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

Umelá inteligencia

Zadanie 3 – Problém obchodného cestujúceho

Zakázané prehľadávanie (Tabu search)

Lucia Murzová
Ing. Ivan Kapustík - Streda 13:00
20.11.2021

Lucia Murzová
Streda 13:00

Obsah

Lucia Murzová
Streda 13:00

1. Problém obchodného cestujúceho

Travelling salesman problem (TSP) alebo aj problém obchodného cestujúceho je jedna z najznámejších optimalizačných úloh. V úlohe je daných N miest, pričom je možné sa dostať z každého mesta do každého ďalšieho. Cena cesty medzi dvoma mestami zodpovedá Euklidovej vzdialenosti – vypočíta sa pomocou Pytagorovej vety. Obchodný cestujúci sa snaží nájsť najkratšiu možnú cestu tak, aby každé mesto navštívil práve raz a na konci sa vrátil do mesta, v ktorom svoju cestu začal. V našom zadaní budeme pracovať s 20 – 40 mestami.

Lucia Murzová
Streda 13:00

2. Tabu search – zakázané prehľadávanie

Tabu search alebo aj zakázané prehľadávanie je algoritmus, ktorý využíva na hľadanie riešenia v priestore možných stavov lokálne vylepšovanie (optimalizáciu). Znamená to, že z aktuálneho stavu si vytvorí nasledovníkov a presunie sa do takého, ktorý má lepšie ohodnotenie.

Ak neexistuje nasledovník s lepším ohodnotením a nenašli sme dostatočne dobré riešenie, nachádzame sa v lokálnom extréme a je potrebné sa z neho dostať. Algoritmus preto vyberie horšieho nasledovníka a zároveň uloží aktuálny stav do zoznamu zakázaných stavov – tabu listu. Zoznam zakázaných stavov je nevyhnutný, aby sa algoritmus z horšieho nasledovníka nevrátil opäť do lokálneho extrému a nevytvoril nekonečný cyklus. Zoznam je zároveň krátky, aby jeho kontrola netrvala veľmi dlho. Keď presiahne maximálnu stanovenú veľkosť a je potrebné do neho vložiť nový stav, najstarší sa zahodí.

3. Opis riešenia

Súbor const.py

V tomto súbore sú zadané parametre, ktoré sa v priebehu programu nemenia, no sú veľmi dôležité na jeho fungovanie a pre testovanie je potrebné ich často meniť. Nastavené sú tu hodnoty: počet miest, šírka a výška plochy, na ktorej sa budú generovať mestá, veľkosť tabu listu, počet iterácií, ktoré môže program urobiť a vstupný súbor, z ktorého sú načítavané polohy miest v prípade nastavenia ich načítania namiesto vygenerovania náhodných polôh.

Globálne premenné

- Polohy_miest: [ ] – pole s načítanými / vygenerovanými vstupnými polohami miest
- Najkratšia_cesta_lokálne: Cesta – lokálne najlepšia nájdená cesta
- Najkratšia_cesta_globálne: Cesta – globálne najlepšia nájdená cesta
- Číslo_iterácie: int – počet prejdených iterácií
- Tabu_list: [ ] – zoznam zakázaných stavov
- Susedia: [ ] – pole vygenerovaných susedov aktuálne najlepšej cesty
- Najlepšie_lokálne: [ ] – pole s lokálne najlepšími cestami z každej iterácie
- Najlepšie_globálne: [ ] – pole s globálne najlepšími cestami z každej iterácie
- Iterácie: [ ] – pole s číslami iterácií

Reprezentácia stavov

Trieda Cesta – každý stav je reprezentovaný ako objekt Cesta, ktorá má uložený zoznam miest v poradí, v ktorom boli navštívene a tiež dĺžku cesty pri takomto usporiadaní miest. Trieda Cesta má v sebe implementovanú aj metódu pre vypočítanie dĺžky svojej cesty.

Generovanie susedných stavov

Pri každej iterácii dochádza ku generovaniu susedných stavov tak, že každé mesto v zozname aktuálne najlepšieho stavu sa vymení s každým ďalším mestom. Napríklad zo stavu 1 2 3 4 5 vzniknú:

2 1 3 4 5	1 3 2 4 5	1 2 4 3 5
3 2 1 4 5	1 4 3 2 5	1 2 5 4 3
4 2 3 1 5	1 5 3 4 2	
5 2 3 4 1		1 2 3 5 4

Týmto spôsobom generovania susedných stavov vždy vznikne (POČET MIEST – 1) +(POČET MIEST – 2)+...+(1) = **((POČET MIEST – 1) * POČET MIEST)/2** nových susedných stavov.

Pri generovaní neprebieha kontrola s predošlými stavmi, takže tieto sa môžu opakovať (napr. z 2 1 3 4 5 môže znovu vzniknúť 1 2 3 4 5). Táto kontrola nie je potrebná nakoľko ak stav nie je najlepší, tak ho program nevyberie a ak je najlepší, už bol pravdepodobne vybraný a nachádza sa v tabu liste a teda ho program znovu nevyberie. Takýto program by bol navyše aj časovo menej efektívny.

Vstup

Vstup predstavujú načítané alebo vygenerované náhodné polohy miest (na osi X a Y) reprezentované celými číslami od 0 po nastavený rozmer plochy. Polohy miest sa načítajú do globálneho poľa polohy_miest, z ktorého sa neskôr čítajú pre vypočítanie dĺžok jednotlivých ciest.

Pri načítaní miest zo súboru program po prečítaní súboru kontroluje dĺžku načítaného zoznamu pre prípad, že v súbore bolo zadaných menej miest ako bolo potrebné načítať. (Nastavenie pre načítanie alebo vygenerovanie je možné nastaviť vo funkcii main odkomentovaním / zakomentovaním daných funkcií.)

Výstup

Ako výstup je vypísaná globálne najkratšia nájdená cesta – poradia miest a dĺžka tejto cesty. Program na konci tiež vykreslí graf s najlepšími lokálne nájdenými cestami a zároveň s globálne najlepšou cestou v jednotlivých iteráciách. Na vykreslenie tohto grafu sa využívajú globálne premenné najlepšie_lokálne [ ], najlepšie_globálne [ ] a iterácie [ ], do ktorých sa v každej iterácii tieto hodnoty vkladajú.

Následne program tiež vykreslí mapu s číslami miest podľa poradia ako boli načítané / vygenerované a tiež čiary vykresľujúce globálne najkratšiu nájdenú cestu.

Algoritmus

Program na začiatku vygeneruje alebo načíta zadané polohy miest, podľa uvedeného počtu miest a zadanej výšky a šírky mapy v súbore const.py. Následne funkcia novy_susedia() v cykle generuje nových susedov lokálne najlepšej cesty a kontroluje lokálne / globálne maximá až pokiaľ nedosiahne zadané maximum počtu iterácií.

Kontrola susedov prebieha vo funkcii nájdi_lokálne_najlepšiu, kde prechádza všetkých vytvorených susedov a hľadá najkratšiu a druhú najkratšiu cestu z tých, ktoré nie sú zapísané v tabu liste. Na začiatku vždy nastaví najkratšiu na lokálne najlepšiu a druhú najkratšiu na hodnotu MAX_INT. Pokiaľ je najkratšia cesta lepšia ako aktuálne lokálna najlepšia, vylepší lokálne najlepšiu a skontroluje, či táto cesta nie je aj globálne najlepšia.

Ak je najkratšia cesta aj po prejdení všetkých susedov rovnaká ako lokálne najkratšia (teda tá, na ktorú bola na začiatku nastavená), nepodarilo sa v susedoch nájsť kratšiu cestu, ktorá nie je v tabu liste. V tomto prípade pridá momentálne lokálne najlepšiu cestu do tabu listu a ako lokálne najlepšiu nastaví druhú najlepšiu. Následne program skontroluje, či nebola presiahnutá nastavená dĺžka tabu listu a prípadne odstráni „najstaršiu“ cestu v zozname – na pozícii 0. Na konci funkcie nájdi_lokálne_najlepšiu() tiež program pridá do globálnych polí najlepsie_lokalne[] a najlepsie_globalne[] dané hodnoty pre aktuálnu iteráciu.

Lucia Murzová
Streda 13:00

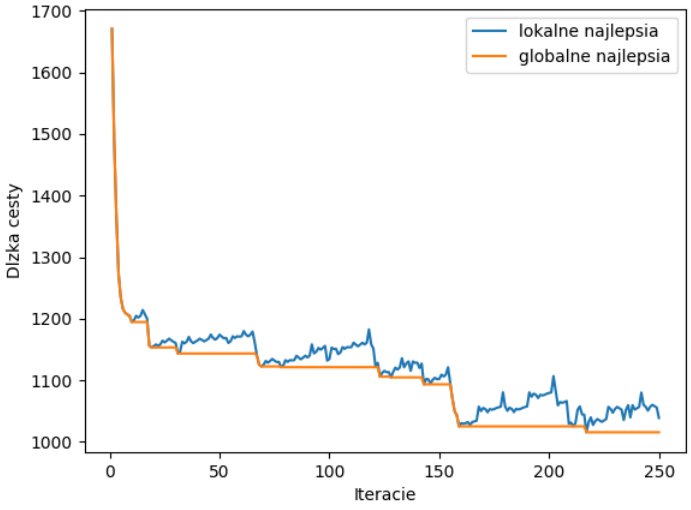
4. Testovanie

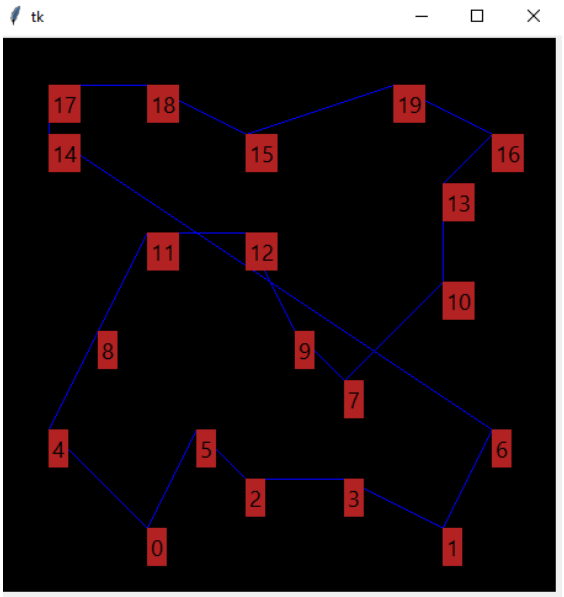
Program som testovala pre rôzny počet miest – 20/30/40 a taktiež pre každý z týchto počtov aj pre rozdielne veľkosti tabu listu – od 10 do 70. Podľa testov sa ukázalo, že dĺžka tabu listu je dôležitá hlavne pri nižšom počte miest – 20, pri 30 a 40 mestách podľa testov vyzerá že dĺžka tabu listu nezohráva veľkú úlohu.

Nižšie sú uvedené testovacie scenáre pre 6 vstupných súborov s uvedeným počtom miest, rozmerom plochy a pôvodnou dĺžkou cesty. K jednotlivým súborom sú uvedené rôzne scenáre testovania – rôzne dĺžky tabu listu, dĺžku výslednej cesty a tiež výstup programu vo forme grafu a nakreslenej najkratšej cesty.

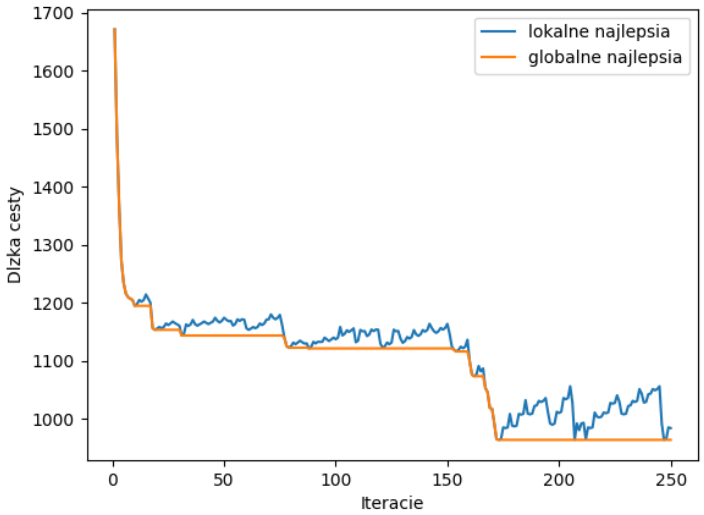
1. Testovací súbor vstup.txt (vzorové riešenie zo zadania) – 20 miest, rozmer plochy 200x200, pôvodná dĺžka cesty = 1 964

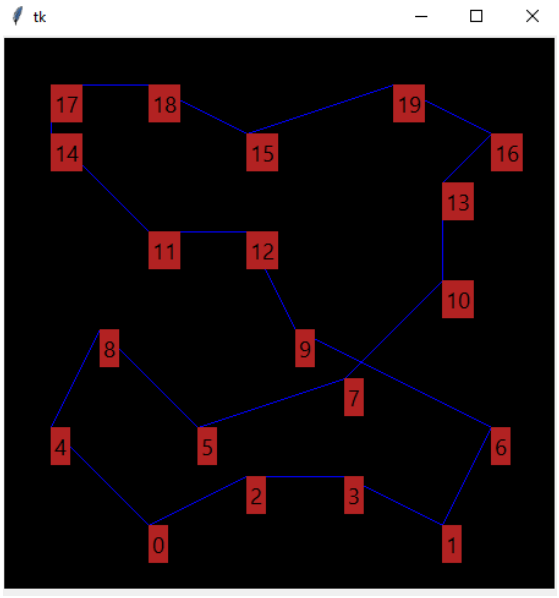
- veľkosť tabu 30, cesta 1 015





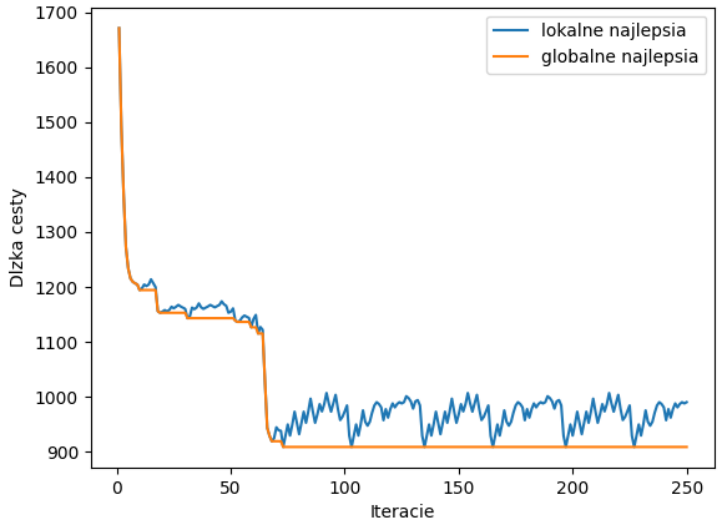
- veľkosť tabu 20 / 25, cesta 963

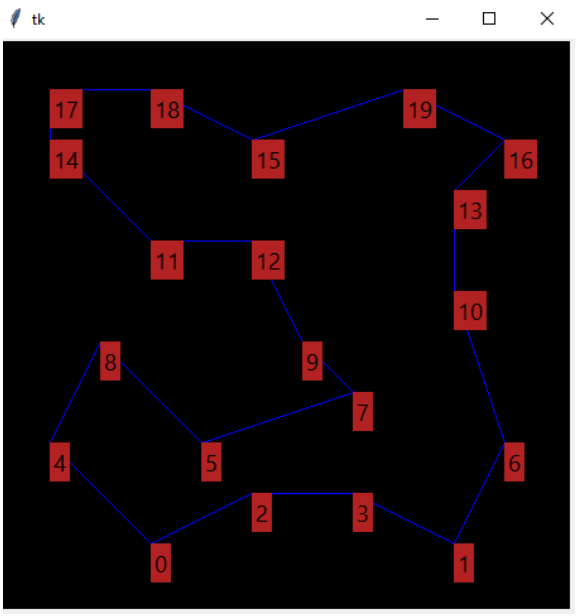




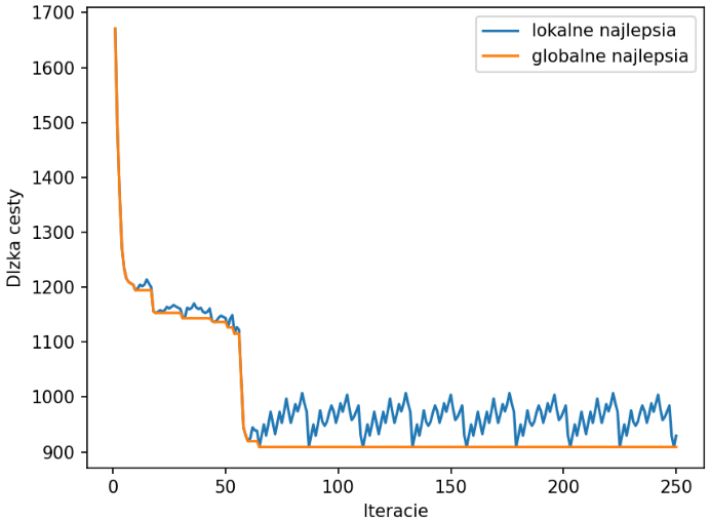
Lucia Murzová
Streda 13:00

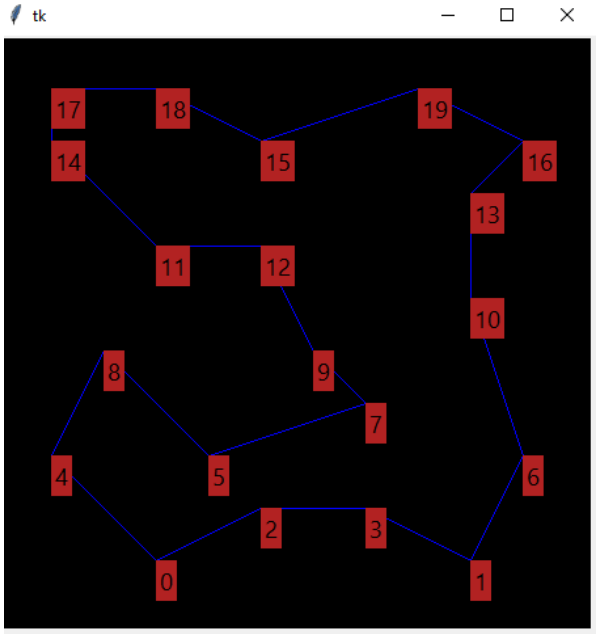
- tabu 15, cesta = 909





- tabu 10, cesta = 909

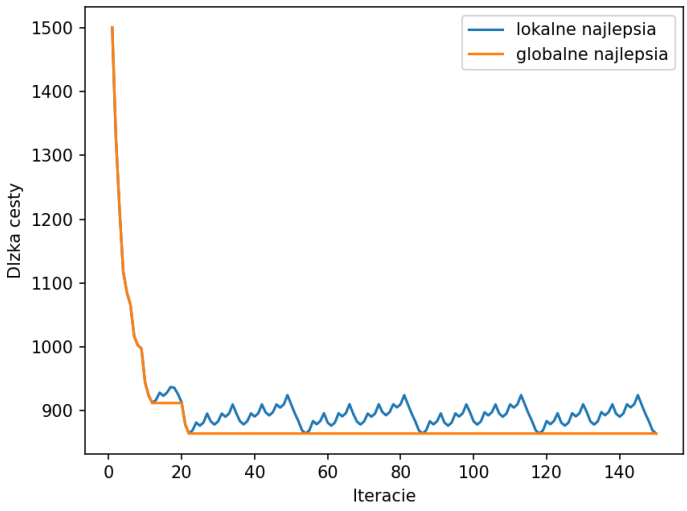


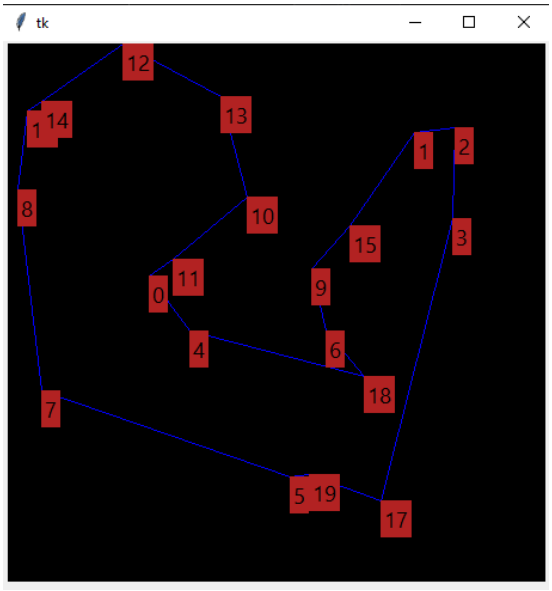


Lucia Murzová
Streda 13:00

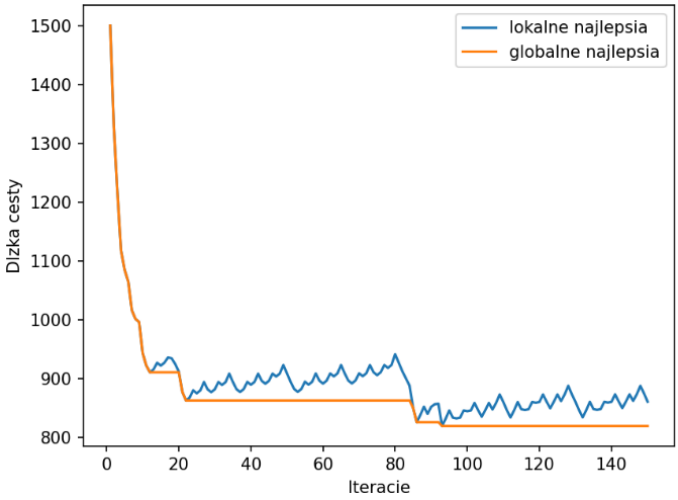
2. Testovací súbor vstup1.txt, 20 miest, veľkosť 200x200, pôvodná dĺžka cesty 1 774

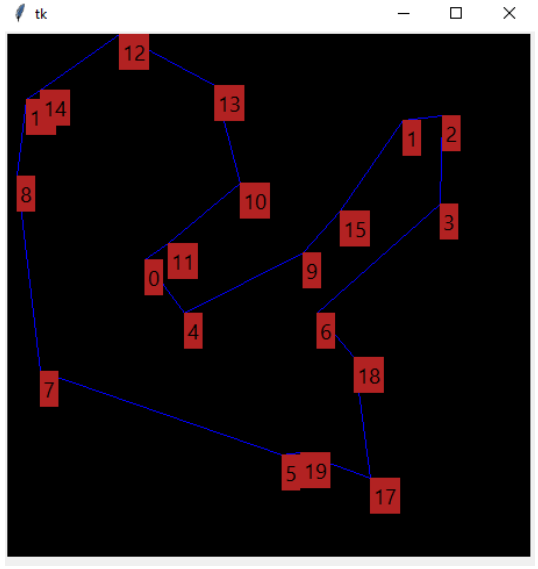
- dĺžka tabu 10 a 15, dĺžka cesty = 863



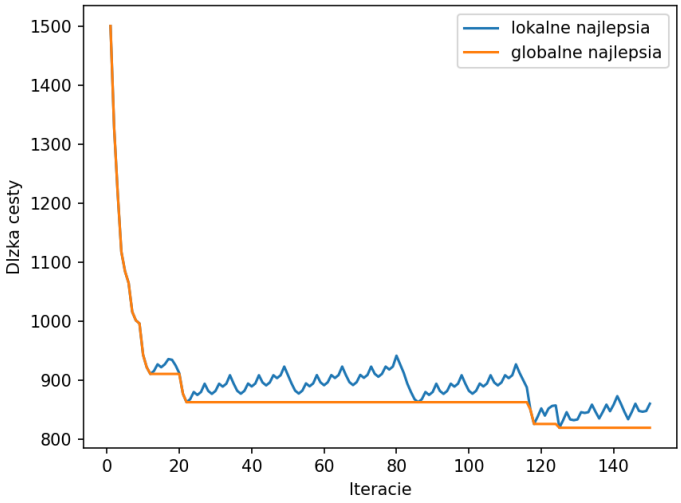


- dĺžka tabu 20 / 22 / 25, dĺžka cesty = 819





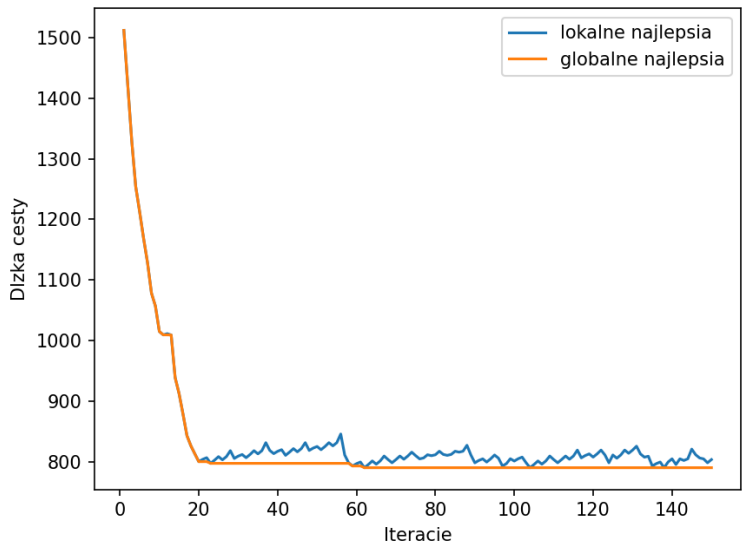
- dĺžka tabu 30 / 35 / 40 / 45 / 50 / 55, dĺžka cesty = 819 (rovnaká ako pri predošlom, našlo ale riešenie neskôr)

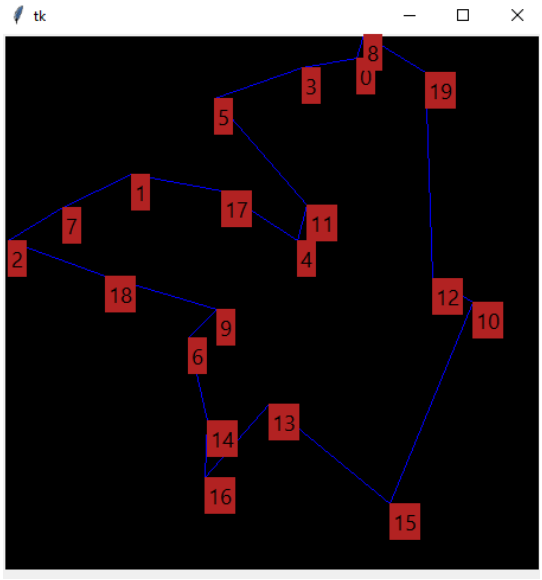


Lucia Murzová
Streda 13:00

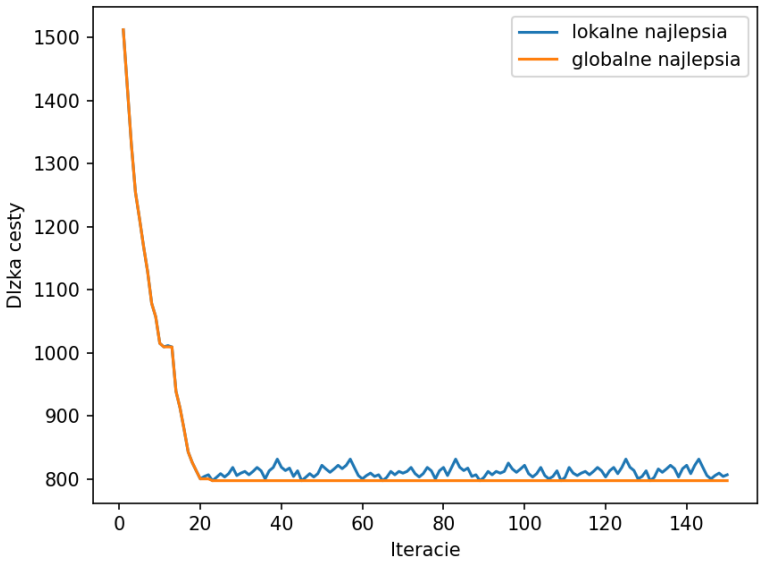
3. Testovací súbor vstup2.txt, 20 miest, veľkosť 200x200, pôvodná dĺžka 1 807

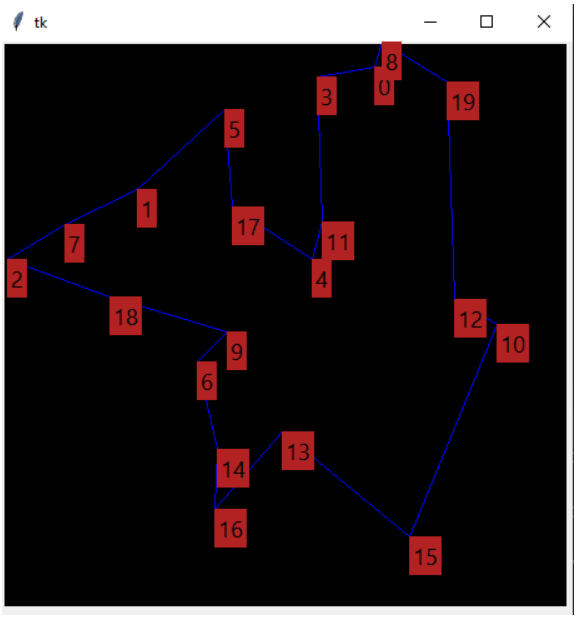
- dĺžka tabu 15 / 20 / 25 / 35 / 55, dĺžka cesty = 790





- dĺžka tabu 10, dĺžka cesty = 797

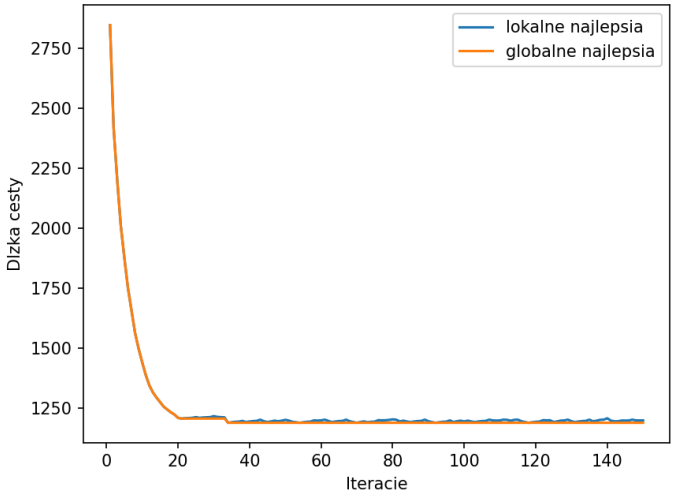


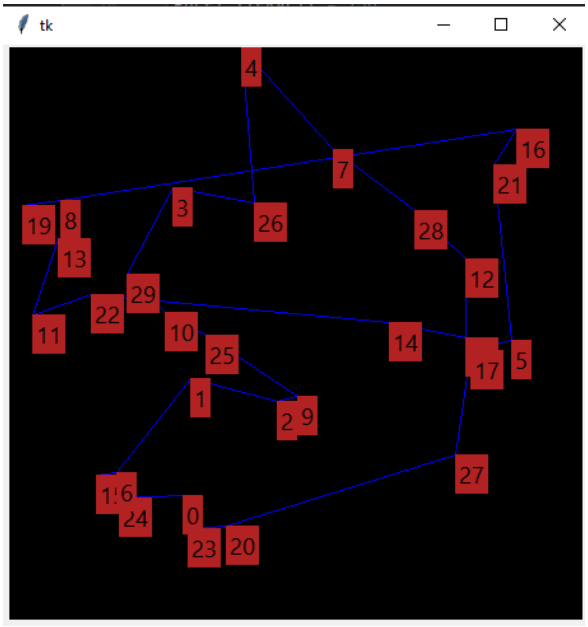


Lucia Murzová
Streda 13:00

4. Testovací súbor vstup3.txt, 30 miest, plocha 200x200, pôvodná dĺžka cesty = 3 315

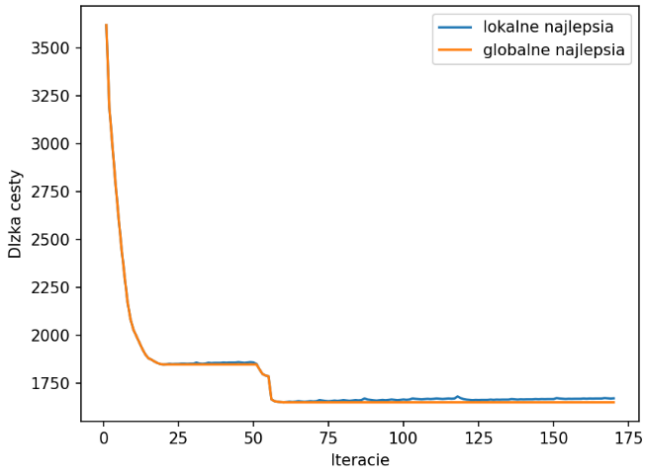
- dĺžka tabu 10 / 20 / 25 / 30 / 40 / 45, dĺžka cesty = 1 189

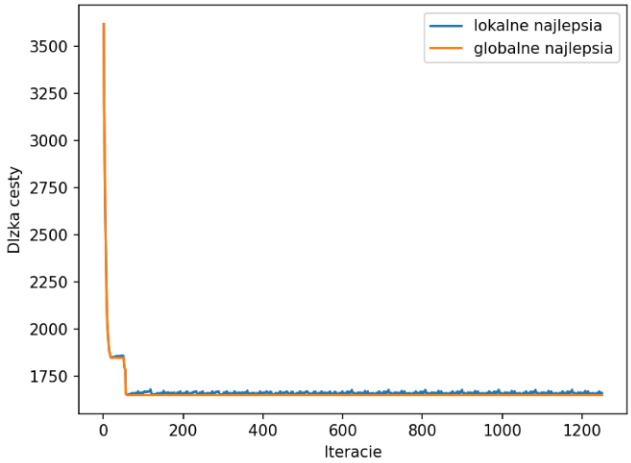


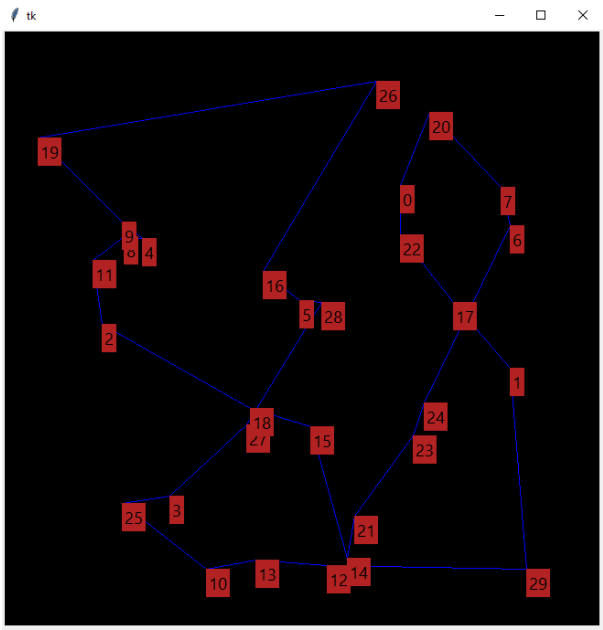


5. Testovací súbor vstup4.txt, 30 miest, plocha 300x300, pôvodná dĺžka cesty 4 145

- dĺžka tabu 10 / 15 / 20 / 25 / 27 / 30 / 50, dĺžka cesty = 1 648



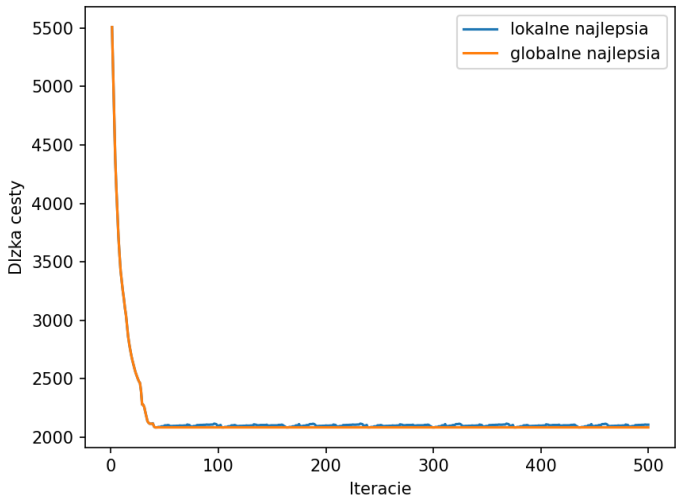


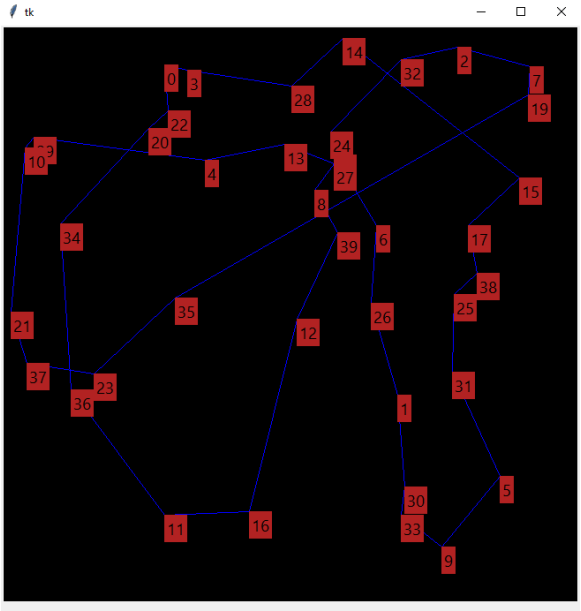


Lucia Murzová
Streda 13:00

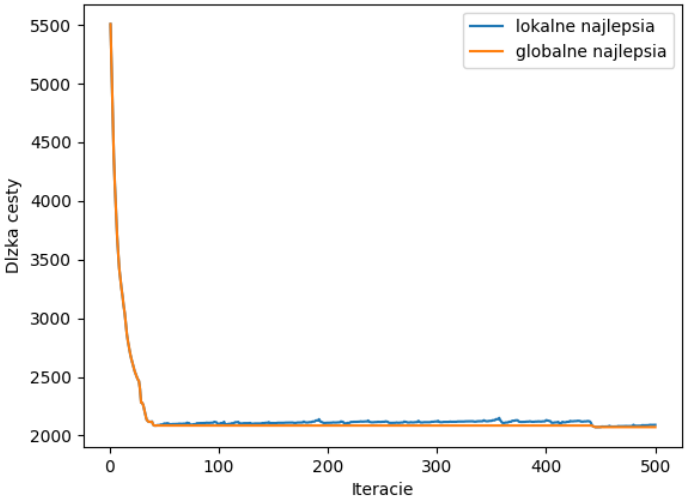
6. Testovací súbor vstup5.txt, 40 miest, plocha 300x300, pôvodná dĺžka 6 139

- dĺžka tabu 20 / 30, dĺžka cesty = 2 084





- dĺžka tabu 50, dĺžka cesty = 2071 (nájdená až približne v 450 iterácii)



5. Zhodnotenie testovania a riešenia

Podľa vykonaných testov sa ukázalo, že dĺžka tabu listu zohráva veľkú úlohu, no hlavne pri menšom počte miest – 20. V prvom prípade bolo najlepšie riešenie nájdené pri dĺžke tabu listu 10 a 15 približne okolo 75 iterácie. Pri dlhšom tabu liste program hľadal najkratšiu cestu značne pomalšie a aj tak nenašiel tak dobré riešenie ako s kratším tabu listom.

V druhom prípade bol scenár trochu iný. Pri dĺžke tabu 10 a 15 našiel program najlepšie riešenie takmer okamžite – do 20 iterácie, avšak išlo len o lokálne minimum. Pri tabu dĺžke 20, 22 a 25 sa podarilo nájsť optimálnejšiu cestu okolo 90 iterácie. Pri tabu 30, 35, 40, 45, 50 a 55 bola cesta rovnaká ako pri menších tabu listoch, našlo ju však až o približne 30 iterácií neskôr.

Pri 30 a 40 mestách sú výsledky takmer identické pri každej otestovanej dĺžke tabu listu. Môže to byť spôsobené tým, že mestá už sú vygenerované moc na kope a najoptimálnejšiu cesty nájde takmer ihneď – do 50 iterácie. Ďalšie prehľadávanie sa pohybuje vo veľmi podobných číslach a nevie nájsť optimálnejšiu cestu.

Pri 20 mestách podľa vykreslených máp program pravdepodobne nájde najkratšiu možnú cestu. Pri 30 a viac mestách riešenie nie je úplne najlepšie no je postačujúco optimálne.

Priemerné percentuálne vylepšenie finálnej cesty oproti pôvodnej ceste pri rôznych počtoch miest:

- Pri 20 mestách bolo priemerné vylepšenie o 55%
- Pri 30 mestách bolo priemerné vylepšenie o 65%
- Pri 40 mestách bolo priemerné vylepšenie o 67%