Predpisana drevesa z najmanjšim/največjim Wienerjevim indeksom

Klemen Hovnik
Matija Gubanec Hančič
Jan Rudof

7. november 2018

1 Uvod

Naj bo $G = (V(G), E(G))$ enostaven povezan neusmerjen graf. *Wienerjev indeks* (oziroma *Wienerjevo število* $W(G)$) je definiran kot

$$W(G) = \frac{1}{2} \sum_{u \in V(g)} \sum_{v \in V(g)} d_G(u, v). \tag{1}$$

Tukaj označimo z $d_G(u, v)$ razdaljo med vozliščem u in v v grafu G.

Naša naloga je, da analiziramo lastnosti dreves z določenim številom vozlišč in fiksno maksimalno stopnjo vozlišč, ki imajo najmanjši Wienerjev indeks. Podobno nas zanimajo tudi lastnosti dreves na določenem številu vozlišč s fiksnim premerom, ki imajo največji možni Wienerjev indeks.

2 Opis dela

Za izvedbo projekta smo si izbrali programski jezik *Sage*, saj ta že vsebuje orodja za delo z grafi, prav tako pa ima tudi generator dreves in že vgrajeno funkcijo za izračun Wienerjevega indeksa. Najprej smo se lotili izračuna Wienerjevih indeksov na preprostih grafih z malo vozlišči, da vidimo, kako naj bi ta struktura grafov z minimalnimi oziroma maksimalnimi indeksi izgledala v splošnem.

Za iskanje najmanjših Wienerjevih indeksov pri določenem številu vozlišč in pri fiksni maksimalni stopnji smo naredili slovar slovarjev, kamor smo shranjevali optimalno dobljena drevesa. Podobno smo naredili slovar slovarjev za drevesa z maksimalnim Wienerjevim indeksom pri določenem številu vozlišč in pri fiksnem premeru grafa. Seveda smo to naredili le za grafe z malo vozlišči, saj se kaj kmalu izkaže, da je naš algoritem za izračun indeksov časovno prepotraten.

Tako smo si pripravili vse potrebno za analizo strukture dreves, ki nam bo

pomagala kasneje pri iskanju indeksov večjih grafov. Ideja je, da bi glede na dobljene rezultate na grafih z manj vozlišči lahko predvidevali, kako naj bi ta drevesa izgledala v splošnem.

3 Načrt za nadaljnje delo

Naslednji korak našega dela bo konstrukcija *genetskega algoritma*.

Genetski algoritem je metahevristika, navdihnjena s strani procesov naravne selekcije in spada v razred *razvojnih algoritmov*. Uporablja se za generiranje kvalitetnih rešitev v optimizaciji, ki temeljijo na operatorjih kot so mutacija, križanje in selekcija.

V genetskem algoritmu se uporabi množica kandidatov za rešitev, ki jih nato razvijamo do optimalne rešitve. Vsak kandidat ima določene lastnosti, katere lahko spremenimo oziroma lahko mutirajo. Evolucija rešitev se ponavadi začne na naključni izbiri kandidatov, katere potem s pomočjo iteracije razvijamo. Na vsakem iterativnem koraku se potem oceni primernost novih kandidatov za optimizacijski problem. Najboljše kandidate potem uporabimo za naslednji korak iteracije in tako dalje. Na koncu se algoritem zaključi, ko doseže maksimalno število iteracijskih korakov oziroma, ko dobi najboljši približek optimalni rešitvi.

Naša začetna množica kandidatov bodo optimalna drevesa, ki smo jih dobili z našim prvim enostavnim algoritmom. Nato bomo ustvarili genetski algoritem, ki bo iz teh začetnih podatkov generiral nova optimalna drevesa. Ta postopek bomo nadaljevali (in pri tem selekcionirali iz novo nastalih dreves le najboljše za naslednje korake), dokler ne bomo prišli do optimalnih dreves za določeno število vozlišč. Pri tem bomo morali paziti, da se bo ohranjala maksimalna stopnja vozlišč, oziroma v drugem primeru, premer.

Na koncu bomo primerjali algoritma in pogledali, kdaj se ustavi naš enostavni algoritem za iskanje grafov z max/min Wienerjevim indeksom oziroma za katero število vozlišč je genetski algoritem še učinkovit.