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

# ZADANIE 3(c) - Problém obchodného cestujúceho

Martin Nemec
FIIT STU
Cvičenie: Štvrtok 12:00
17.11.2021

# 1 Zadanie úlohy

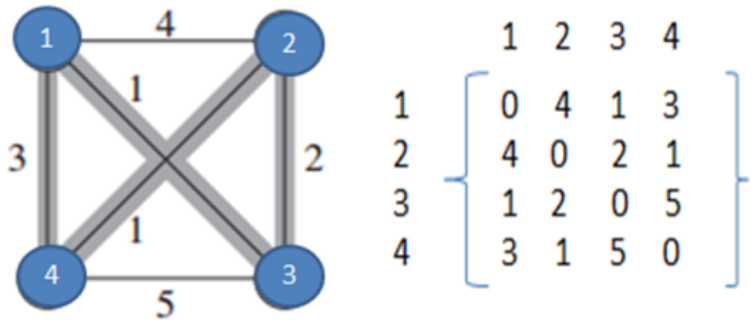
Je daných aspoň 20 miest (20 – 40) a každé má určené súradnice ako celé čísla X a Y. Tieto súradnice sú náhodne vygenerované. (Rozmer mapy môže byť napríklad 200 * 200 km.) Cena cesty medzi dvoma mestami zodpovedá Euklidovej vzdialenosti – vypočíta sa pomocou Pytagorovej vety. Celková dĺžka trasy je 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.

# 2 Opis riešenia

Na začiatku je zvolený počet miest. Na základe tohto počtu pomocou funkcie generate_places() sa náhodne vygenerujú tieto miesta na mape 200x200, pričom súradnice x a y každého bodu sú v intervale <1;200>.

Následne sa zavolá funkcia create_matrix_of_distances(), ktorá do dvojrozmerného poľa uloží maticu susedností každého vrchola s každým. Daná susednosť je vypočítaná pomocou Pytagorovej vety.




Príklad matice susednosti

Vytvorí sa náhodná permutácia miest, ktorá je ako currentSolution, čiže aktuálne riešenie. Následne sa volajú riešenia pomocou danných algoritmov - zakázané hľadanie a simulované žíhanie.

## 2.1 Tabu search

Na začiatku ako bestSolution sa zoberie currentSolution. Následuje cyklus, ktorý má napr. 10 000 iterácií.

Následne v cykle sa do premennej next_solutions vložia permutácie aktuálneho riešenia vymieňaním vždy dvoch náhodných vrcholov. Počet týchto permutácií som zvolil ako počet miest krát 4. K tomuto počtu susedov som došiel na základe skúšania a testovania - ak tých susedov bolo menej, horšie boli výsledné cesty a ak tých susedov bolo viac napr. 100, tak sa predlžoval čas. Pri 20 miest sa z aktuálneho riešenia vždy vytvorí 80 susedov, pričom vždy sa odstrňujú duplikáty.

Následne si toto pole permutácií zoradím podľa ceny cesty od najmenšej.

Následuje cyklus, ktorý zoberie zoradené pole a prehľadáva od najmenšej či sa nenachádza v tabu liste. Ak sa nachádza zoberie ďalšieho možného, ktorý sa nenachádza v tabu liste. Ak takéhoto nájde zoberie ho ako currentSolution a pridá ho do tabu listu.

Ak dĺžka tabu listu presiahne nejakú hodnotu napr. 50, vymaže sa prvý údaj z tabu listu.

```
if tabu_list:
    for i in new:
        if i not in tabu_list:
            currentSolution = i
            tabu_list.append(i)
            if len(tabu_list) > 50:
                tabu_list.remove(tabu_list[0])
            break
else:
    tabu_list.append(new[0])
    currentSolution = new[0]
```

Ak dané riešenie je lacnejšie ako najlepšie riešenie, tak sa nastaví aj za najlepšie riešenie. Cyklus sa opakuje.

## 2.2 Simulated annealing

Simulované žihanie je na podobnom princípe ako tabu search, s malými úpravami. Na začiatku sa nastaví teplota - odporúča sa 20-50 (prednáška 7). Začína cyklus pokiaľ teplota nie je 0 alebo menšia. Po vstupe do cyklu sa teplota zmenší o 3 a následuje cyklus s napr. 10 000 iteráciami ako v Tabu searchi.

Hľadanie susedných permutácií je rovnaké ako pri tabu. Avšak teraz sa nezoradia od najmenšej cesty, ale vyberie sa jedna náhodná permutácia. Ak cesta permutácie je menšia ako currentSolution tak za currentSolution sa nastaví táto náhodná permutácia. Ak nie je menšia ako currentSolution, znamená to že toto náhodné riešenie musí byť vačšie. Urobí sa rozdiel náhodnej permutácie od currentSolution. A ak tento rozdiel je menší ako náhodne vygenerované číslo v intervale <0,1> tak to zoberie ako riešenie a nastaví to do currentSolution.

Tým, že sme nezobrali menšie riešenie ako to aktuálne to nevadí, pretože ten rozdiel je veľmi malý. A aj ak ten rozdiel bude menší ako 1, tak musí byť menší aj ako náhodne vygenerované číslo z int. <0,1>. Tým pádom ak aktuálna cesta má hodnotu 1000 a tá horšia cesta bude mať hodnotu 1000,3 je to zanedbateľné.

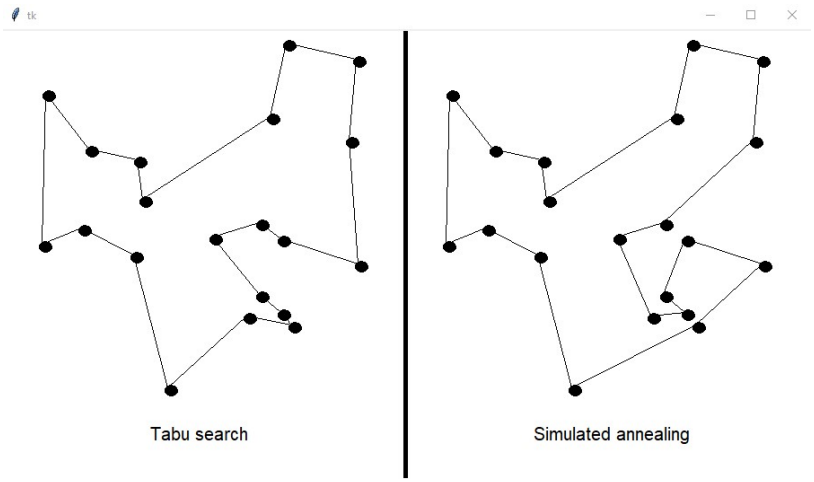
Pri testovaní som zistil, že horšie riešenie zoberie niekedy 0 krát, niekedy 1, a niekedy aj okolo 200 krát

```
if routeLength(matrix_of_distances, randomSol) <= routeLength(matrix_of_distances, currentSolution):
    currentSolution = randomSol
else:
    delta = routeLength(matrix_of_distances, randomSol) - routeLength(matrix_of_distances, currentSolution)
    u = random.uniform(0, 1)
    if u > delta:
        currentSolution = randomSol
        count += 1
```

Ako pri tabu searchi ak currentSolution má lacnejšiu cestu ako bestSolution tak bestSolution = currentSolution.

# 3 Výstup

Príklad výstupu z programu s vykreslením grafov:




```
[15, 7, 12, 14, 0, 4, 17, 1, 10, 8, 9, 2, 5, 16, 3, 19, 18, 6, 11, 13]
Route cost - tabu search: 825.32395062421
[3, 9, 8, 17, 4, 0, 14, 12, 7, 15, 13, 11, 6, 18, 19, 16, 1, 10, 2, 5]
Route cost - simulated annealing: 901.5663700918052
Time of tabu search: 4.261111736297607s
Time of simulated annealing 24.911064624786377s
```

# 4 Testovanie

**Všetky údaje v tabuľkách sú priemery z 5tich spustení !**

Testoval som rôzne veľkosti tabu listu a rôzne počty iterácií a pri simulovanom žíhaním som testoval pri rôznych teplotách a rôznych iteráciách.

| Legenda | |
|---|---|
| | počet iterácií |
| | dĺžka tabu listu |
| | teplota |

Legenda (modrá - počet iterácií, zelená - dĺžka tabu listu, oranžová - teplota)

## 4.1 Testovanie s 20 mestami

### Tabu search

| TABU SEARCH - 20 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 itercií | 0,428s | 791,8 |
| 50 tabu length | | |
| 5 000 | 2,11 | 770 |
| 50 | | |
| 10 000 | 4,28 | 770 |
| 50 | | |
| 100 000 | 42,56 | 770 |
| 50 | | |
| 10 000 | 4,23 | 770 |
| 100 | | |
| 10 000 | 4,57 | 770 |
| 500 | | |
| 10 000 | 4,63 | 770 |
| 1000 | | |

Pri tabu searchi s 20 mestami som vykonal tieto testy. Môžeme vidieť že pri zvyšovaní iterácií a zvyšovaní veľkosti tabu listu sa priemerný čas predlžuje, pričom dĺžka cesty je vždy rovnaká. To znamená že program nachádzal najlepšie riešenie.

### Simulated annealing

| SIMULATED ANNEALING - 20 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 itercií | 1,948 | 874 |
| 30 | | |
| 5 000 | 9,68 | 878 |
| 30 | | |
| 10 000 | 19,54 | 872 |
| 30 | | |
| 30 000 | 58 | 880 |
| 30 | | |

| SIMULATED ANNEALING - 20 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 | 2,46 | 944 |
| 40 | | |
| 5 000 | 12,42 | 899 |
| 40 | | |
| 10 000 | 24,82 | 844 |
| 40 | | |
| 30 000 | 74,39 | 932 |
| 40 | | |

| SIMULATED ANNEALING - 20 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 | 3,02 | 915 |
| 50 | | |
| 5 000 | 15,21 | 894 |
| 50 | | |
| 10 000 | 30,23 | 907 |
| 50 | | |
| 30 000 | 90,18 | 933 |
| 50 | | |

Pri simulovanom žíhaní môžeme vidieť že priemerné hodnoty dĺžky ciest sú skoro vždy rovnaké. Jemne sa zmenšujú pri zväčšovaní teploty a iterácií. Priemerný čas sa postupne zvyšuje ako sa zvyšovali počty iterácií a teploty

## 4.2 Testovanie so 40 mestami

### Tabu search

| TABU SEARCH - 40 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 itercií | 1,55 | 1118 |
| 50 | | |
| 5 000 | 7,95 | 1114 |
| 50 | | |
| 10 000 | 15,59 | 1052 |
| 50 | | |
| 30 000 | 47,89 | 1062 |
| 50 | | |

| TABU SEARCH - 40 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 iterci | 1,57 | 1124 |
| 200 | | |
| 5 000 | 7,82 | 1084 |
| 200 | | |
| 10 000 | 15,48 | 1099 |
| 200 | | |
| 30 000 | 46,51 | 1057 |
| 200 | | |

| TABU SEARCH - 40 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 itercií | 1,57 | 1100 |
| 1000 | | |
| 5 000 | 8,07 | 1107 |
| 1000 | | |
| 10 000 | 16,082 | 1054 |
| 1000 | | |
| 30 000 | 47,62 | 1045 |
| 1000 | | |

Pri 40 mestách môžeme vidieť že dĺžky ciest sa postupne zmenšujú ako sme pridávali iterácie. Ak porovnávame časy pri rovnakých iteráciach a rôznych veľkostiach tabu listu, môžeme vidieť že pri vyššej dlžke tabu listu sa aj predlžuje čas hľadania. Medzi najoptimálnejšie nastavenie by som zvolil dĺžku tabu listu 200 a počet iterácií 10 000.

### Simulated annealing

| SIMULATED ANNEALING - 40 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 itercií | 6,1 | 1349 |
| 30 | | |
| 5 000 | 30,35 | 1381 |
| 30 | | |
| 10 000 | 60,87 | 1322 |
| 30 | | |

| SIMULATED ANNEALING - 40 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 | 7,742 | 1308 |
| 40 | | |
| 5 000 | 38,83 | 1387 |
| 40 | | |
| 10 000 | 77,39 | 1350 |
| 40 | | |

| SIMULATED ANNEALING - 40 places | | |
|---|---|---|
| | sekundy | dĺžka cesty |
| 1 000 | 9,47 | 1251 |
| 50 | | |
| 5 000 | 47,12 | 1246 |
| 50 | | |
| 10 000 | 95,08 | 1296 |
| 50 | | |

Pri testovaní môžeme vidieť že časy pri 40 mestách už su o dosť väčšie ako pri 20. Pri zvyšovaní iterácií a teploty nie sú výrazné lepšie priemerné cesty. V tomto prípade by som volil len 1000 iterácií a začiatočnú teplotu 50. Aj keď môže niekedy nájsť neoptímálnu cestu ale priemerné výsledky ukazujú že na nájdenie celkom optimálnej dlžky cesty stačí 9,47s.

# 5 Záver

V závere by som zhodnotil že lepšie sa mi podarilo implementovať tabu search, je to efektívnejšie a rýchlejšie.