SLOVENSKÁ TECHNICKÁ UNIVERZITA
Fakulta informatiky a informačných technologií
Ilkovičova 2, 842 16 Bratislava 4

Zadanie č. 3 - Riešenie problému obchodného cestujúceho algoritmov zakázaného prehľadávania a simulovaného žíhania

Dávid Penťa ID: 110871

Umelá inteligencia

FIIT STU 2021/2022

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

Program na začiatku vytvorí náhodnú trasu, ktorú si zapamätá ako vektor, napríklad: [19, 4, 6, 8, 2, 17, 18, 14, 13, 3, 15, 10, 5, 1, 11, 16, 9, 12, 7, 0]. Na nájdenie okolia som najprv používal metódu výmeny dvoch náhodných miest v trase, ale zistil som, že algoritmy zakázaného prehľadávania a simulovaného žíhania dosahujú lepšie výsledky, keď sa používa metóda otočenia podreťazca (reversed sub-string). Tá funguje vybraním dvoch náhodných miest a otočením trasy medzi nimi. Napríklad:

Pôvodná cesta:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]

Cesta po otočení podreťazca:

[0, 1, 2, 3, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]

V testovaní som tieto dva spôsoby hľadania susedov spolu porovnal.

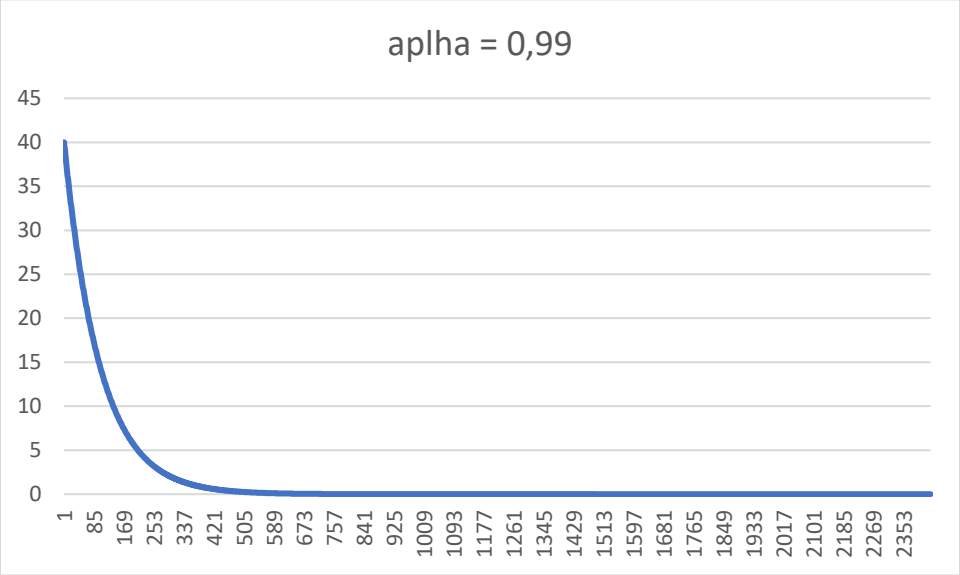
Zakázané prehľadávanie (tabu search)

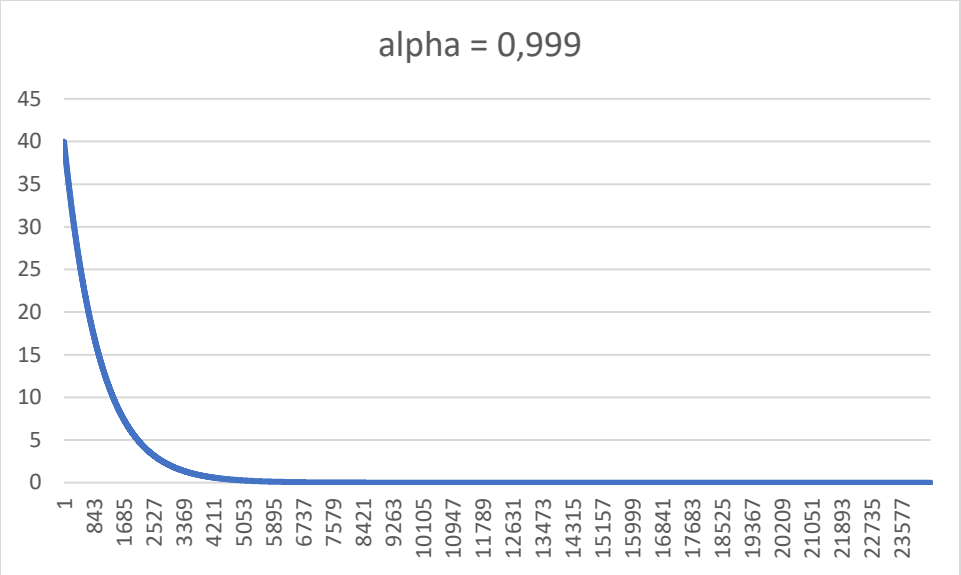
Program vytvorí 20 náhodných susedov z okolia a z nich nájde toho s najkratšou trasou, ktorý nie je v zakázanom liste. Potom ho pridá do zakázaného listu. Ak sa presiahla najväčšia povolená dĺžka zakázaného listu, tak z neho vymaže prvok na prvom mieste, teda ten najstarší. Potom ho porovná so zatiaľ najlepším nájdením riešením a ak zistí že je lepší, tak ho aktualizuje. Tento postup program zopakuje, dokým sa n-krát nenájde lepšie riešenie. Zistil som, že program dokáže optimalizovať riešenie aj pre 40 miest, ak je n rovné 300.

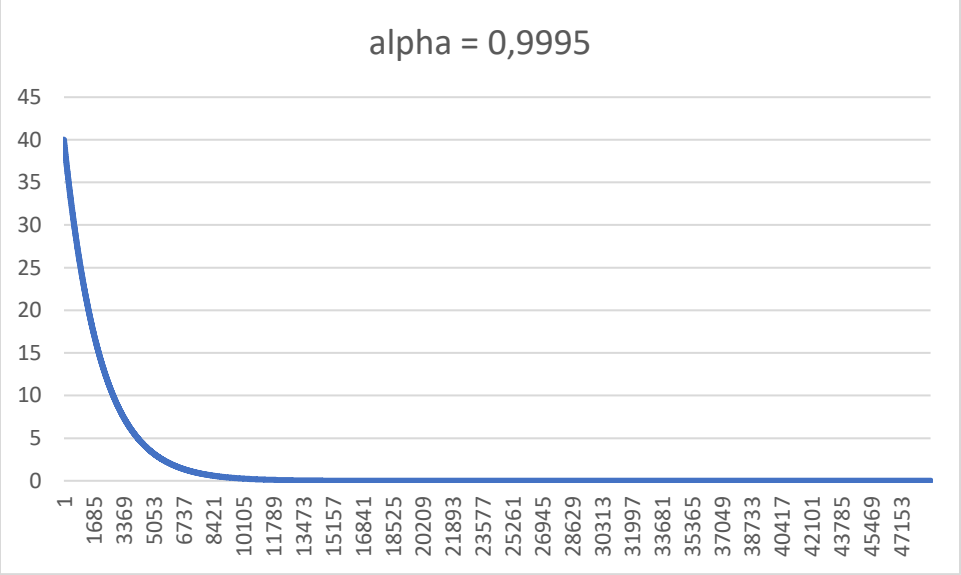
Simulované žíhanie (simulated annealing)

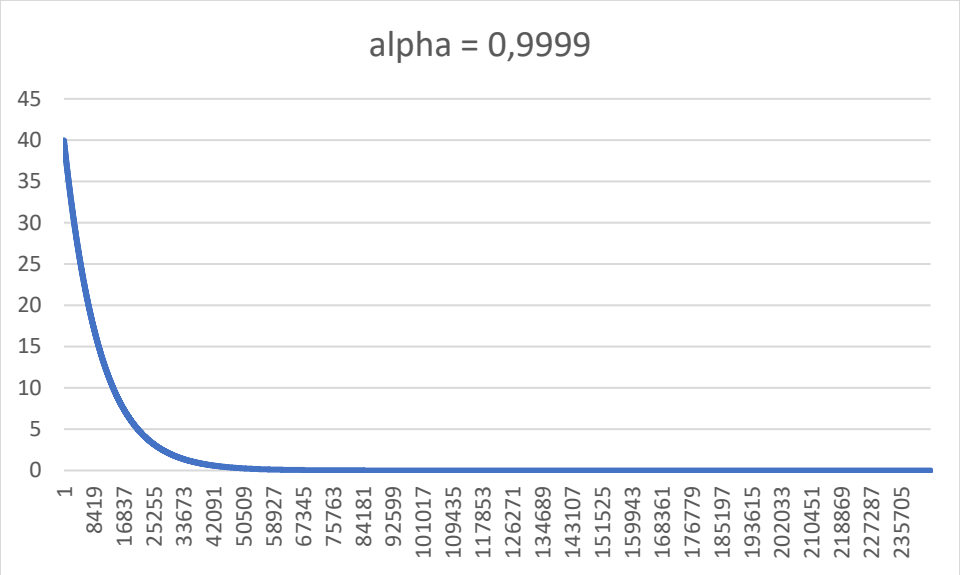
Program začína v určenej teplote, väčšinou medzi 10 a 80 stupňov a trvá, dokým sa neschladí na teplotu 0.0000001 stupňov. Teplotu znižuje tak, že ju násobí premennou alpha, ktorá je o niečo menšia ako 1 a vzniká tak neustále znižujúca sa geometrická postupnosť. Program si vytvorí náhodného suseda a ak je lepší, tak doň určite prejde. Potom ho porovná so zatiaľ najlepším nájdeným riešením a ak zistí, že je lepší, tak ho aktualizuje. Ak je horší, tak použije vzorec $e^{\frac{-|cost1-cost2|}{temp}}$ na zistenie pravdepodobnosti, že doň prejde. *Cost1* a *cost2* sú ceny trás, *temp* je aktuálna teplota. Ak je vypočítaná pravdepodobnosť väčšia ako náhodne vygenerovane číslo od 0 do 1, tak doň prejde.

Vplyv veľkosti premennej alpha na počet opakovania programu:




alpha = 0,999
45
40
35
30
25
20
15
10
5
0
1
843
1685
2527
3369
4211
5053
5895
6737
7579
8421
9263
10105
10947
11789
12631
13473
14315
15157
15999
16841
17683
18525
19367
20209
21051
21893
22735
23577


alpha = 0,9995
45
40
35
30
25
20
15
10
5
0
1
1685
3369
5053
6737
8421
10105
11789
13473
15157
16841
18525
20209
21893
23577
25261
26945
28629
30313
31997
33681
35365
37049
38733
40417
42101
43785
45469
47153



Krivka je síce vždy rovnaká, ale mení sa počet opakovaní programu.

Ak sa alpha = 0,99, tak sa program bude 2429-krát opakovať.

Ak sa alpha = 0,999, tak sa program bude 24401-krát opakovať.

Ak sa alpha = 0,9995, tak sa program bude 48816-krát opakovať.

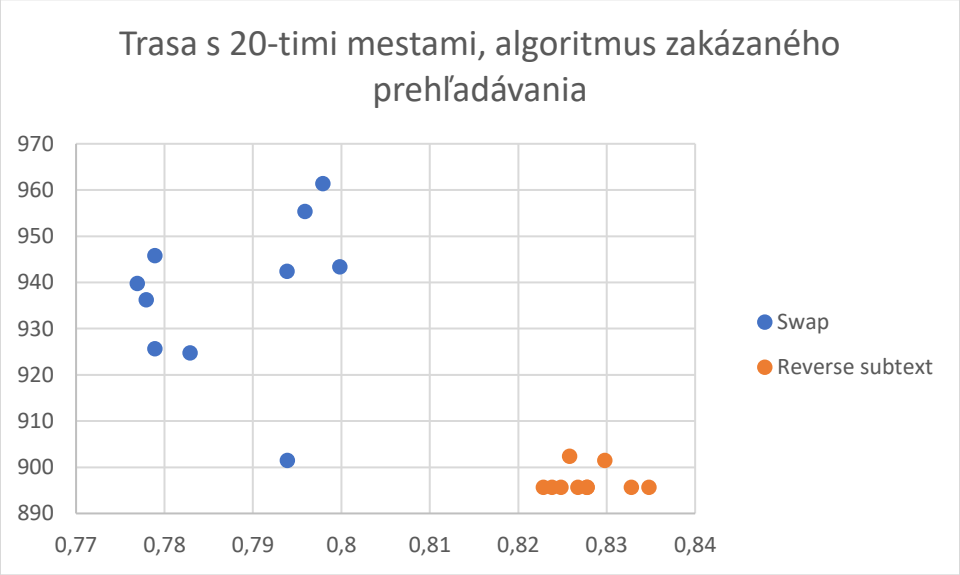
Ak sa alpha = 0,9999, tak sa program bude 244111-krát opakovať.

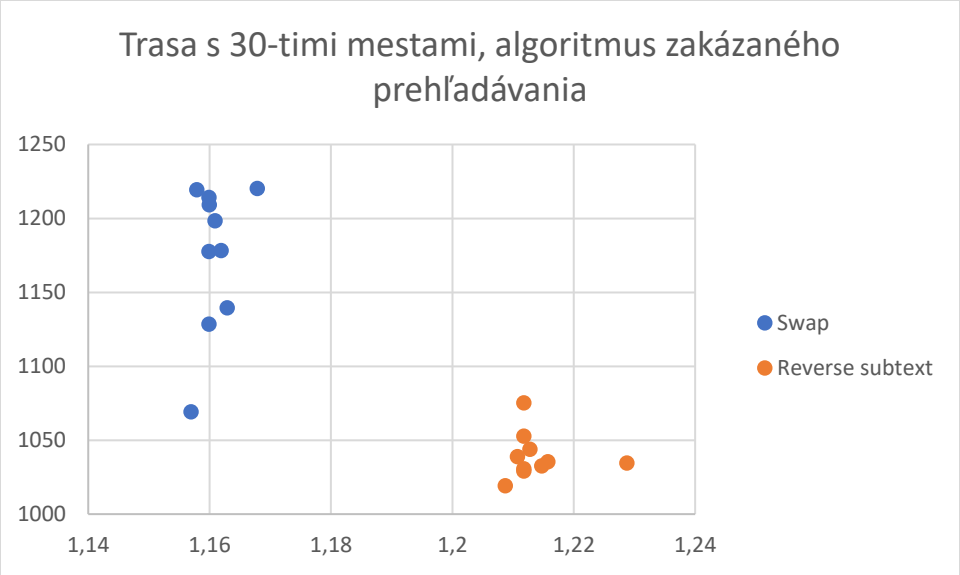
Testovanie

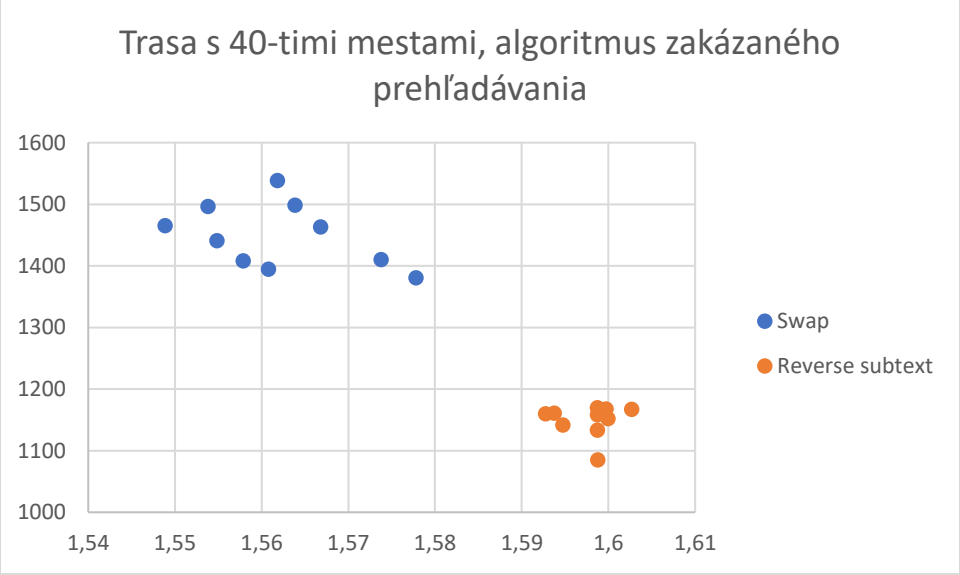
Vykonal som viacero rôznych testov pre oba algoritmy.

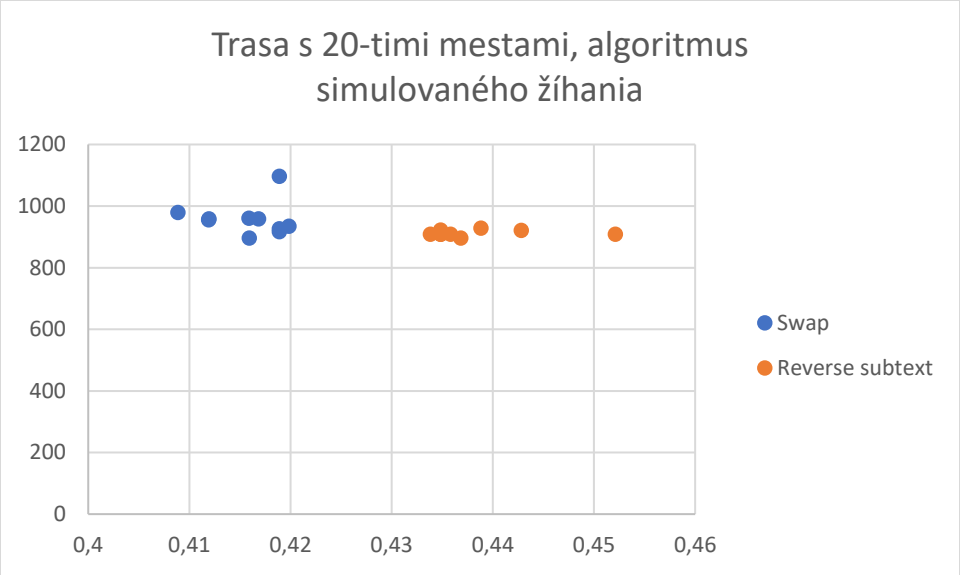
Výmena dvoch náhodných miest vs. metóda otočenia podreťazca

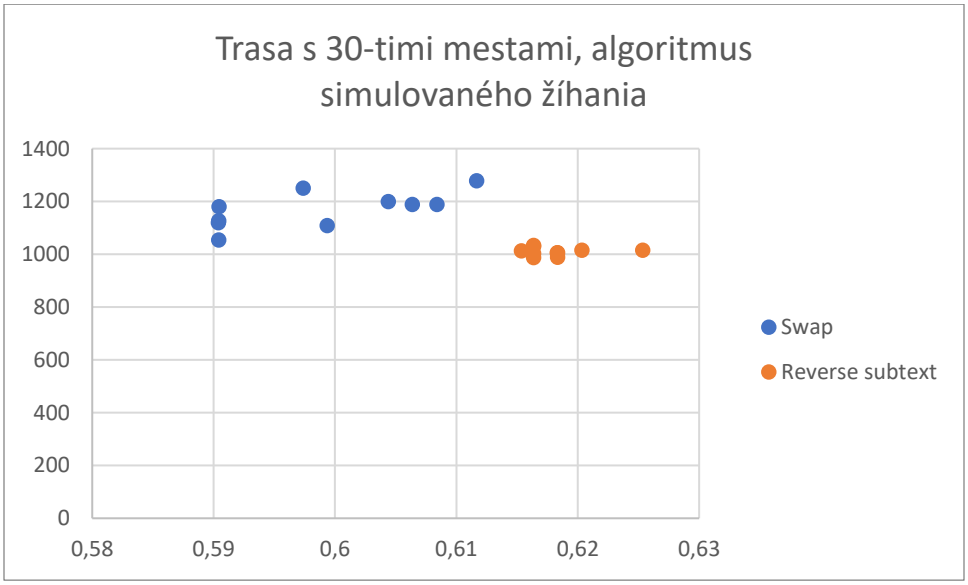
Najprv som testoval nájdenie susedov metódou výmeny dvoch náhodných miest a metódou otočenia podreťazca. Oba algoritmy som testoval tak, že som pri počte miest 20, 30 a 40, našiel 10 riešení metódou výmeny dvoch náhodných miest a 10 riešení metódou otočenia podreťazca.

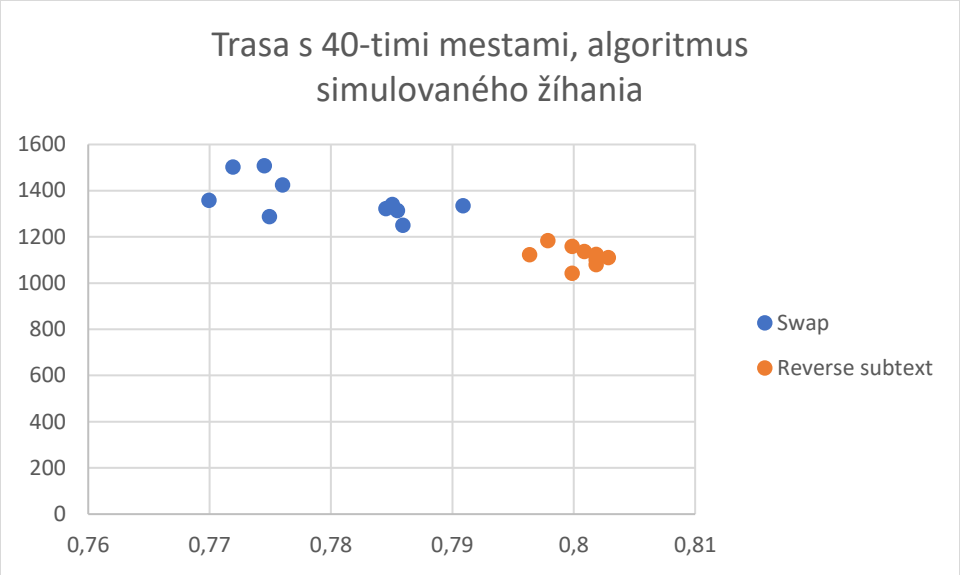



Trasa s 30-timi mestami, algoritmus zakázaného prehľadávania
1250
1200
1150
1100
1050
1000
1,14
1,16
1,18
1,2
1,22
1,24
Swap
Reverse subtext


Trasa s 40-timi mestami, algoritmus zakázaného prehľadávania
1600
1500
1400
1300
1200
1100
1000
1,54
1,55
1,56
1,57
1,58
1,59
1,6
1,61
Swap
Reverse subtext


Trasa s 20-timi mestami, algoritmus simulovaného žíhania
1200
1000
800
600
400
200
0
0,4
0,41
0,42
0,43
0,44
0,45
0,46
Swap
Reverse subtext


Trasa s 30-timi mestami, algoritmus simulovaného žíhania
1400
1200
1000
800
600
400
200
0
0,58
0,59
0,6
0,61
0,62
0,63
Swap
Reverse subtext

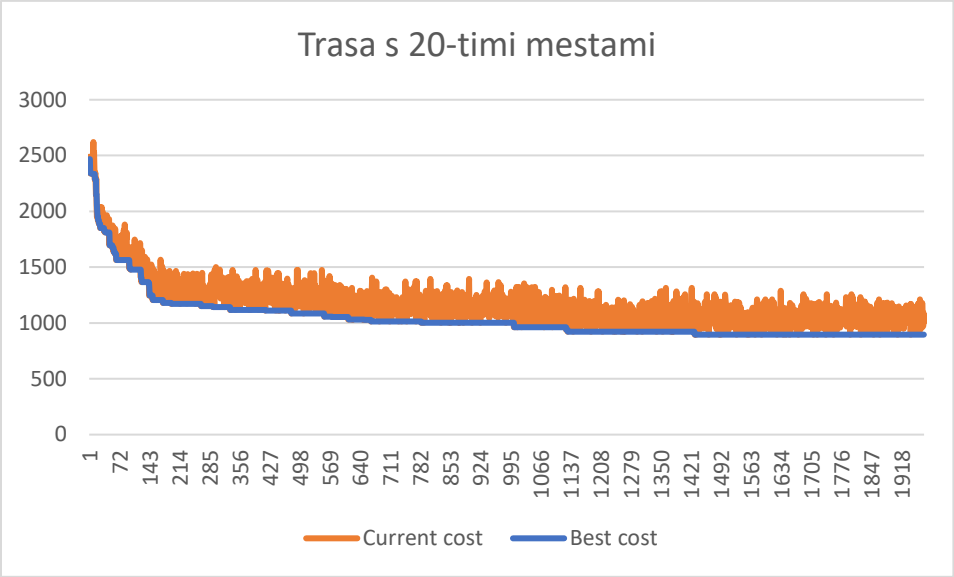


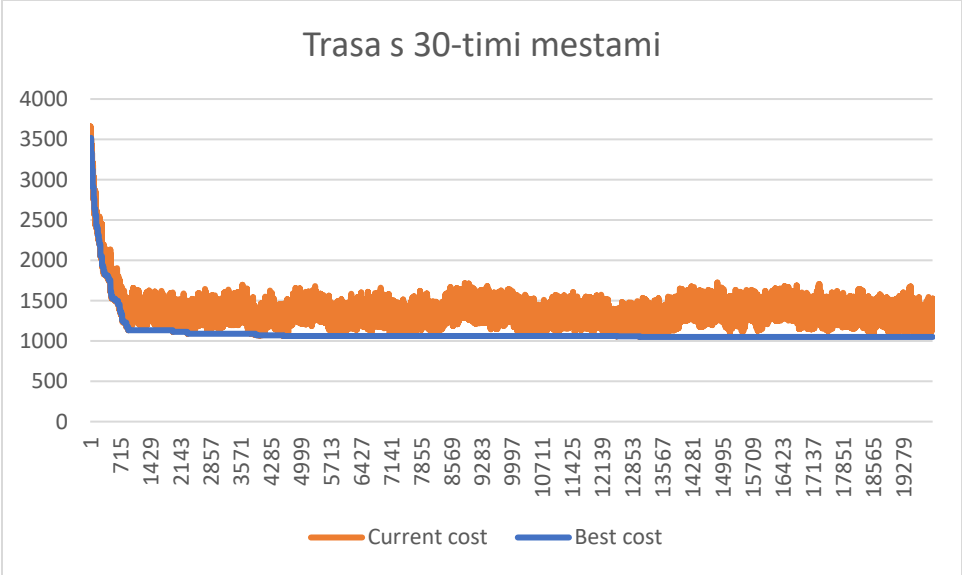
Zistil som, že metóda výmeny dvoch náhodných miest je rýchlejšia ale metódy otočenia podreťazca dokáže nájsť konzistentne lepšie riešenia.

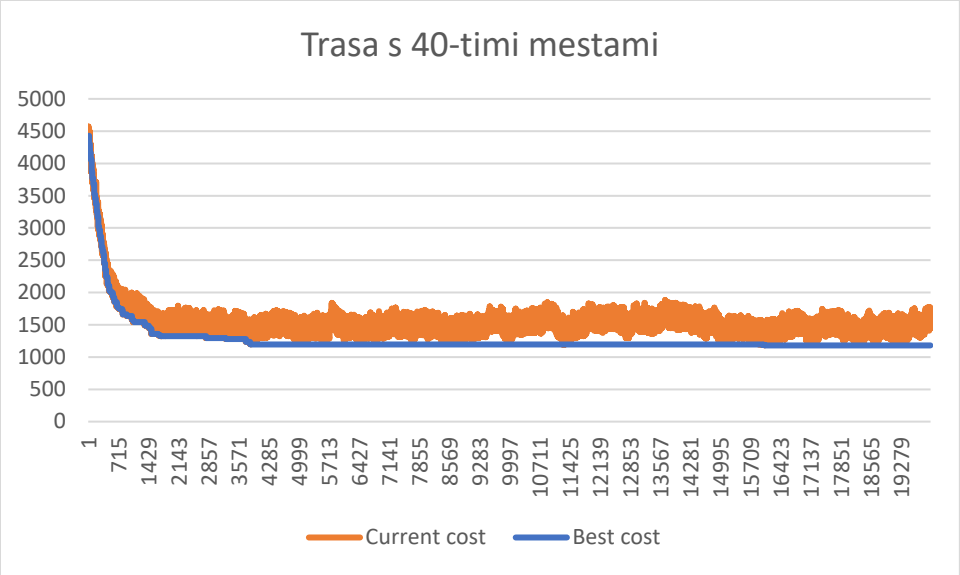
Najlepšie nájdené riešenie vs. aktuálne riešenie

Porovnával som najlepšie nájdené riešenie a aktuálne riešenie pri použití oboch algoritmov.

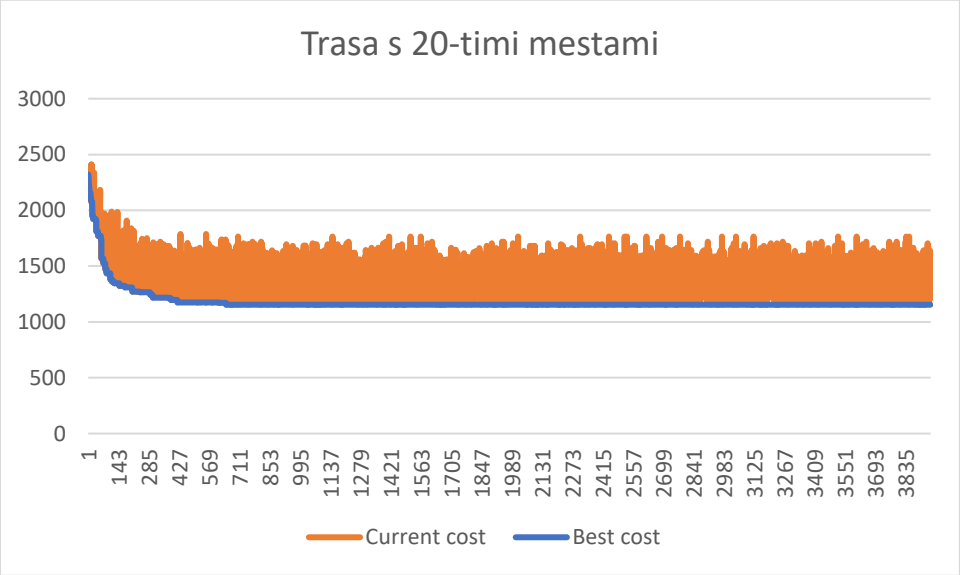
Algoritmus zakázaného prehľadávania

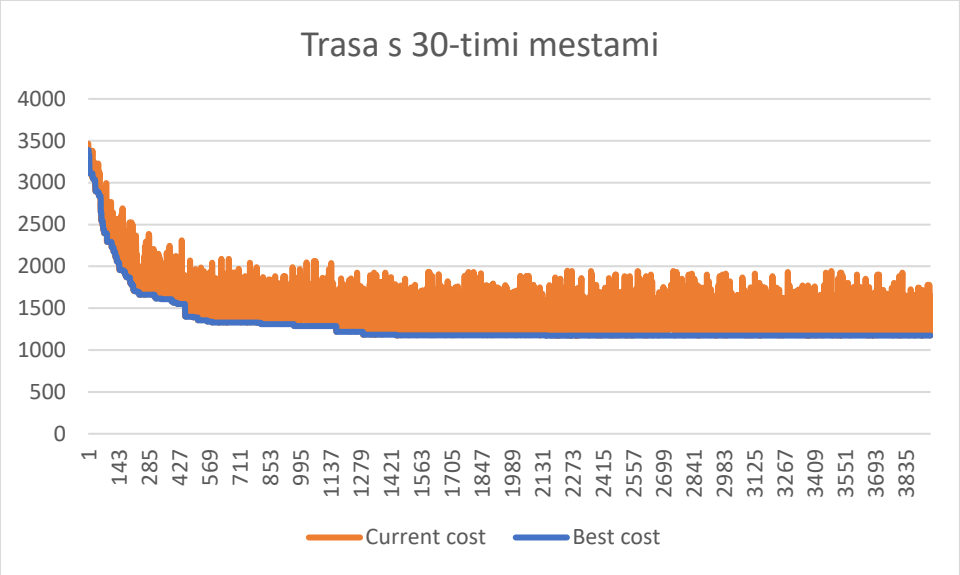


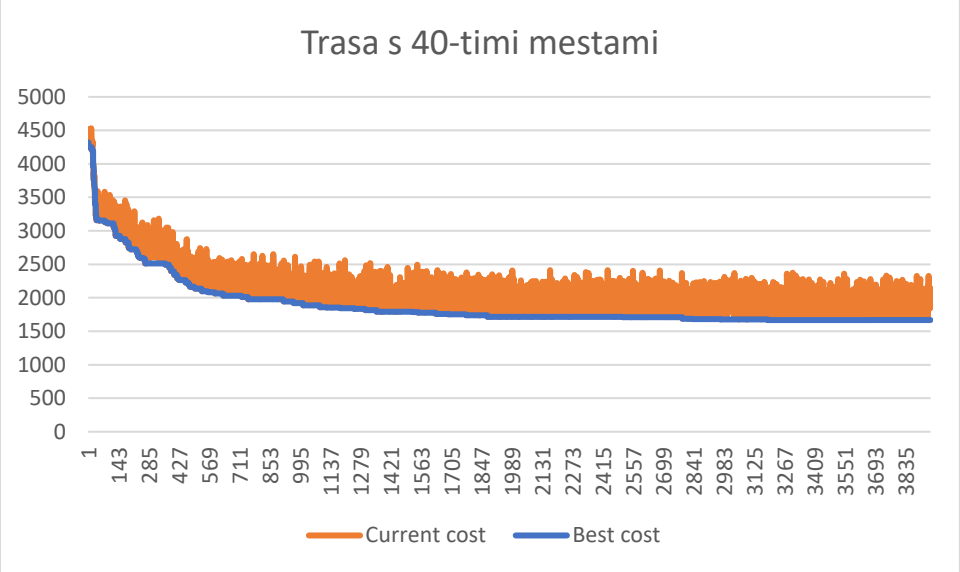




Algoritmus simulovaného žíhania



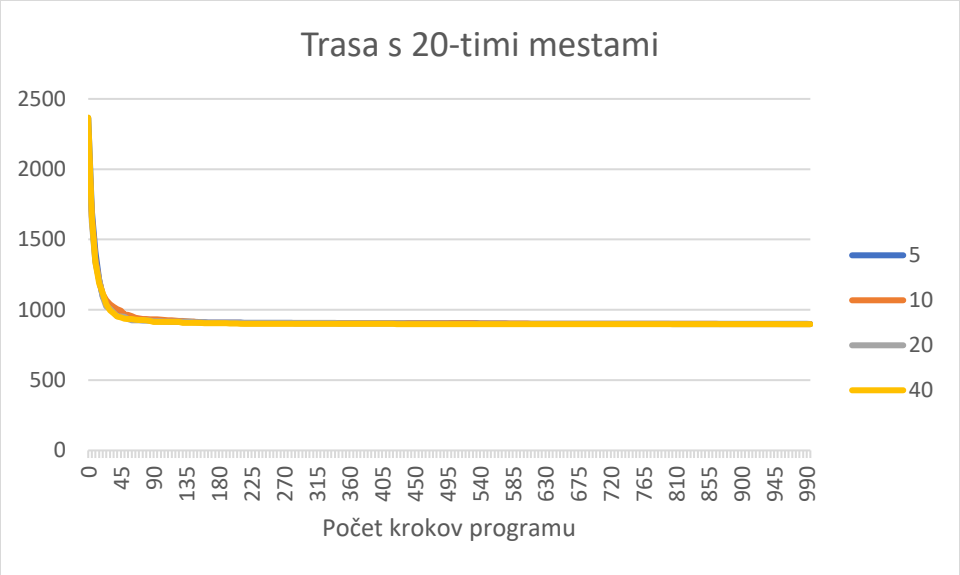


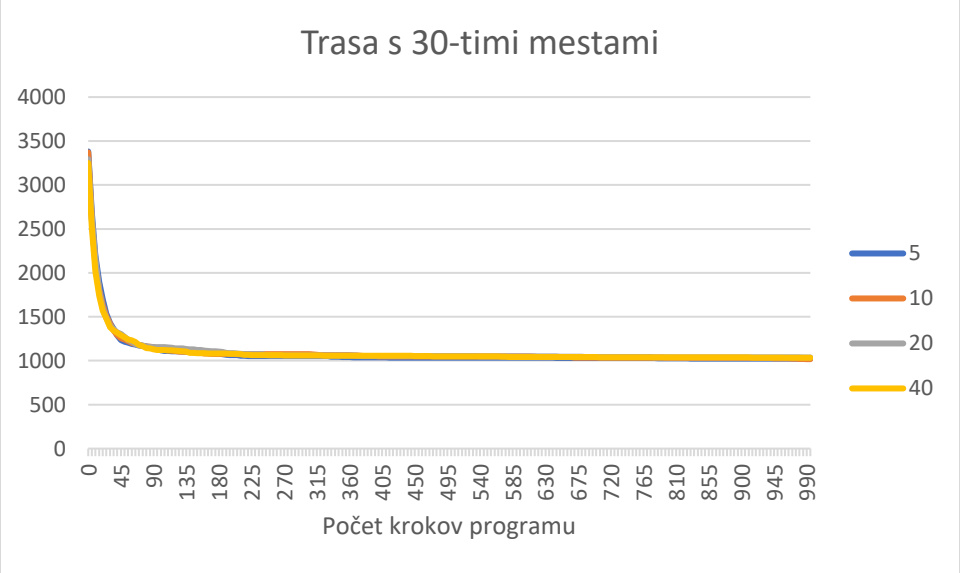


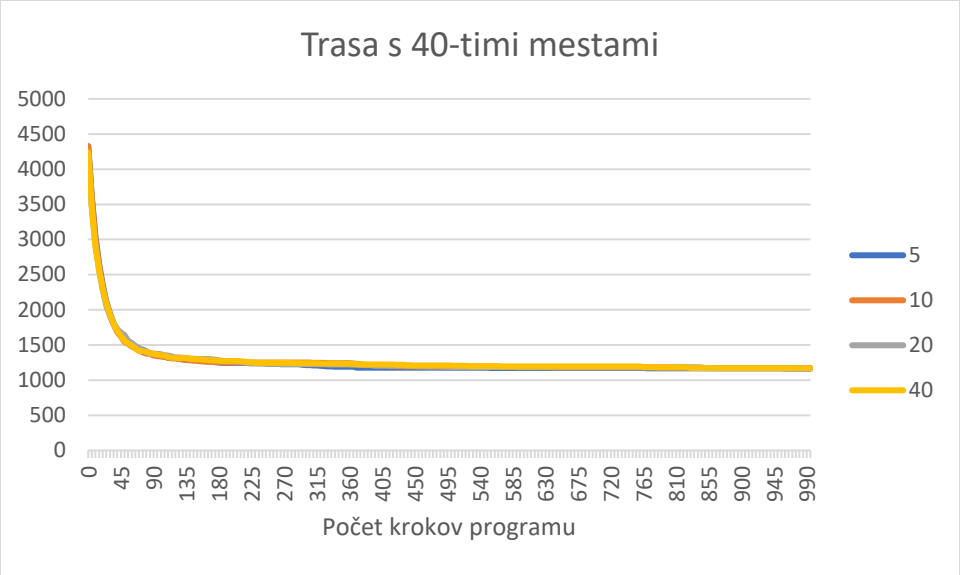
Z testovania vyplýva, že pri oboch algoritmoch zvyšovanie počtu miest zvyšuje počet iterácií algoritmov na nájdenie optimalizovaného riešenia.

Vplyv veľkosti zakázaného listu na priebeh hľadania najkratšej trasy

Pri testovaní som spriemeroval 50 behov programu pre štyri vybrané veľkosti zakázaného listu.



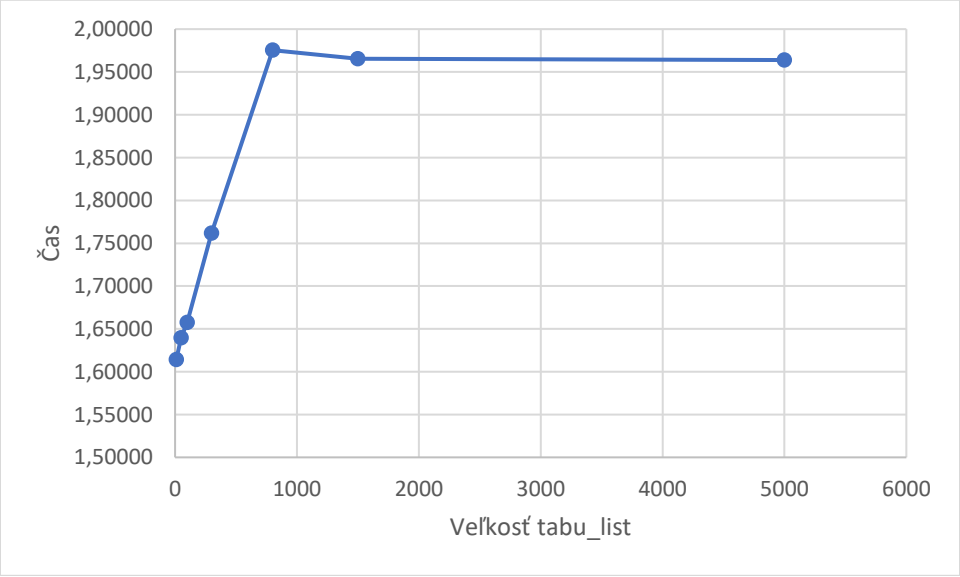




Z testovania vyplýva, že zmena veľkosti zakázaného listu nemá vplyv na priebeh hľadania najkratšej trasy. Testoval som aj väčšie veľkosti zakázaného listu a nenašiel som žiadne zlepšenie.

Vplyv veľkosti zakázaného listu na čas behu programu

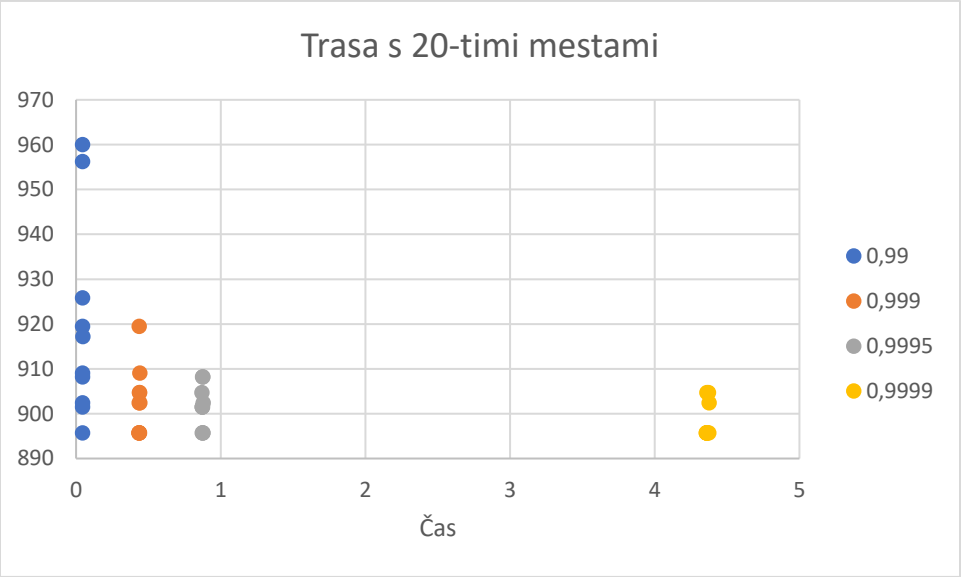
Pri testovaní som spriemeroval 50 behov programu pre štyri vybrané veľkosti zakázaného listu.

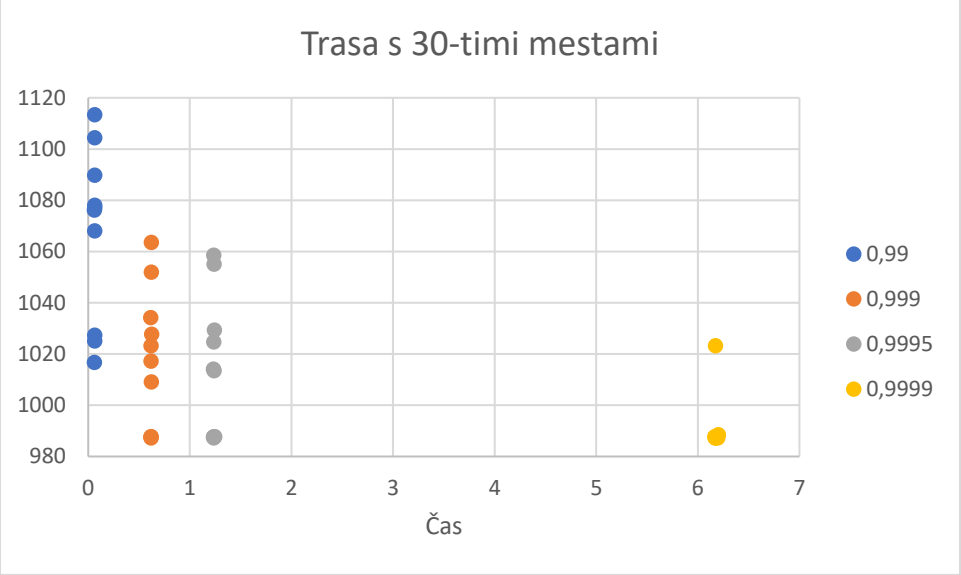


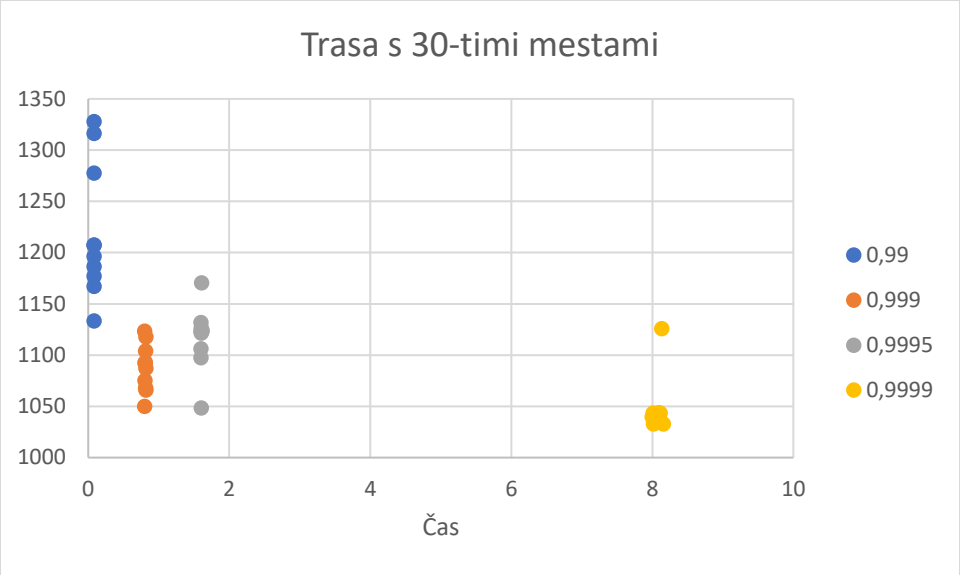
Zvyšovanie maximálnej veľkosti zakázaného listu zvyšuje aj celkový čas riešenia, až pokým nie je väčšia, ako najväčšia veľkosť zakázaného listu, ktorú program reálne dosiahne.

Vplyv rozvrhu zmeny pravdepodobnosti výberu horšieho nasledovníka na výsledný čas programu a najlepšie nájdené riešenie

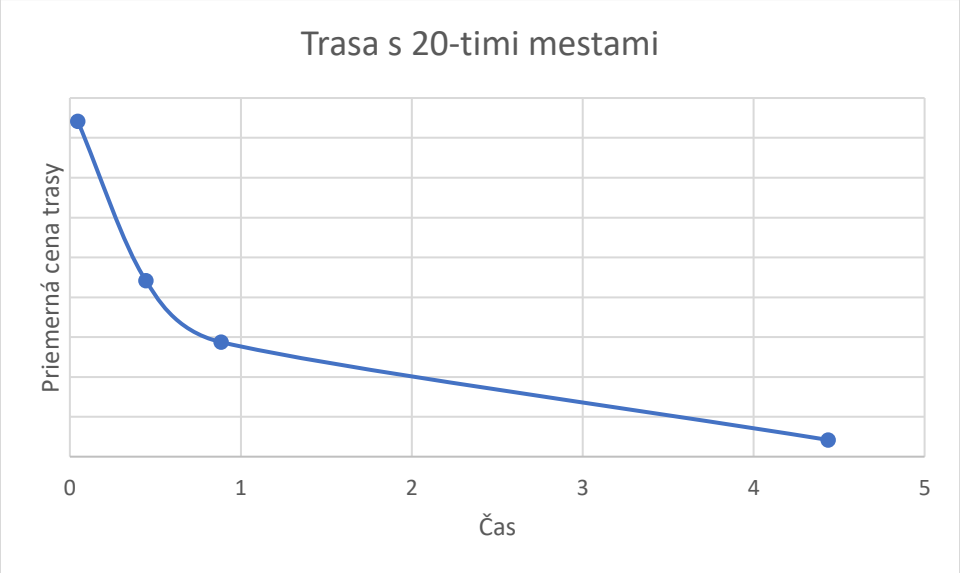
Na graf som umiestnil 10 náhodných riešení pre 4 rôzne hodnoty alpha.

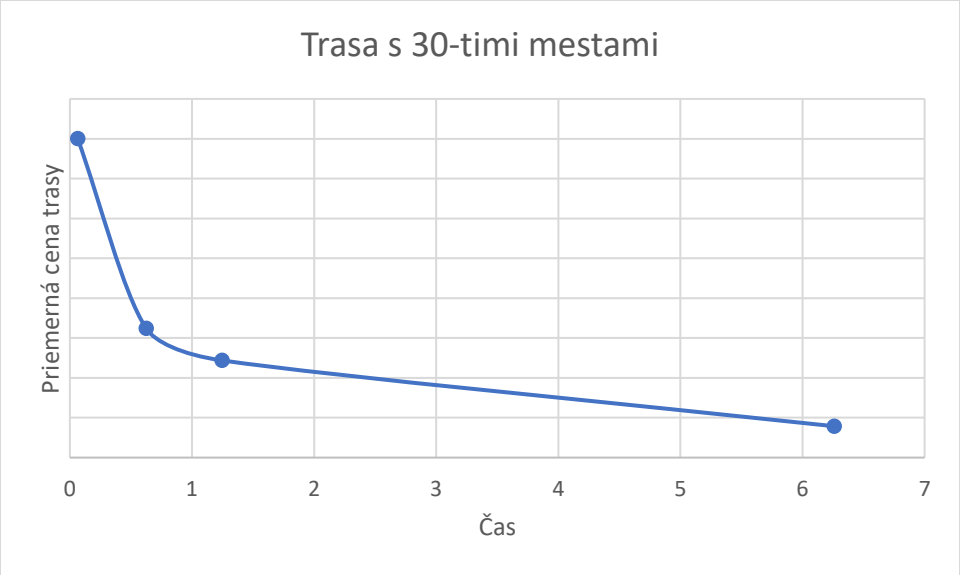


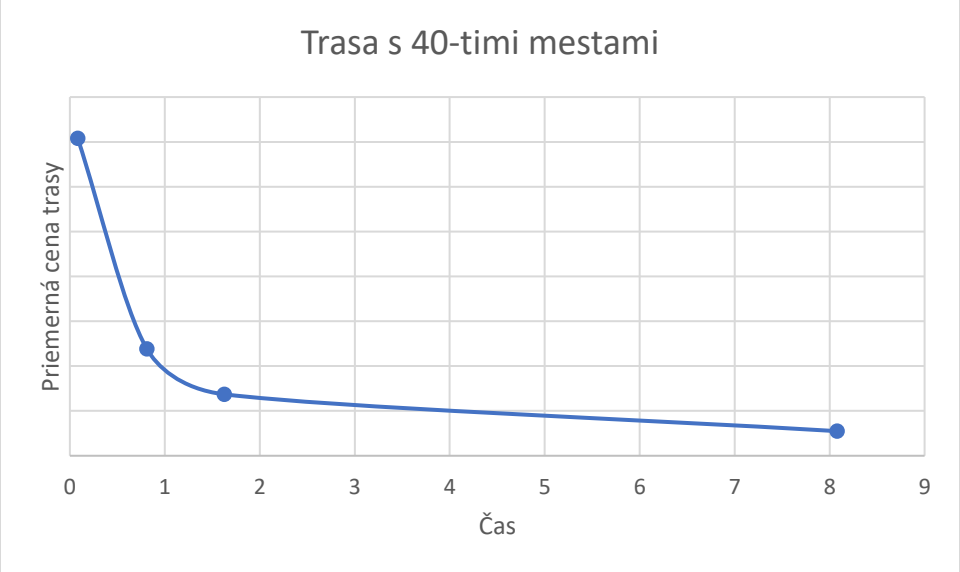




Na graf som umiestnil priemer 50 riešení pre 4 rôzne hodnoty alpha.







Zistil som, že zvyšujúca sa hodnota alpha, zvyšuje celkový čas programu ale priemerne nachádza lepšie riešenia. Ideálna hodnota alpha je v tomto prípade okolo 0,9995.

Zhodnotenie

Z testovania som zistil:

1. Metóda výmeny dvoch náhodných miest je rýchlejšia ako metódy otočenia podreťazca, ale väčšinou nájde horšie riešenia.
2. Pri oboch algoritmoch zvyšovanie počtu miest, zvyšuje počet iterácií algoritmov na nájdenie optimalizovaného riešenia.
3. Zmena veľkosti zakázaného listu nemá vplyv na priebeh hľadania najkratšej trasy, ale zvyšuje celkový čas riešenia, až pokým nie je väčšia ako najväčšia veľkosť zakázaného listu, ktorú program reálne dosiahne.
4. Zvyšujúca sa hodnota alpha zvyšuje celkový čas programu, ale priemerne nachádza lepšie riešenia. Ideálna hodnota alpha je v tomto prípade okolo 0,9995.

Testovať by sa dalo v zakázanom prehľadávaní aj napríklad vyberanie z väčšieho počtu susedov, čo v tomto prípade bolo vždy 20. Tieto riešenia sa dajú zlepšovať ešte rôznymi možnosťami, napríklad miesto toho, aby algoritmy začínali nad náhodným vektorom, tak by mohli začínať s vektorom vytvoreným hladovým algoritmom. Alebo miesto neustáleho počítania vzdialenosti medzi dvomi mestami Pytagorovou vetou, by sa mohla na začiatku vytvoriť matica vzdialeností.