne a decision_index sa nezvyšuje*).
Ak však mních narazí, je pred ním prekážka alebo pohrabaný piesok a ma na výber či sa otočí vpravo alebo vľavo tak sa zvolí otočenie podľa rozhodnutia v liste rozhodnutí mnícha na indexe na ktorý ukazuje premenná decision_index a následne sa táto premenná zvýši o 1. A pokračuje v hrabaný v novom smere.
Ak mních narazil a nemôže sa nikde otočiť a nie je na rohových políčkach záhrady tak skončil hrabanie.

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

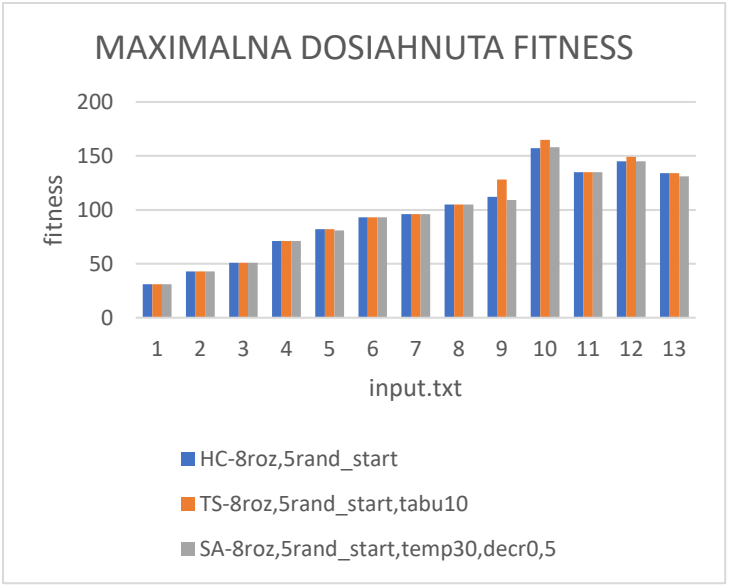
Spôsob testovania

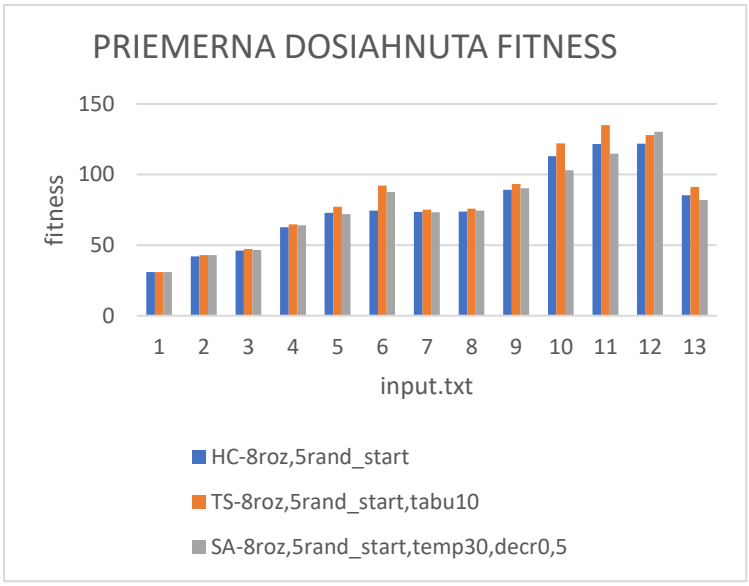
Testovanie prebehlo na 13 záhradách, ktoré sú priložené ako textové súbory začalo sa od rozmeru 6x6 až po rozmer 18x18 s cca 10% kameňov pre daný rozmer záhrady pre 2 typy nastavení. Ďalej rôzne nastavenia jednotlivých algoritmov sa testovali na záhrade, ktorá je aj na stránke predmetu.

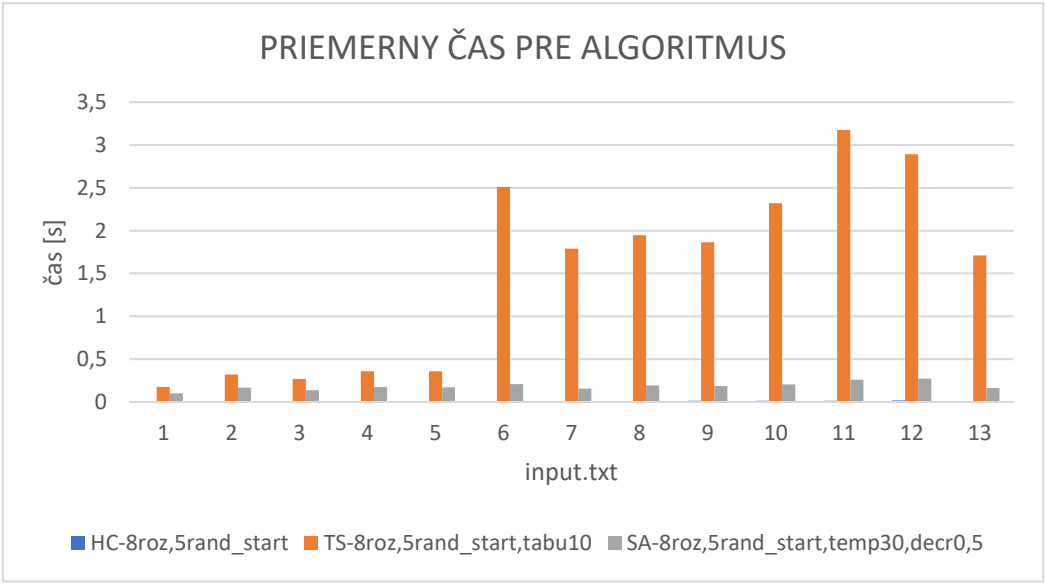
Testovanie na 13 záhradách

Prvé nastavenia:

```
decisions_size = 8
number_of_monks_random_start = 5
tabu_list_size = 10
annealing_temperature = 30
annealing_cooling = 0.5
iterations_of_algorithms = 100
```



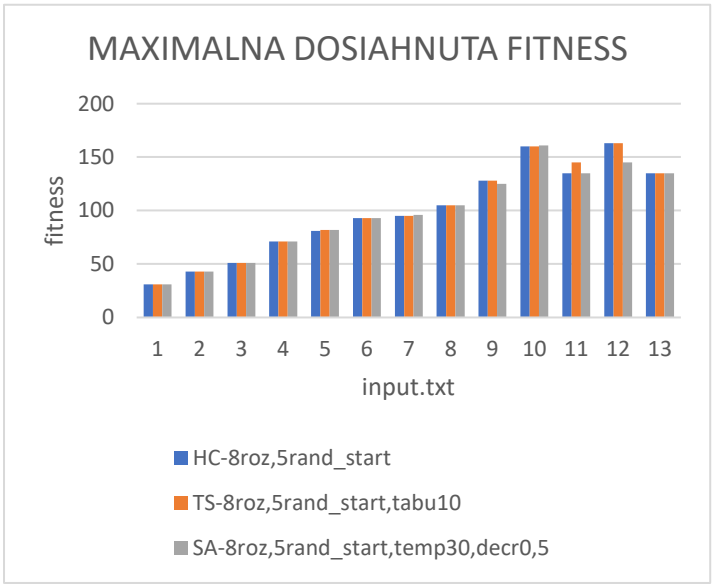


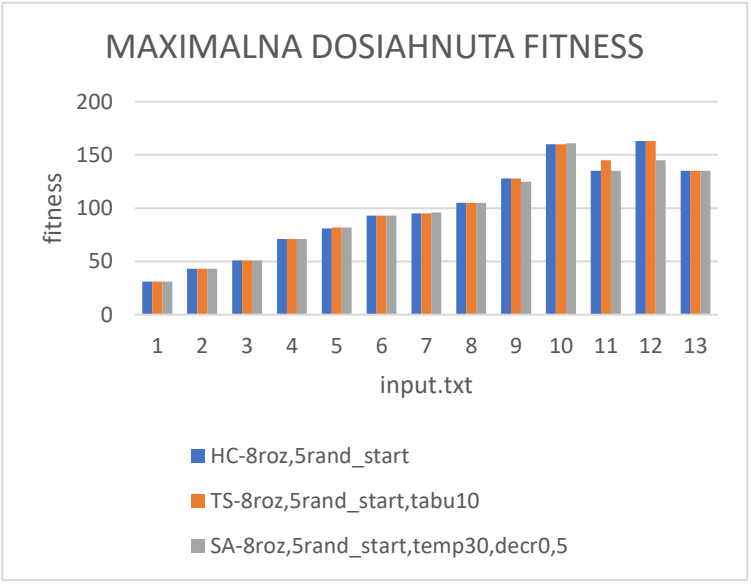


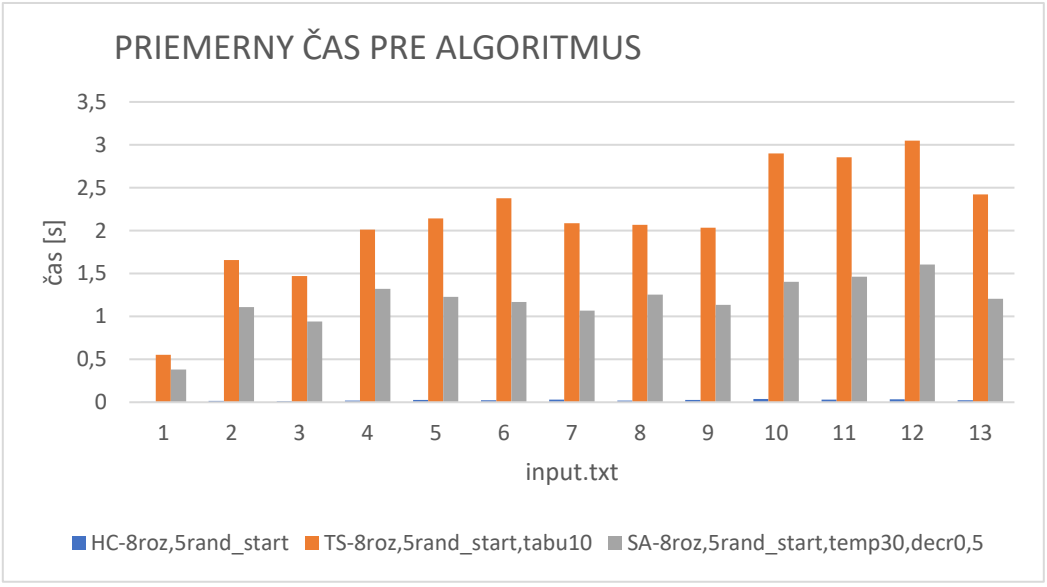
Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Druhé nastavenia:

```
decisions_size = 16
number_of_monks_random_start = 20
tabu_list_size = 30
annealing_temperature = 30
annealing_cooling = 0.2
iterations_of_algorithms = 200
```

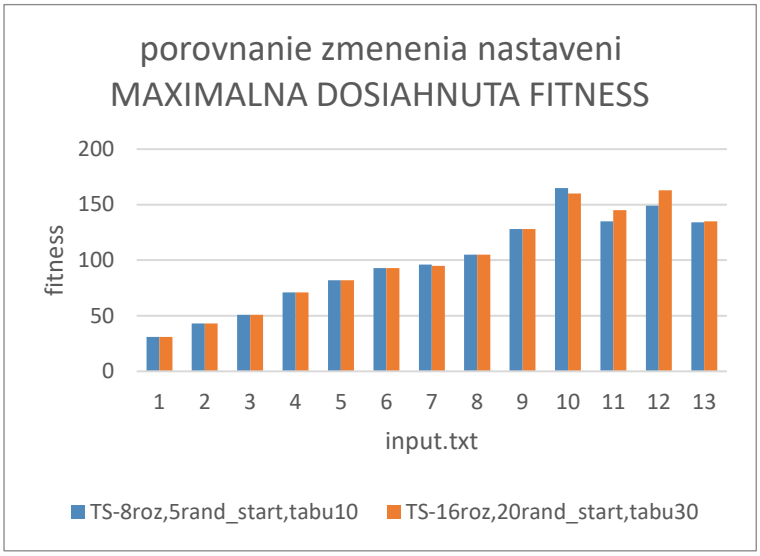


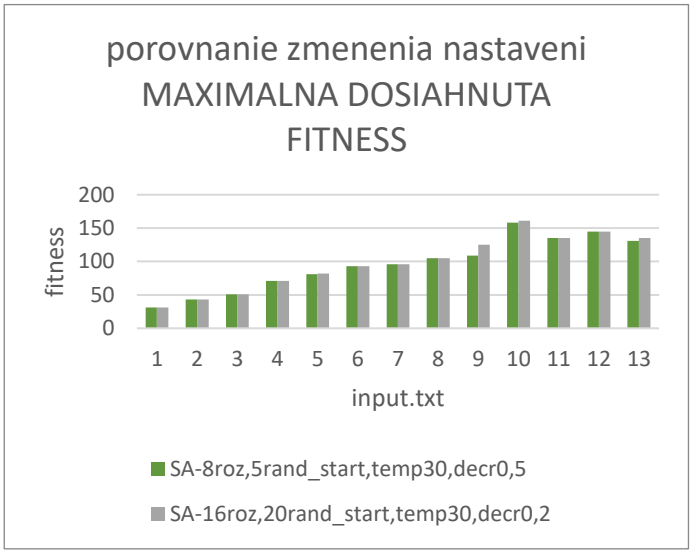


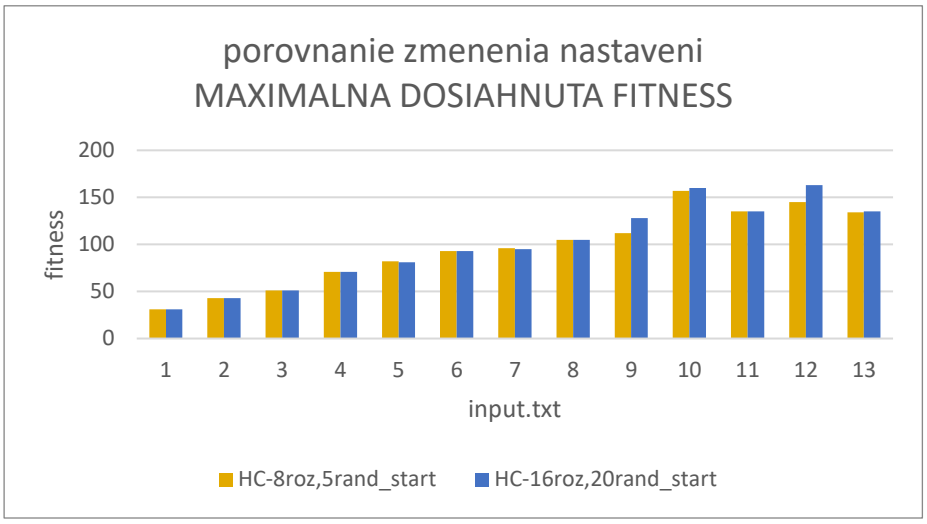


Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Porovnanie maximálnej dosiahnutej fitness pre každý algoritmus:

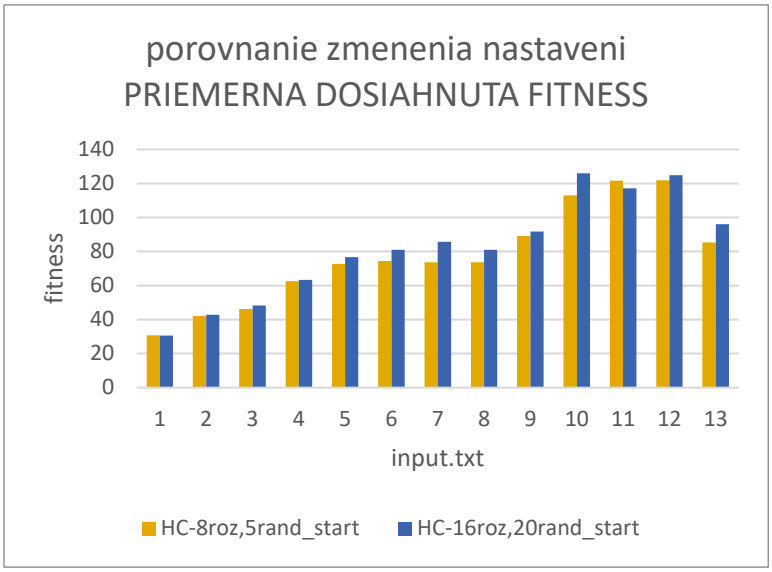


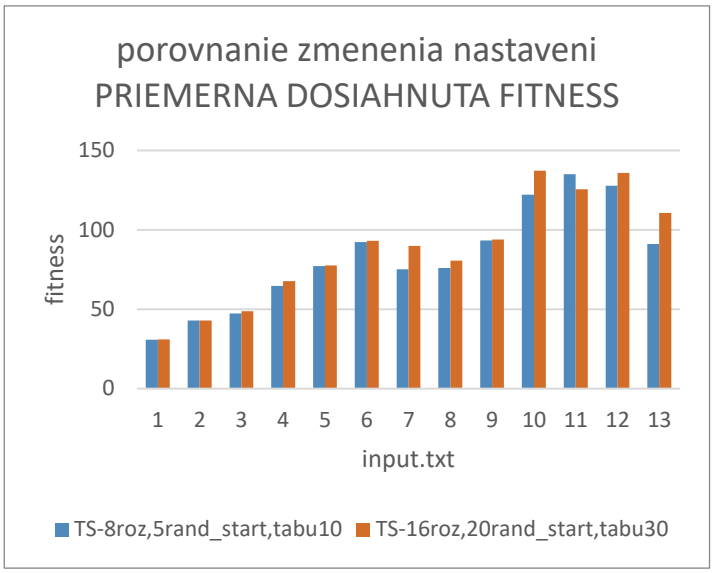


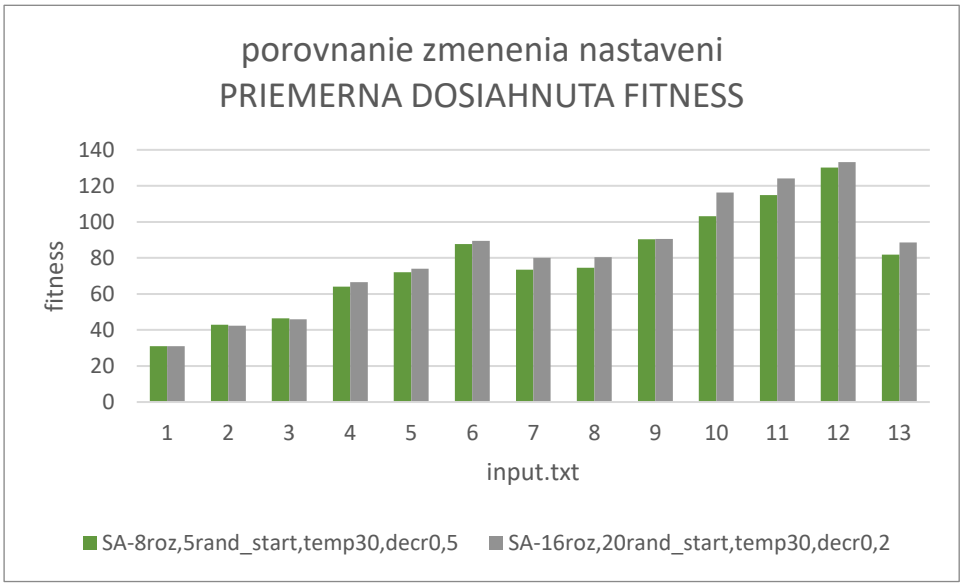


Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Porovnanie priemernej dosiahnutej fitness pre každý algoritmus:

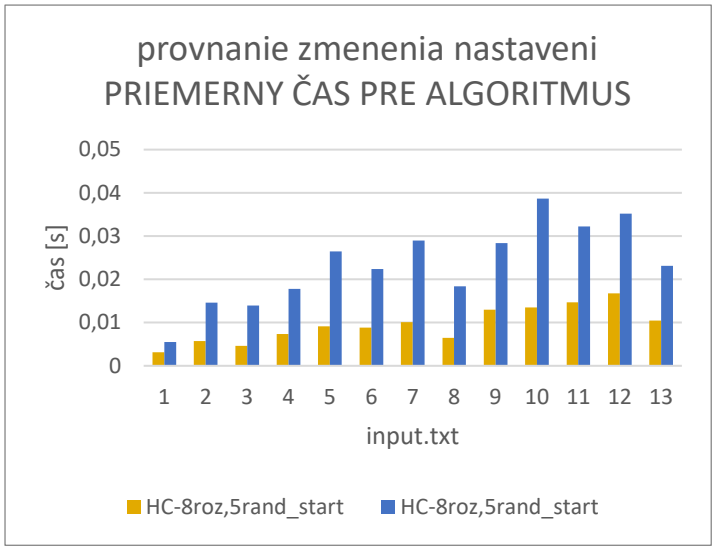


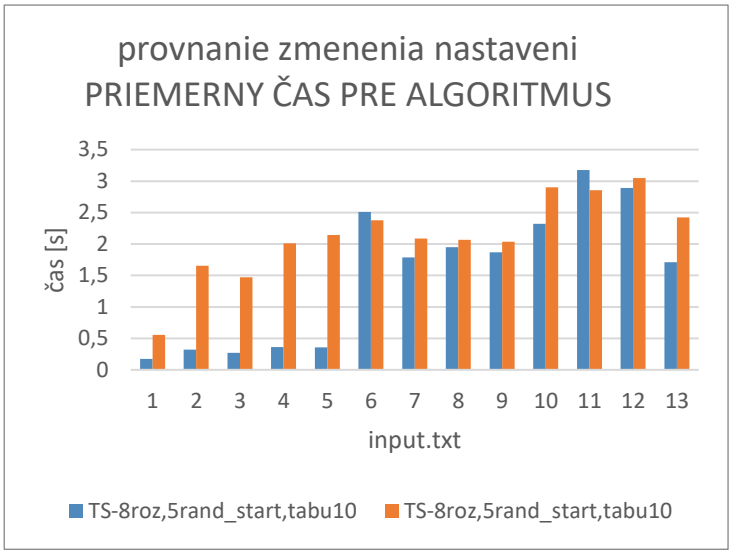


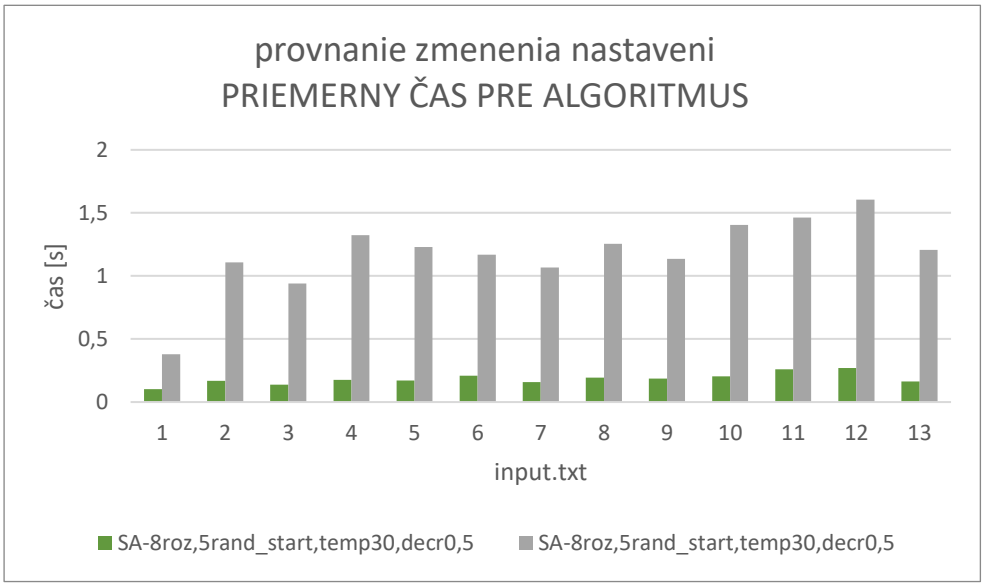


Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Porovnanie priemerného času trvania pre každý algoritmus:



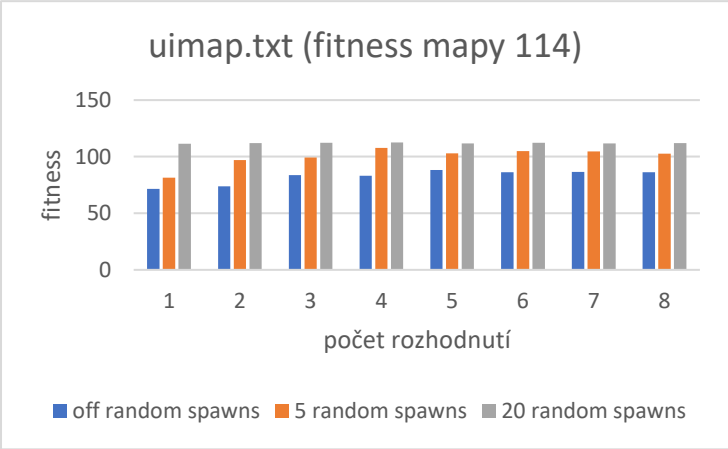


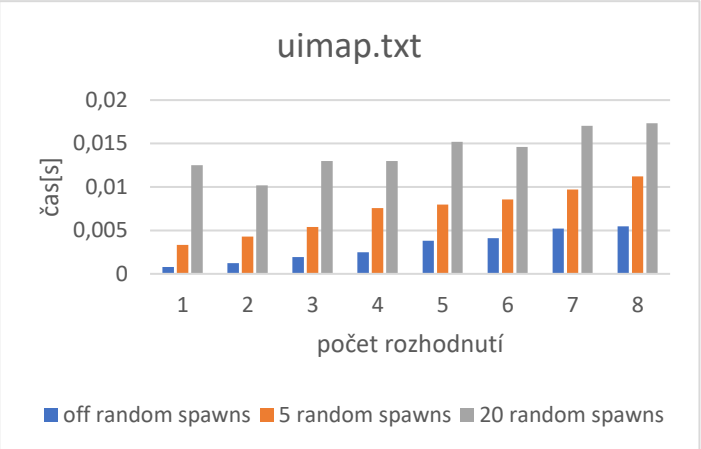


Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

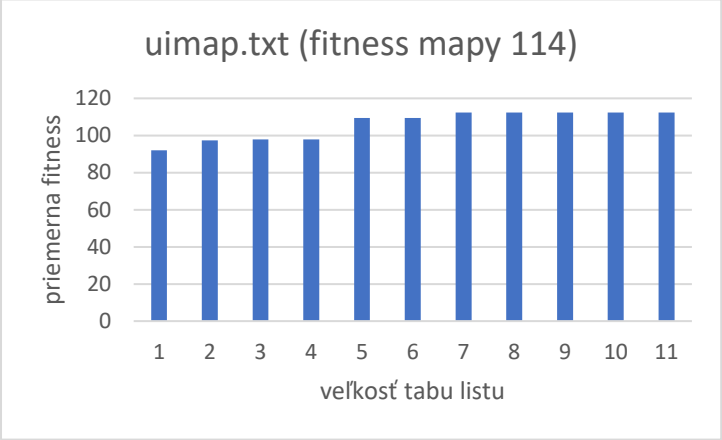
Testovanie nastavení algoritmov na záhrade, ktorá je na stránke predmetu (uimap.txt).

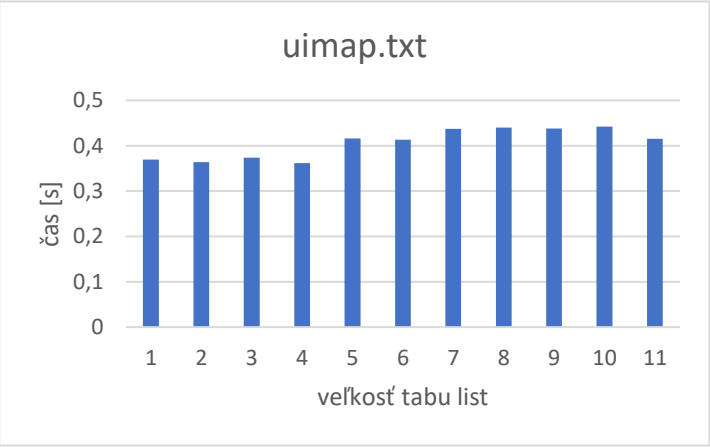
HC: Boli menené nastavenia, ktoré ovplyvňujú **HC** a to počet rozhodnutí a počet susedných mníchov s náhodným začiatkom.



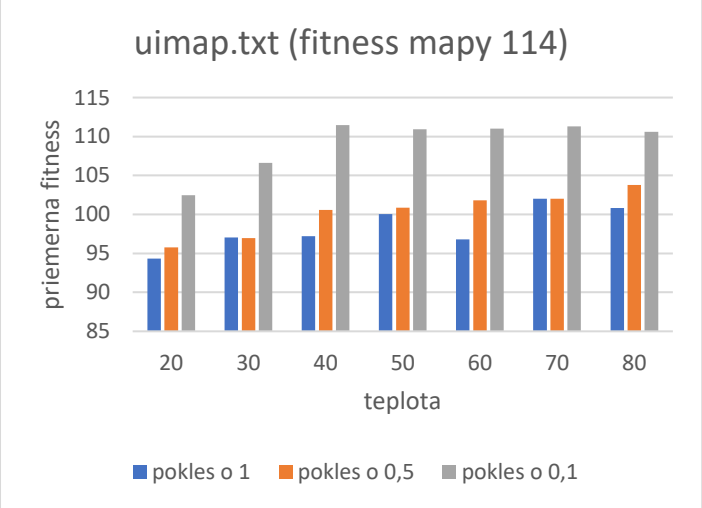


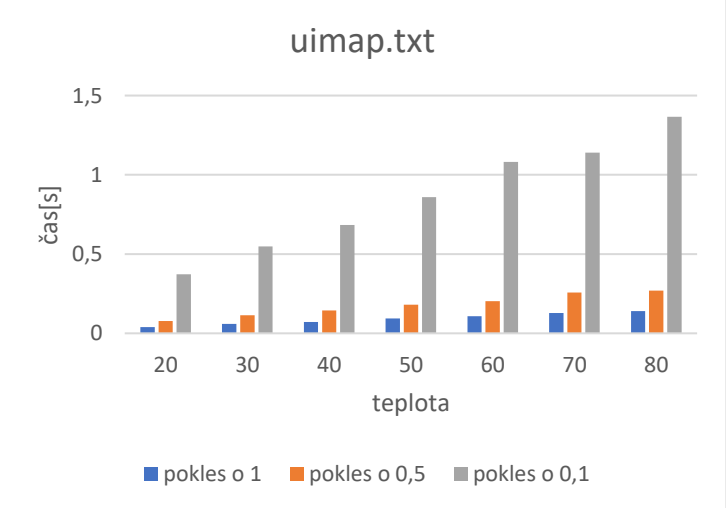
TS: Pre tabu search boli nastavene 4 rozhodnutia [-1,-1,1,-1], 4 susední mnísi s náhodným štartom. A postupne bola menená veľkosť tabu listu.





SA: Menila sa počiatočná teplota a pokles teploty za 1 iteráciu algoritmu





Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia

Zhodnotenie riešenia

Hill climbing algoritmus bol najrýchlejší, potom nasledoval algoritmus simulovaného žíhania a časovo najnáročnejší bol tabu search. Nevýhoda hill climbing je že sa vždy dostane do lokálneho maxima, ktoré nie vždy je globálne maximum hoci je rýchli. Pri tabu search je možnosť vyjsť z lokálneho maxima, ktoré nie je globálne pomocou tabu listu, a príde nám to aby sme postupne zliezali z kopca z lokálnym maximom a následne máme možnosť sa dostať do globálneho maxima avšak na úkor toho že sa môžeme zacykliť ak sa zo žiadneho stavu nie sme schopný sa dostať do globálneho maxima. Pri simulovanom žíhaní to je už viac o percentuálnej šanci akceptovať horší stav, ktorý môže viesť ku globálnemu maximu takže tu ide skôr o náhodu že zrovna akceptujeme, horší stav, ktorý nás dovedie ku globálnemu maximu. Pri simulovanom žíhaní by bolo možno dobré vyskúšať aj aká je jeho účinnosť pre zotriedený list mníchov podľa fitness, ale výsledok by bol asi taký že by sme boli málokedy schopný zísť dostatočne na taký nízky fitness aby sme sa dokázali dostať do globálneho maxima, z tohto dôvodu je zotriedené pri simulovanom žíhaní vypnuté.

Z výsledkov testovania môžeme usúdiť že susedný mnísi, ktorý začínajú na iný začiatočných pozíciách dokážu o najlepšie vylepšiť algoritmus a jeho výslednú fitness keď zvyšovať iba počet rozhodnutí už nepomáha. Pri tabu search je kritické zvoliť správnu veľkosť tabu listu, nesmie byť príliš malá aby sme boli schopný zísť z lokálneho maxima a prejsť do globálneho maxima pripadne prejsť cez viac lokálnych maxím a dôjsť do globálneho a ak je tabu list príliš veľký tak nás to stojí veľa času a nemusíme ani nájsť globálne maximum lebo s danými nastaveniami nemusíme byť schopný sa tam dostať, máme malý počet rozhodnutí mnícha.

Pri simulovanom žíhaní je najideálnejšie zvoliť nie moc rýchle ani moc pomalé ochladzovanie, v našom prípade je ideálne zvoliť pre simulované žíhanie zvoliť počiatočnú teplotu okolo 40 a znižovať o 0.5 aby to nám algoritmus nezabral príliš veľa času ale pokiaľ čas nie je problém môžeme zvoliť aj pomalšie ochladzovanie.

Vylepšenie, ktoré má napadá tak je nejako aplikovať rozhodnutia aj na výber ďalšieho vstupu do mapy, to by mohlo vylepšiť fitness mnícha, pri simulovanom žíhaní mať list zotriedený ak je najlepší mních v susedoch tak ho prehlásim za globálneho a pokračujem ak však nie je tak príde na rad podmienka akceptovania aj horšieho stavu ak prejde tak za zvolí nejaký mních vzadu v liste (napr. vyberie sa z posledných 10). a nemusíme prechádzať nezotriedený list susedných mníchov).

Adrián Vančo
ID: 103171
Predmet: Umelá inteligencia