Wiener inverse interval problem

Tjaša Renko
Darjan Pavšič

6. november 2019

1 Predstavitev problema

1.1 P1

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

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 metahevristike 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 metahevristika. 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 $W(\mathscr{T}_{n+1})$, torej želiva to doseči pri novem definiranju sosedov. Ko bova optimizirala postopke in dobila zadostno število iskanih dreves, se bova morala spomniti še, kako jih primerjati. ...bom še končala

2.2 P2

...sledi

2.3 Nadaljnje delo -> se združi s prejšnjo točko

Ko bova na podlagi dovolj velikega vzorca dreves videla, katera so za najina problema najbolj ugodna, bova s pomočjo genetskega algoritma postopoma izbirala le tista najugodnejša in jih na vsakem koraku križala ali modificirala. Ta postopek bova ponavljala, dokler bo postopek na posameznem koraku še vračal ugodnejša drevesa.