

# Reševanje problema trgovskega potnika s k-optimalnim in Lin-Kernighanovim algoritmom

Žan Jernejčič in Ines Šilc

Fakulteta za matematiko in fiziko

13. januar 2020

## Definicija problema

Problem trgovskega potnika ozziroma Travelling salesman problem (krajše TSP) je problem, kjer imamo podanih  $n$  mest in razdalje med vsemi (za vsak par mest imamo torej podano, koliko sta si oddaljeni). Zanima nas, ali lahko običemo vsako mesto in se na koncu vrnemo v prvotno mesto. Če označimo  $d_{i,j}$  kot razdaljo med  $i$ -tim in  $j$ -tim mestom, iščemo torej:

$$\min_{\pi \in S_n} \sum_{i=1}^{n-1} d_{\pi(i), \pi(i+1)} + d_{\pi(n), \pi(1)}$$

kjer je  $S_n$  množica vseh permutacij danih  $n$  mest.

Naivna rešitev je očitna, pogledamo  $(n - 1)!$  kombinacij, torej iz vsakega mesta v vsako drugo mesto, si zapišemo vse kombinacije in kakšno razdaljo smo prepotovali, ter izberemo tisto možnost, kjer je bila razdalja najkrajša.

# Programsko okolje

Za projekt sva uporabila programski jezik `python`. Uporabljala sva naslednje pakete:

- `networkx`: za definiranje in generiranje grafov
- `random`: za generiranje naključnih števil in seznamov
- `matplotlib`: za izrisovanje grafov
- `time`: za mertive časovne zahtevnosti
- `itertools`: za generiranje prvotne permutacije

Za preverjanje veljavnosti poti v grafu za problem potujočega trgovca, morava vedeti:

- Vsaka pot trgovca bo morala imeti vsa vozlišča primarnega grafa
- Vsaka pot trgovca bo morala imeti točno toliko povezav kot vozlišč
- Dolžina poti bo morala biti enaka številu vozlišč - 1

# Osnovni algoritem

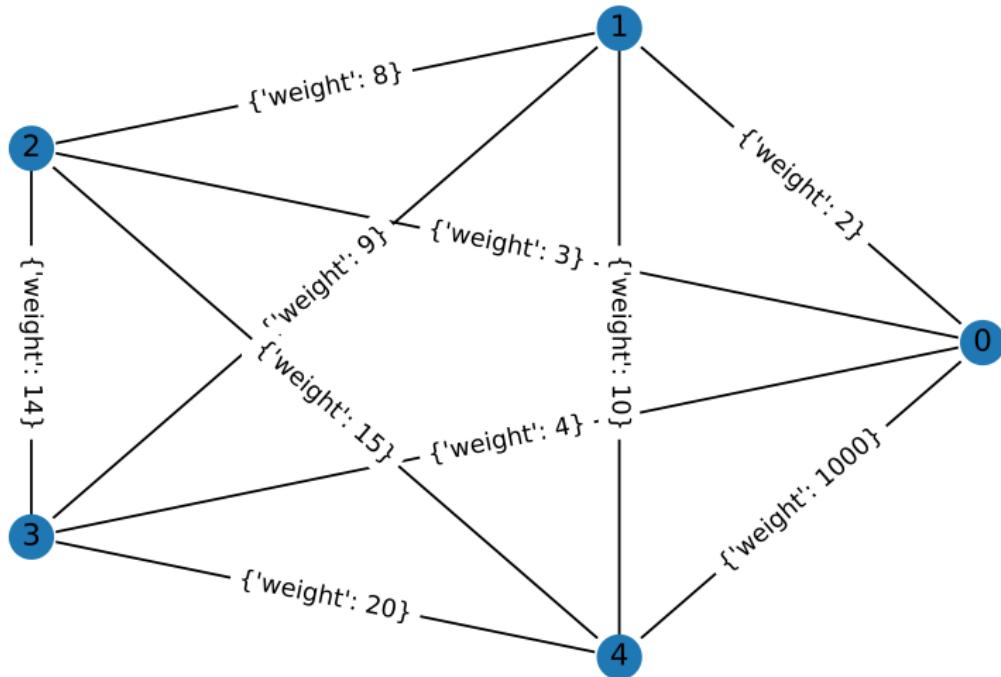
```
def dva_opt(graf, pot):  
  
    najboljsa_pot = pot  
    cena = cena_poti(graf, pot)  
  
    izboljsanje = True  
    while izboljsanje:  
        izboljsanje = False  
        for i in range(1, len(pot) - 2):  
            for j in range(i + 1, len(pot)):  
                if j - i == 1: continue  
                nova_pot = pot[:]  
                nova_pot[i:j] = pot[j - 1:i - 1:-1]  
                nova_cena = cena_poti(graf, nova_pot)  
                if nova_cena < cena:  
                    najboljsa_pot = nova_pot[:]  
                    cena = nova_cena  
                    izboljsanje = True  
    pot = najboljsa_pot  
  
    return (pot)
```

# Lin Kernighanov algoritem

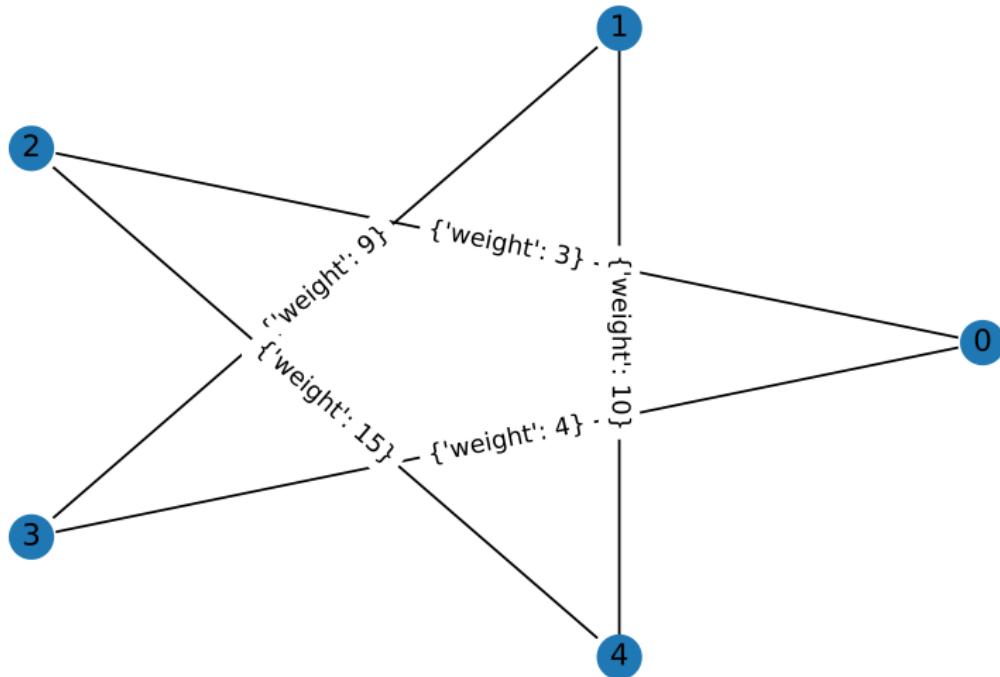
- Pri k-opt algoritmih na vsakem koraku zamenjamo  $k$  povezav, pri LK algoritmu pa pogledamo kakšen  $k$  se nam splača uporabiti.
- Povečujemo  $k$  dokler ne najdemo zamenjave povezav, ki izboljša obhod.
- Rešitve so lokalni ali celo globalni minimum.
- Vrne bolj točne rezultate.

# Primer grafa

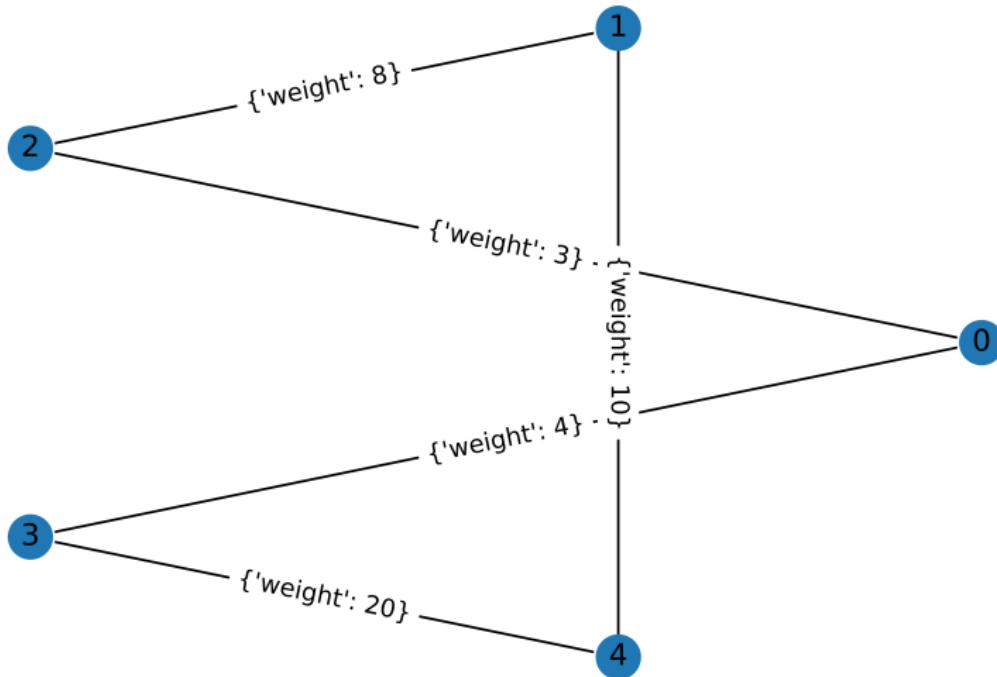
$$\begin{bmatrix} 0 & 2 & 3 & 4 & 1000 \\ 2 & 0 & 8 & 9 & 10 \\ 3 & 8 & 0 & 14 & 15 \\ 4 & 9 & 14 & 0 & 20 \\ 1000 & 10 & 15 & 20 & 0 \end{bmatrix} \quad (1)$$



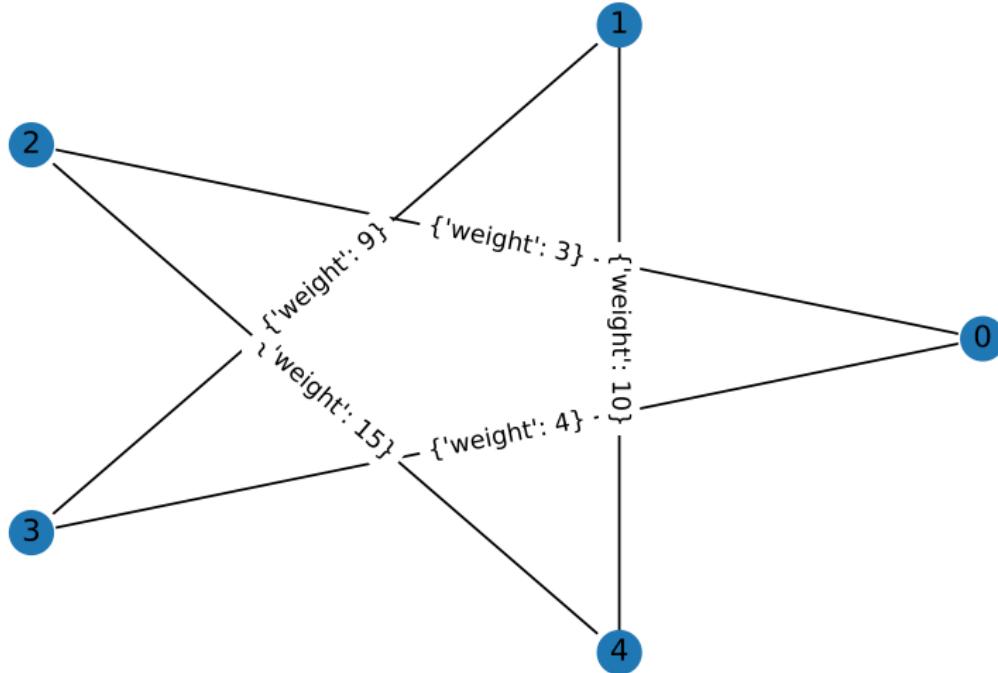
Slika 1: Poln graf na 5 vozliščih



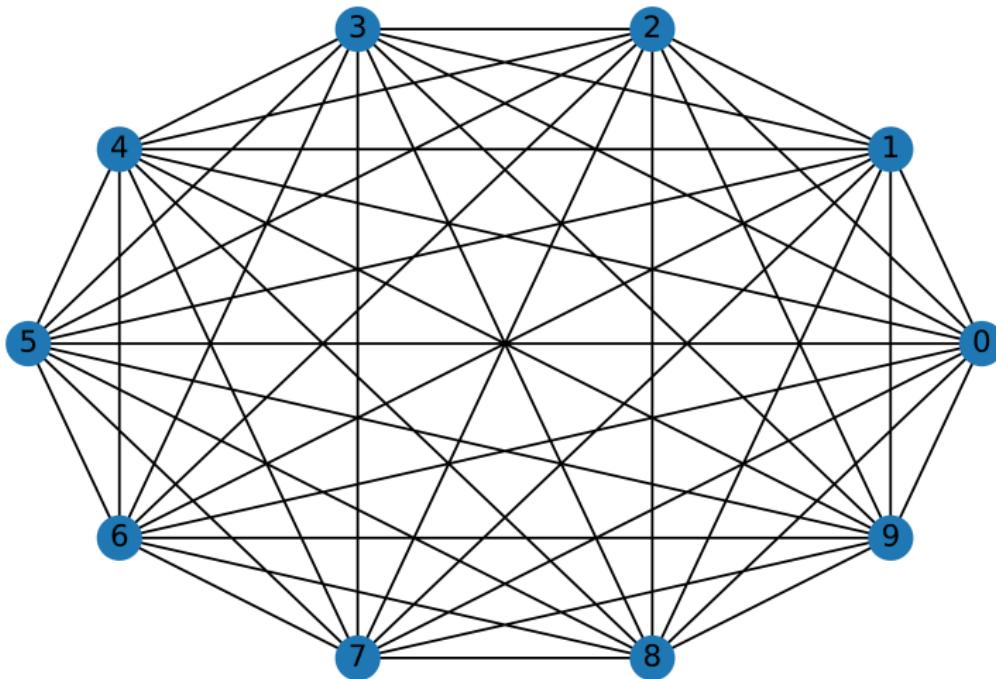
Slika 2: 2-opt



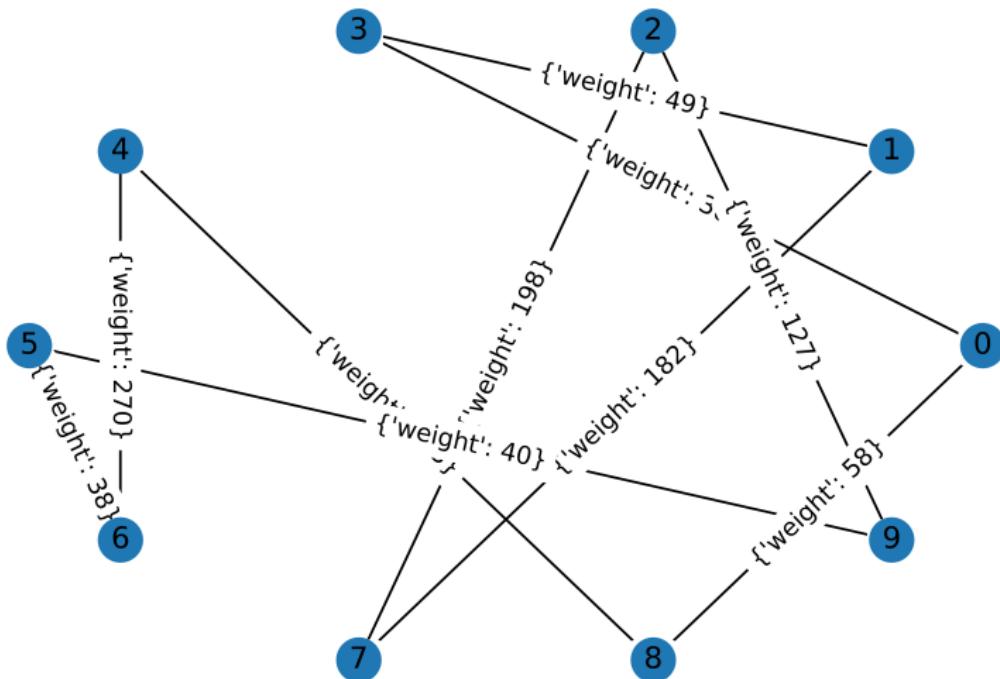
Slika 3: 3-opt



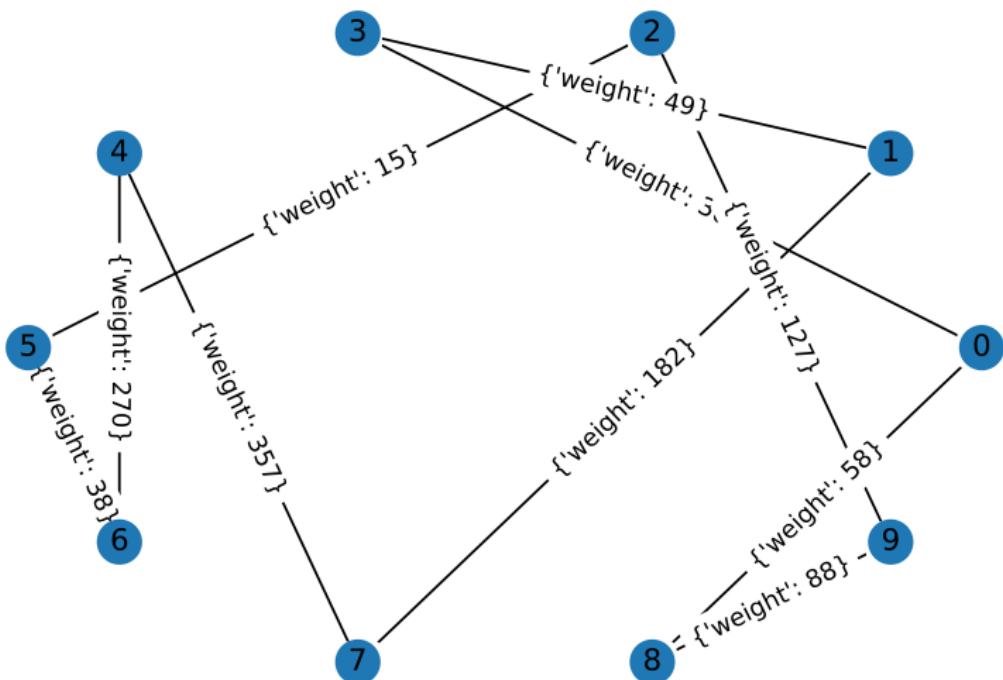
Slika 4: LK



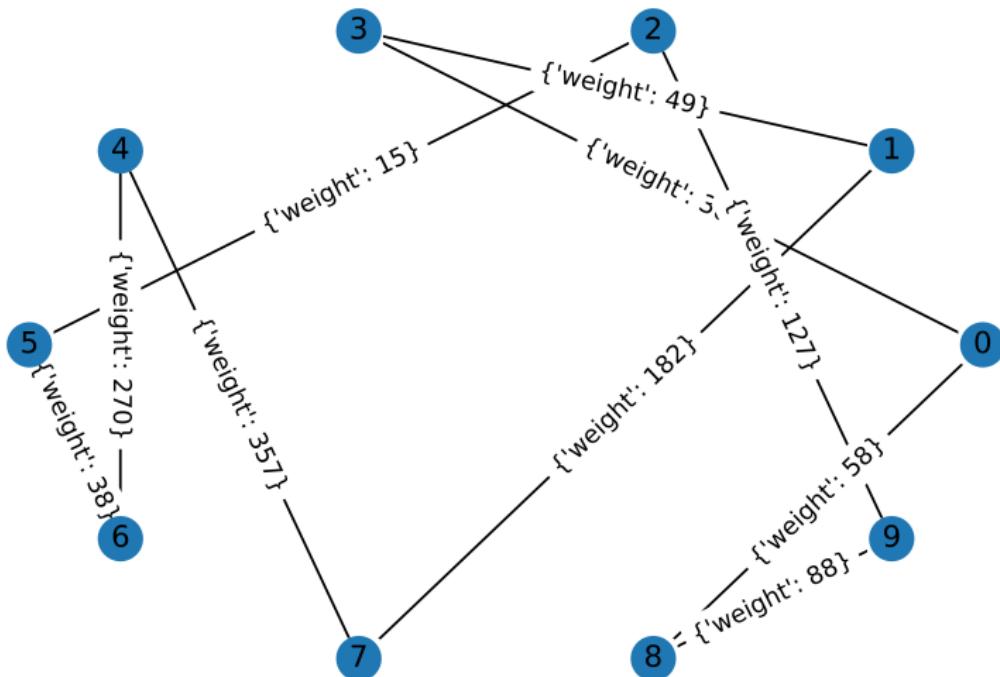
Slika 5: Poln graf na 5 vozliščih



Slika 6: 2-opt



Slika 7: 3-opt



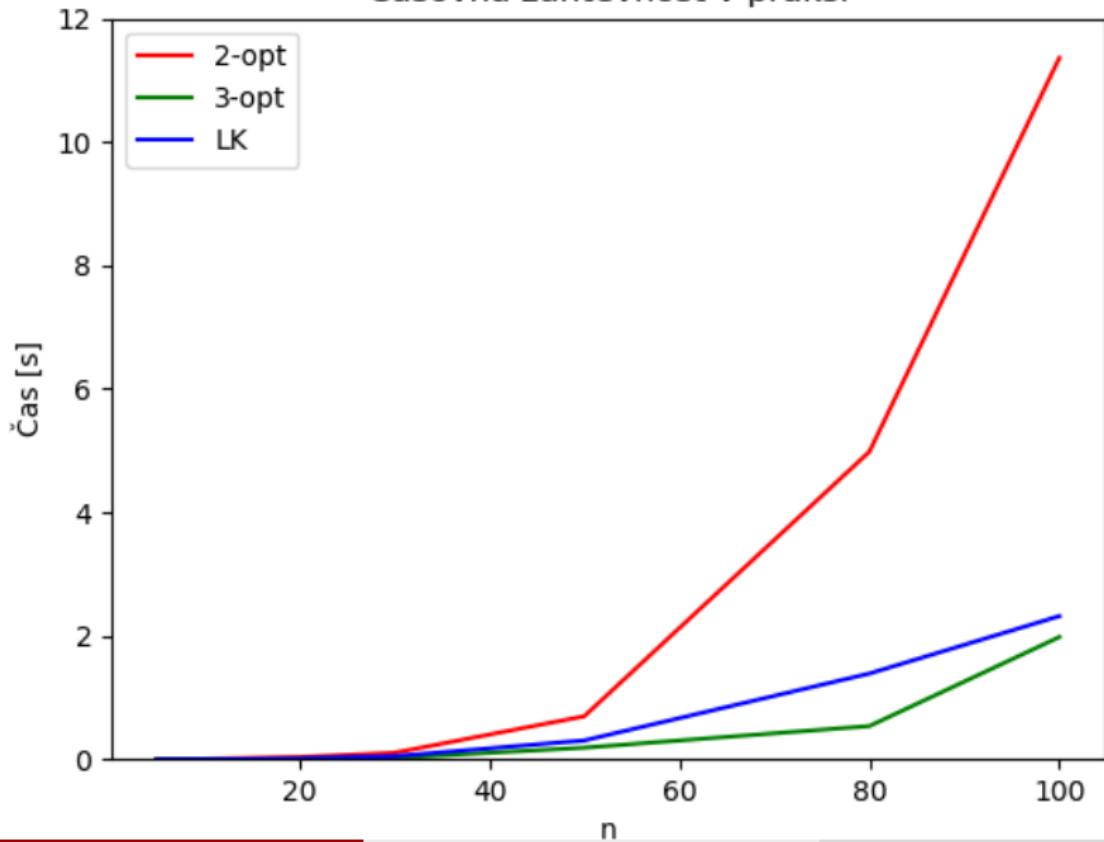
Slika 8: LK

# Časovna zahtevnost

$n$	začetek	2-Opt	cena 2-Opt	3-Opt	cena 3-Opt	LK	cena LK
5	49	0,000675	41	0,000276	41	0,000154	41
10	4903	0,004748	1595	0,000600	1484	0,000611	1484
20	9047	0,03257	1388	0,00801	1555	0,01151	1517
30	11825	0,10143	2255	0,01806	1907	0,04639	1729
50	23556	0,69378	2833	0,18529	2496	0,30218	2319
80	37169	4,97926	3352	0,53484	2890	1,38603	2362
100	46984	9,4767	3404	1,97813	2446	2,31723	2235
200	98344	194,093	4673	22,9016	3007	13,7700	2832
300	156543	1127,92	5365	162,040	2986	94,0348	2630
500	244733	7912,40	6689	402,52	3328	496,36	2900

Tabela 1: Tabela časov izvajanja algoritmov in cen v odvisnosti od  $n$

## Časovna zahtevnost v praksi



- ① A. Hagberg, D. Schult, P. Swart: *NetworkX Reference, Release 2.4*, [ogled 2. 1. 2020], dostopno na [https://networkx.github.io/documentation/stable/\\_downloads/networkx\\_reference.pdf](https://networkx.github.io/documentation/stable/_downloads/networkx_reference.pdf)
- ② *Optimization with 2-OPT - Part 1*, [ogled 3. 1. 2020], dostopno na <http://pedrohfsd.com/2017/08/09/2opt-part1.html>
- ③ *2-opt*, [ogled 3. 1. 2020], dostopno na <https://en.wikipedia.org/wiki/2-opt>
- ④ *3-opt*, [ogled 4. 1. 2020], dostopno na <https://en.wikipedia.org/wiki/3-opt>
- ⑤ *3-opt: basic algorithm*, [ogled 4. 1. 2020], dostopno na <https://tsp-basics.blogspot.com/2017/03/3-opt-iterative-basic-algorithm.html>
- ⑥ D. Karapetyan, G. Gutin, *Lin-Kernighan Heuristic Adaptations for the Generalized Traveling Salesman Problem*, [ogled 8. 1. 2020], dostopno na <https://arxiv.org/pdf/1003.5330.pdf>