

Wienerjev indeks

Tjaša Renko
Darjan Pavšič

7. november 2019

1 Predstavitev problema

1.1 P1

Za fiksno število vozlišč n in drevo T je \mathcal{T}_{n+1} množica vseh dreves na $n + 1$ vozliščih, dobljena iz T z dodajanjem lista enemu iz vozlišč T , $W(\mathcal{T}_{n+1})$ pa množica vrednosti Wienerjevih indeksov dreves iz \mathcal{T}_{n+1} . Poiskati želimo tako drevo T na n vozliščih, da bo moč množice $W(\mathcal{T}_{n+1})$ čim manjša (čim večja).

1.2 P2

Za fiksno število vozlišč n iščemo drevo T z največjim premerom, da bo veljalo; obstajata u in w iz T , da list u pripet na v in w tak, da z odstranitvijo povezave uv in dodajanjem uw spremenimo vrednost indeksa za 1.

2 Opis dosedanjega in načrt za nadaljnje delo

Za izvedbo naloge sva se odločila za programski jezik *Python*, za lažje delo z grafi pa sva si pomagala s knjižnico `networkx`, tako da lahko sedaj grafe predstavljamo kot objekte, sposodila pa sva si tudi vgrajeni funkciji računanja poti med poljubnima vozliščema grafa ter računanja premera.

2.1 P1

Za začetek sva potrebovala ustrezne grafe. Ker je izomorfnih dreves na n vozliščih preveč za preverjanje vseh, sva generirala nekaj naključnih s pomočjo Kruskalovega algoritma. Poskusila sva tudi z vgrajenimi funkcijami, a so se

izkazale za neuporabne, saj so vračale med seboj preveč podobna drevesa.

Za časovno optimalno računanje Wienerjevih indeksov sva najprej shranila dolžine najkrajših poti med vsakima točkama grafa. Tako sva potem lahko izračunala indeks vsakega posameznega novega drevesa v linearnem času; potrebovala sva namreč le vsoto poti iz vozlišča, ki sva mu dodala list, vsoto vseh poti ter število vozlišč; vse te podatke pa sva že imela.

Ko sva dobila podatke za na začetku generirana drevesa, sva na vsakem izvedla algoritem za iskanje lokalnih ekstremov v okolici grafa, in sicer vrsto metahevrstike z angleškim imenom *simulated annealing*. Ta na vsakem koraku preveri stanje sosednjega grafa in se s padajočo verjetnostjo odloča, ali se bo vanj preselila, hkrati pa shranjuje najboljši rezultat do sedaj. Imela sva težave z definiranjem sosednjega drevesa, zato nameravava preiskusiti par načinov in izbrati ugodnega. Pri prvotnem sva odstranila naključno povezavo, nato pa povezave dodajala in odstranjala, dokler nisva dobila povezanega drevesa, kar se ni izkazalo za učinkovit postopek.

Po končanem algoritmu *simulated annealing* je bilo treba samo še preveriti najboljšega izmed vseh rezultatov, ki jih je podala omenjena metahevrstika. Razmišljala sva, da bi na začetku generirala večje število grafov in zadnji algoritem izvedla le na tistih, ki dajejo boljše rezultate, kar bi bilo hitreje. To bo dobro natanko takrat, ko bodo imeli sosednji grafi podobno moč množice $W(\mathcal{T}_{n+1})$, torej želiva to doseči pri novem definiranju sosedov. Poskusila bova z menjavo povezave pri listih, tako kot je to opisano v problemu P2. Nameravava poskusiti tudi na drugačen način, in sicer razmišljava o implementaciji genetskega algoritma. Za to morava razmisliti, kako primerno križati ali kako drugače modificirati ugodni drevesi. Ko bova optimizirala postopke in dobila zadostno število iskanih dreves, se bova morala spomniti še, kako jih primerjati. Za prvih par naravnih števil nameravava tudi preveriti točno rešitev.

2.2 P2

Napisala sva funkcijo za generiranje vseh izomorfnih dreves na n vozliščih ter funkcije, ki najprej iz vseh grafov poberejo tiste z iskano lastnostjo, nato pa izmed teh tistega z največjim premerom, če kak tak graf sploh obstaja. To je točna rešitev problema, a je časovno zahtevna ne le generacija grafov, temveč tudi računanje premera. Načrt zdaj je pomikati se po množici ustreznih grafov tako, da najprej narediva par poljubnih dreves na n vozliščih s premerom $n - 2$ in lastnost preveriva za njih, nato vzameva premer $n - 3$ in tako dalje,

morda pa tudi na vsakem izmed teh dreves izvesti kakšno metahevrstiko ali kaj podobnega. Skrbi naju, da po takšnem postopku dolgo časa sploh ne bi našla grafa z željeno lastnostjo, zato bova morala dobro določiti število izbranih grafov za primerjanje in mogoče zbrati drugačne načine za iskanje.

2.3 Zaključek

Morala bova še implementirati zgoraj omenjene načrte. Sproti bova videla, kaj je potrebno spremeniti, česa se bova znebila in kaj spremenila ali dodala. Poleg iskanja lokalnih optimumov na večjih grafih bova pogledala globalne na manjših ter poskušala sklepati na lastnostih, če se bo pokazal vzorec. Če bo šlo, bova priredila algoritme izdelave prvotnih dreves za testiranje glede na rezultate, ki sva jih dobila na prejšnjih testiranjih.