Praktično delo 3

Tine Fajfar

June 6, 2021

Najprej v poglavju 1 opazujemo čas izvajanja posameznih algoritmov. V poglavju 2 najprej opišemo način generiranja grafov za katerega se izkaže, da je najverjetneje pomembno vplival na rezultate analiz. V poglavju 2.1 si ogledamo kateri izmed algoritmov v splošnem vrne najboljši rezultat, v poglavju 2.2 pa si ogledamo še kako velike so razlike med rezultati posameznih algoritmov. Ugotovimo, da se v naši implementaciji naloge najbolje tako v *času* kot tudi *kvaliteti* rešitve najbolje izkaže požrešni algoritem s strategijo najbližjega soseda.

1 Opazovanje časa izvajanje

V prvem delu sem opravil meritve izvajalnega časa posameznega algoritma pri različnem številu vozlišč v grafu. Generiral sem grafe z $n \in N$ vozlišči, pri čemer je N množica števil $\{10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800\}$. Pri tem sem za vsak n_i iz množice N generiral 30 instanc grafa in nato izvajalne čase posameznega algoritma povprečil na vseh 30 opravljenih meritvah enake velikosti vhodnega problema (slika 1).

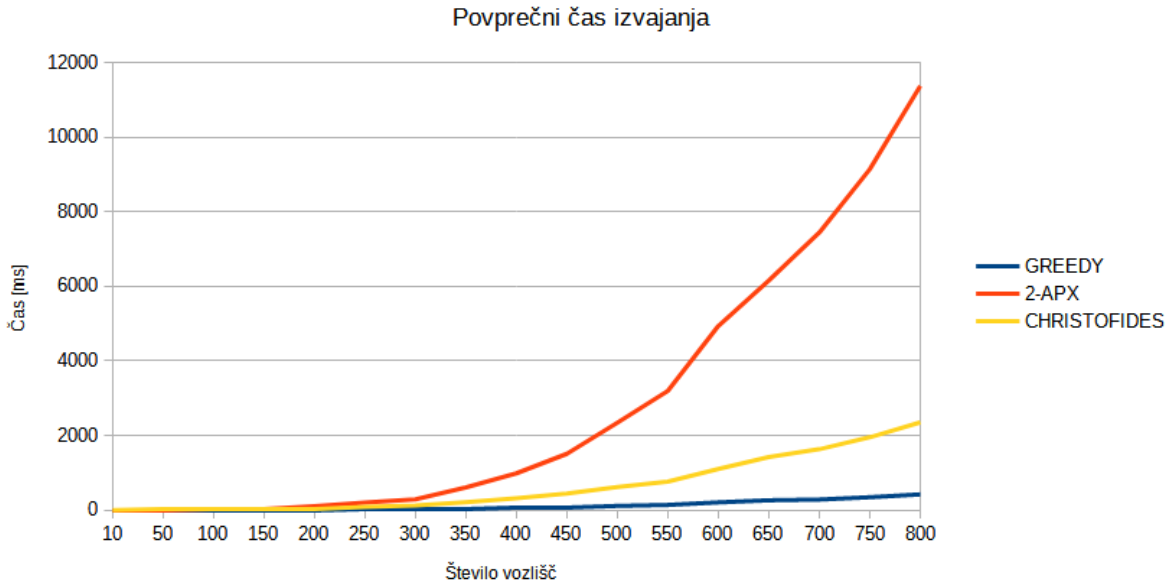

Figure 1: Opazovani čas izvajanja algoritmov glede na število vozlišč vhodnega grafa.

Pri implementaciji požrešnega algoritma sem izbral strategijo najbližjega soseda. Čeprav je pričakovani čas izvajanja $O(n^2)$ vidimo, da računalnik s takim številom vozlišč še nima težav in

rast na grafu bolj spominja na linearno naraščajočo premico. Vseeno, če podrobno pogledamo rast krivulje na intervalu od 10 do 400 in od 400 do 800 lahko opazimo, da gre res najverjetneje za krivuljo, ki opisuje kvadratno funkcijo.

Časovna zahtevnost algoritma Christofides je navzgor omejena z $O(n^4)$. Na grafu res vidimo krivuljo, ki ustreza taki funkciji. Če rezultate primerjamo s požrešnim algoritmom vidimo, da se krivulji lepo ujemata in je pri algoritmu Christofides glede na večji eksponent tudi rast ustrezno hitrejša.

Nazadnje omenimo 2-aproksimativni algoritem, katerega rezultati so presenetljivi. Ta algoritem bi sicer moral imeti časovno zahtevnost $O(n^2)$, problem pa nastane zaradi slabe implementacije algoritma. Ker med samim izvajanjem večkrat naredim kopijo nekega "vmesnega" grafa in to seveda ni "poceni" operacija vidimo, da čas izvajanja hitro poskoči.

Če smo si najprej ogledali algoritem Christofides, pri za katerega se čas izvajanja še zdi uporaben tudi v praksi, pa ne morem tako reči za 2-aproksimativni algoritem. Pri tem se zavedam, da je glavna krivda v moji slabi implementaciji algoritma, ki bi jo moral precej izboljšati.

2 Kvaliteta dobljenih rezultatov

Pri iskanju rešitve sem generiral grafe s številom vozlišč $n = 10, 50, 100, 200, 400$. Za vsako velikost grafa sem generiral 1000 različnih instanc grafov, na katerih sem izvedel posamezne algoritme.

Vsako novo točko sem najprej povezal s prvo točko na grafu, oddaljenost od nje pa sem določil naključno na nekem v naprej določenem intervalu, ki se ni spreminjal. Tako sem dejansko točke postavljal v krog s centrom v prvi točki grafa.

2.1 Število najboljših rezultatov

Kot vidimo na slikah 2, 3 in 4 sem s tem nenamenoma odkril način postavljanja točk, pri katerem se ob dovolj veliki gostiti točk, najbolje izkaže požrešni algoritem.

Pri majhni gostoti točk se sicer povprečno bolje izkaže algoritem Christofides, vendar ta primerjava ni zares smiselna, saj lahko na majhnem številu točk enostavno izvedemo preiskovanje vseh možnosti in dobimo optimalen rezultat.

Na vseh treh slikah 2, 3 in 4 ima 2-aproksimacijski algoritem najnižje število najboljših rezultatov torej vidimo, da očitno ta algoritem ni smiselno uporabljati na grafih "okroglih oblik", saj se bo očitno v splošnem odrezal najslabše izmed treh analiziranih.

2.2 Primerjava kakovosti rezultatov

V poglavju 2.1 nas je zanimalo zgolj, kateri izmed treh obravnavanih algoritmov vrne najboljši rezultat. V tem poglavju bomo primerjali kakovost rešitve, ki jo poda posamezen algoritem, torej dolžino poti, ki bi jo naj moral trgovski potnik opraviti pri obhodu vseh točk grafa.

Na slikah 5, 6 in 7 najprej opazimo, da so meje vedno bolj ravne kar je nekako pričakovano. Torej večji kot je graf oziroma dolžina poti, ki jo moramo prehoditi, bolj so rezultati konsistentni. Namreč razdaljo med točkami smo generirali z uporabo psevdo-naključne funkcije na nekem v

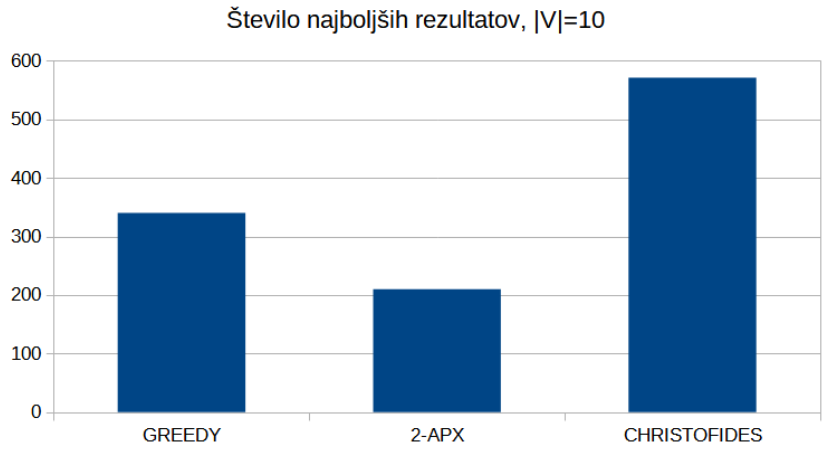

Figure 2: Število primerov, pri katerih je posamezen algoritem dal najboljši rezultat. Če sta dva algoritma dala enak najboljši rezultat, sem primer prištel obema, zato je vsota stolpcev večja od števila primerov, ki jih je bilo 1000.

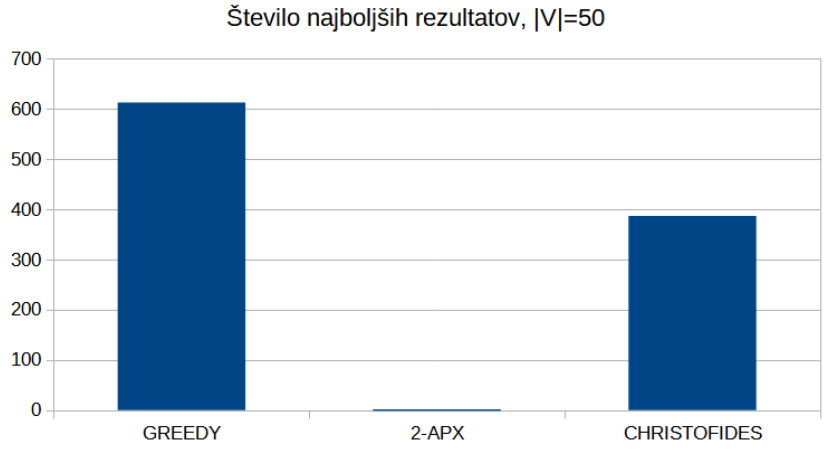

Figure 3: Število primerov, pri katerih je posamezen algoritem dal najboljši rezultat.

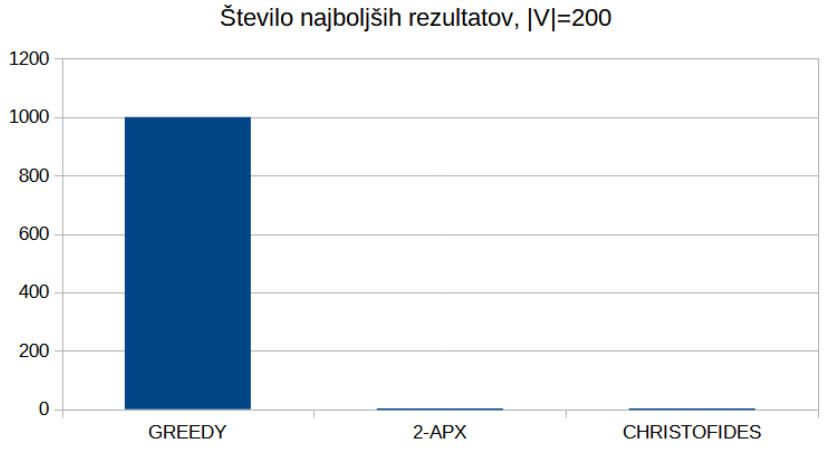

Figure 4: Število primerov, pri katerih je posamezen algoritem dal najboljši rezultat.

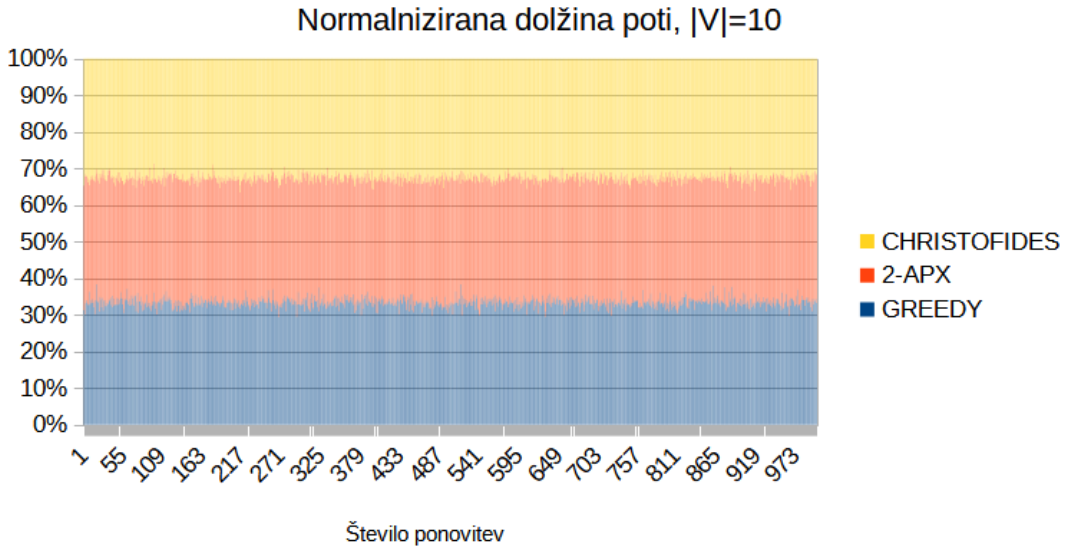

Figure 5: Dolžina vrnjenih poti za ponovitve na 1000 instancah grafov.

naprej določenem intervalu, ki se ni spreminjal in pri velikem številu točk se seveda porazdelitev generiranih razdalj bolje normalizira.

Če sedaj primerjamo samo dolžino poti, ki jo vrnejo algoritmi, opazimo naslednji dve stvari:

- Algoritem Christofides ne glede na število točk grafa vedno vrne rešitev, ki znaša približno 1/3 vseh vrnjenih dolžin poti.
- Požrešni algoritem z večanjem števila točk leze vedno bolj proti 30 odstotkom dolžine vseh poti, njegov delež pa prevzema 2-aproksimacijski algoritem.

Ugotovili smo, da razlike med posameznimi algoritmi v večini primerov niso velike in da se na okroglih grafih, kakršne generiramo v naši implementaciji naloge, najbolje izkaže požrešni algoritem.

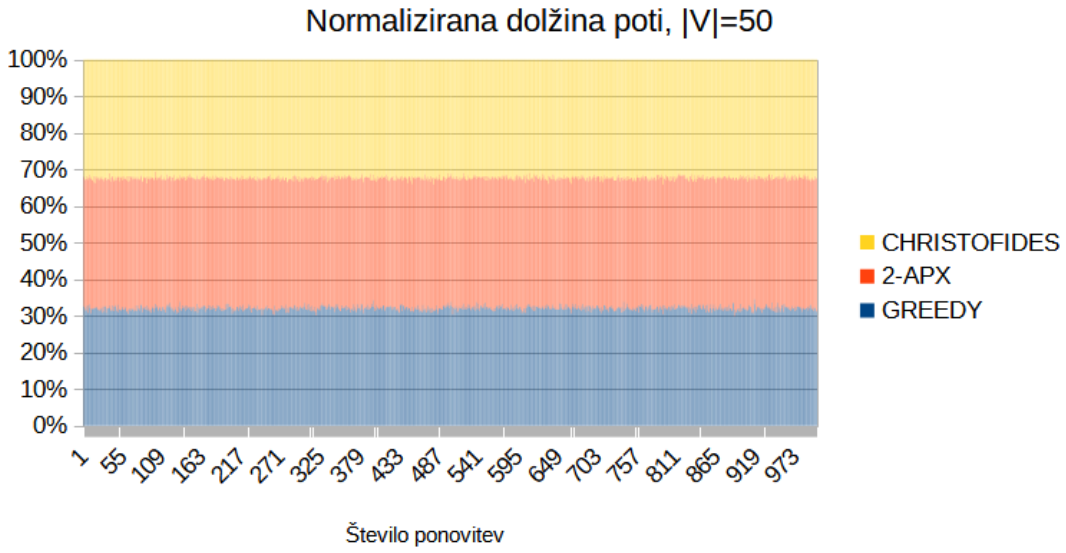

Figure 6: Dolžina vrnjenih poti za ponovitve na 1000 instancah grafov.

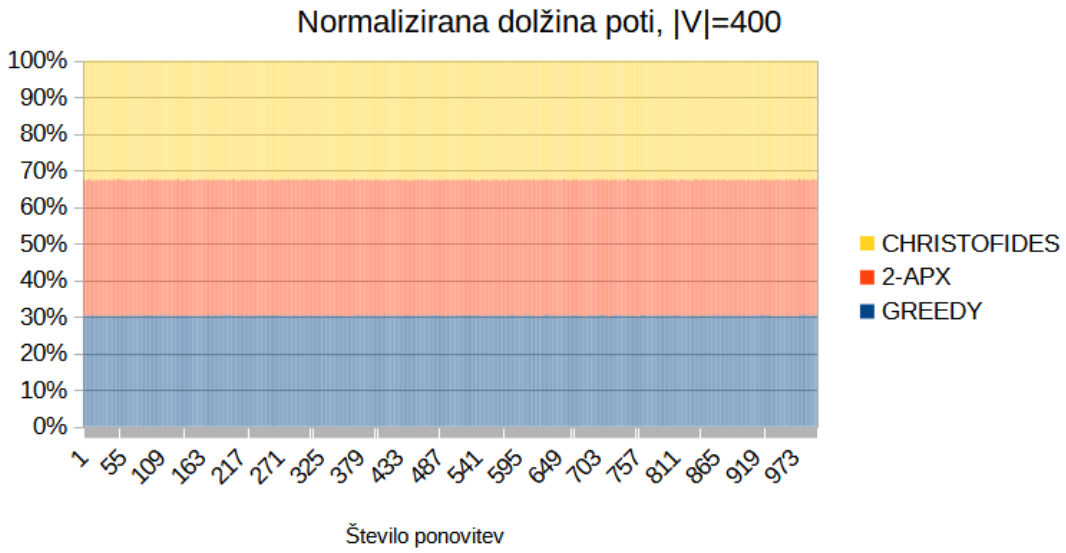

Figure 7: Dolžina vrnjenih poti za ponovitve na 1000 instancah grafov.