Stable roommates problem

Martin Pezdir, Žiga Potrebuješ

21.02.2017

Kazalo

1 Uvod

Naloga najinega projekta je bila sprogramirati metodo, ki bo rešila problem stabilnih sostanovalcev ali angleško stable roomates problem. Ta problem je podoben problemu stabilne poroke, ki sta ga predstavila David Gale in Lloyd Shapley. Problem stabilnih sostanovalcev je verzija omenjenega problema, kjer imamo eno samo množico. Vsaka oseba v množici, ki ima sodo kardinalnost n (torej n je sodo število) razvrsti ostalih $n-1$ oseb v zaporedje glede na njihove preference. Naš cilj je poiskati stabilno prirejanje oziroma razdelitev naše množice v množico $n/2$ parov sostanovalcev tako, da ne obstajati dve osebi, ki nista sostanovalca/sostanovalki, ki preferirata drug drugega bolj, kot pa svoja sostanovalca določena v prirejanju. Gale in Shapley sta pokazala, da obstajajo primeri, v katerih ne moremo določti stabilnega prirejanja v problemu stabilnih sostanovalcev.

2 Opis algoritma na primeru

Na naslednjem enostavnem primeru bova opisala kako deluje algoritem, ki reši problem stabilnih sostanovalcev, katerega je podal Robert W. Irving leta 1985. Recimo, da imamo osebe kot na spodnji sliki in imamo podane tudi njihove preference.

PERSON	1ST CHOICE	2ND CHOICE	3RD CHOICE	4TH CHOICE	5TH CHOICE
Ralph	Penny	Boris	Oliver	Tammy	Ginny
Penny	Oliver	Ginny	Ralph	Boris	Tammy
Boris	Oliver	Tammy	Penny	Ralph	Ginny
Ginny	Ralph	Boris	Tammy	Penny	Oliver
Oliver	Ralph	Penny	Ginny	Tammy	Boris
Tammy	Penny	Ralph	Ginny	Boris	Oliver

Slika 1: Tabela preferenc

V prvi fazi se posamezniki med seboj 'zasnubijo'. To naredijo po naslednjih dveh pravilih:

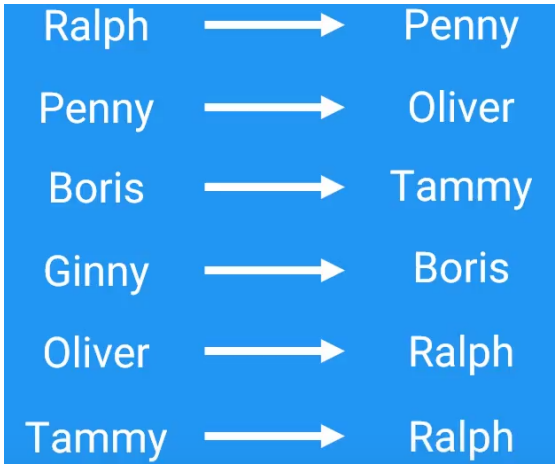
- Če posameznik x prejme snubitev od y potem:
  1. To snubitev takoj zavrne, če že ima drugo, boljšo snubitev(torej snubitev od nekega z, ki je v njegovem seznamu preferenc pred y).
  2. Če je ta snubitev boljša od sedanje, to snubitev zadrži v premislek in zavrne slabšo snubitev, ki jo je zadrževal do sedaj.
- Posameznik x zasnubi vse ostale in sicer v vrstnem redu, v katerem se pojavijo v njegovem seznamu preferenc. Pri nekem potencialnem sostanovalcu se ustavi, ko dobi neko obljubo o premisleku za potencialnega sostanovalca in nadaljuje z naslednjim v njegovem seznamu preferenc, ko se pojavi zavrnitev od nekega drugega sostanovalca, s katerim je do sedaj imel oblikovano obljubo o snubitvi.

V našem primeru torej Ralph zasnubi Penny, katera sprejme saj do sedaj ni prejela nobene druge snubitve. Penny nato zasnubi Oliverja, ta njeno snubitev sprejme zaradi istega razloga kot prej. Toda v naslednem koraku Boris zasnubi Oliverja, kateri ga zavrne saj že ima snubitev od Penny, katera je višje od Borisa na seznamu Oliverjevih preferenc. Pri tem moramo v algoritmu iz seznama preferenc Oliverja zbrisati Borisa in simetrično iz Borisovega seznama zbrisati Oliverja, saj se ta kombinacija ne more več zgoditi. Več o tem v nadaljevanju pri opisu najinega algoritma. Boris nato zasnubi naslednjo na svojemu seznamu preferenc, Tammy, ki ga sprejme saj trenutno nima nobene snubitve. Tako nadaljujemo do zadnje osebe, pri tem pa simetrično brišemo iz seznama preferenc, kombinacije, ki se ne morejo več zgoditi zaradi zavrnitev. Že v tej fazi se lahko zgodi, da stabilno prirejanje ne obstaja-to se zgodi, če katero izmed oseb vse ostale osebe zavrnejo, saj le-te držijo boljšo snubitev od snubitve osebe, ki jo zavrnejo. Torej v tem primeru stabilno prirejanje ne bo obstajalo. V našem primeru se to ne zgodi in tabela po prvi fazi izgleda tako:(rdeče so obarvane zavrnitve):

PERSON	1ST CHOICE	2ND CHOICE	3RD CHOICE	4TH CHOICE	5TH CHOICE
Ralph	Penny	Boris	Oliver	Tammy	Ginny
Penny	Oliver	Ginny	Ralph	Boris	Tammy
Boris	Oliver	Tammy	Penny	Ralph	Ginny
Ginny	Ralph	Boris	Tammy	Penny	Oliver
Oliver	Ralph	Penny	Ginny	Tammy	Boris
Tammy	Penny	Ralph	Ginny	Boris	Oliver

Slika 2: Tabela prve faze

Če se ta faza uspešno konča, torej če ni osebe, katero bi zavrnile vse ostale osebe, potem dobimo pare oseb, katere so dale in sprejele snubitev. Torej v našem primeru, Ralph zasnubi Penny in ta njegovo snubitev drži tudi po koncu prve faze, ker je ni zasnubila nobena oseba, ki bi bila višje na seznamu Penny-inih preferenc kot Ralph. V našem primeru dobimo pare, ki držijo snubitve:



Slika 3: Kdo drži čigavo snubitev po prvi fazi

Sedaj lahko naredimo redukcijo naše tabele preferenc. To tabelo naredimo glede na to, kdo je posamezno osebo zasnubil. Ko vemo, kdo je zasnubil osebo A, lahko izbrišemo vse osebe iz seznama preferenc osebe A, katere so rangirane nižje od osebe, ki je zasnubila osebo A po prvi fazi. V našem primeru je Ralpha zasnubil Oliver, vendar sta Tammy in Ginny že po prvi fazi izbrisani iz seznama preferenc Ralpha zato se tu nič ne spremeni. Penny po prvi fazi drži snubitev od Ralpha zato lahko iz svojega seznama preferenc izbriše Borisa in simetrično iz Borisovega seznama preferenc se izbriše Penny. To nadaljujemo do zadnje osebe, dokler ne dobimo reducirane tabele. Če se v temu koraku slučajno zgodi, da je kak reduciran seznam preferenc prazen, potem spet ne obstaja stabilno prirejanje.

PERSON	1ST CHOICE	2ND CHOICE	3RD CHOICE	4TH CHOICE	5TH CHOICE
Ralph	Penny	Boris	Oliver	Tammy	Ginny
Penny	Oliver	Ginny	Ralph	Boris	Tammy
Boris	Oliver	Tammy	Penny	Ralph	Ginny
Ginny	Ralph	Boris	Tammy	Penny	Oliver
Oliver	Ralph	Penny	Ginny	Tammy	Boris
Tammy	Penny	Ralph	Ginny	Boris	Oliver

Slika 4: Redukcija

V zadnjem koraku ustvarjamo cikle ljudi, glede na njihove druge in zadnje preference v reducirani tabeli preferenc. V tej fazi lahko začnemo s poljubno osebo, ki ima več kot eno osebo v njihovem reduciranem seznamu preferenc, naj bo to oseba p_i. Potem je oseba q_i druga preferenca p_i in p_{i+1} zadnja preferenca q_i. To rekurzivno zaporedje ponavljamo dokler se nek p_i ne ponovi. Ko se to zgodi lahko iz seznamov preferenc izbrišemo osebe p_{i+1} in q_i. Torej glede na spodnjo sliko se zavrneta Boris-Ginny, Tammy-Boris, Ralph-Oliver in Penny-Ralph. To ponavljamo, dokler obstaja oseba, ki ima vsaj dve preferenci v svojem reduciranem seznamu.

p	Ralph	Ginny	Boris	Oliver	Ralph
q	Boris	Tammy	Ralph	Penny	

Slika 5: Zadnji korak

V našem primeru tako dobimo naslednje stabilno prirejanje:

PERSON	1ST CHOICE	2ND CHOICE	3RD CHOICE	4TH CHOICE	5TH CHOICE
Ralph	Penny	Boris	Oliver	Tammy	Ginny
Penny	Oliver	Ginny	Ralph	Boris	Tammy
Boris	Oliver	Tammy	Penny	Ralph	Ginny
Ginny	Ralph	Boris	Tammy	Penny	Oliver
Oliver	Ralph	Penny	Ginny	Tammy	Boris
Tammy	Penny	Ralph	Ginny	Boris	Oliver

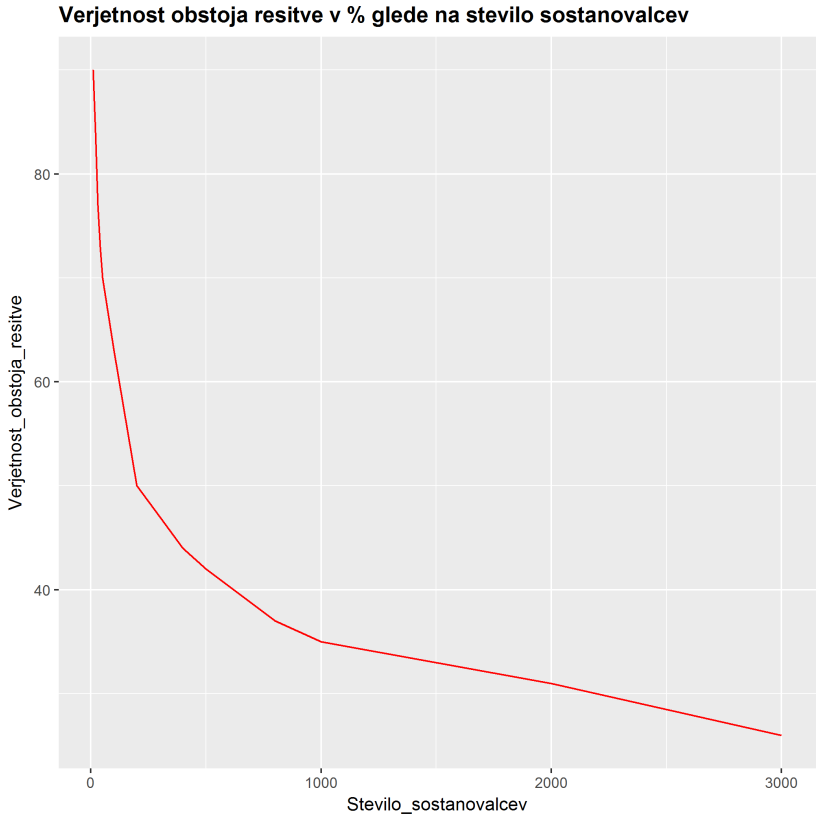
Slika 6: Stabilno prirejanje

3 Algoritem in opis generiranja podatkov

Najin algoritem, ki v polinomskem času izračuna stabilno prirejanje je podrobno opisan v komentarjih v kodi algoritma(datoteka stableroommates.py). Podatke lahko generiramo s pomočjo datoteke generator.py. Datoteka vsebuje preprosto funkcijo generator(n), kjer s pomočjo zank while in for ter s pomočjo csv.writerja zgeneriramo datoteko sostanovalci.csv. Ta datoteka ima n vrstic, v vsaki je ena oseba. Torej v i-ti vrstici je oseba i. Ker imamo v glavnem algoritmu funkcijo zapolni, ki naključno priredi preference tem osebam, tudi če jih te niso podale, bo algoritem vseeno deloval. Če želimo preveriti ali za n oseb obstaja stabilno prirejanje, pri čemer so preference posameznikov naključne, a dopustne moramo najprej zagnati funkcijo generator(n) iz datoteke generator.py. Za naključnost poskrbi funkcija zapolni, ki uporablja random.shuffle. Nato moramo zagnati še funkcijo stableroomate('sostanovalci.csv') iz datoteke stableroommates.py. Pri tem moramo paziti, da bodo vse datoteke shranjene na isti lokaciji na računalniku.

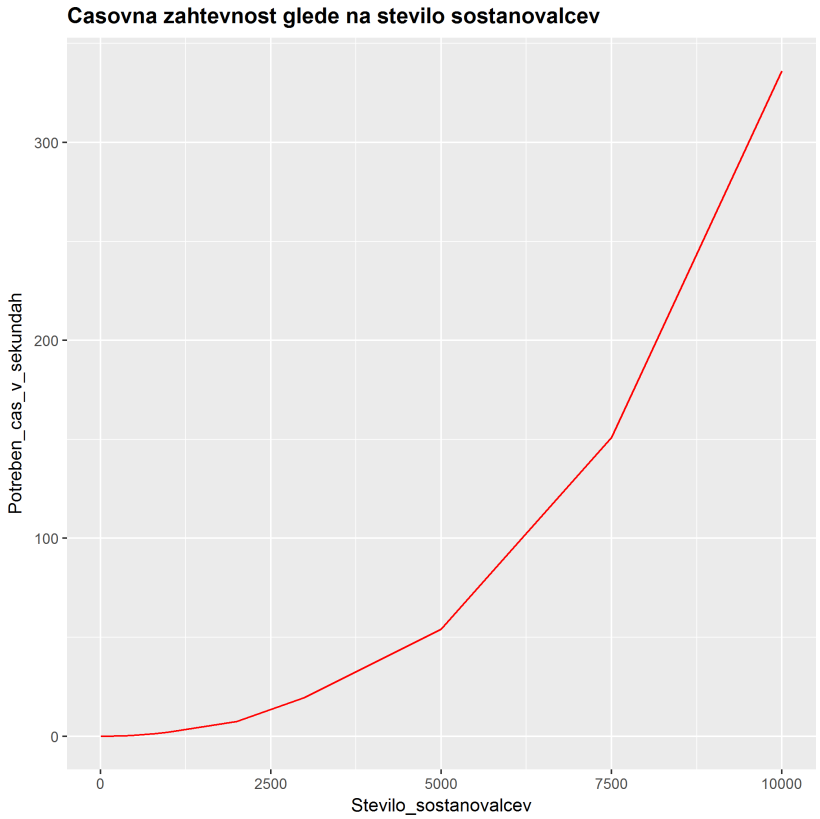
4 Analiza

Ugotovila sva, da za liho število oseb stabilno prirejanje ne bo obstajalo, kar je sicer logično, vendar se splača omeniti. Potem sva analizirala, kako se verjetnost obstoja rešitve spreminja s številom oseb, katerim moramo določiti stabilno prirejanje. To sva naredila s pomočjo funkcije verjetnostobstoja(dat) v datoteki stableroommates.py. Ta funkcija sprejme csv datoteko v kateri so našteti samo sostanovalci brez preferenc, zato da vsakič generiramo naključne preference. Nato izračuna kolikokrat od 100-tih poskusov bo rešitev obstajala. To sva ponovila 10krat in izračunala povprečje verjetnosti. Ugotovila sva, da se verjetnost obstoja stabilnega prirejanja zmanjšuje s povečevanjem števila sostanovalcev. Rezultati so prikazani v spodnjem grafu:



Slika 7: Verjetnost obstoja rešitve glede na število oseb

Analizirala sva tudi časovno zahtevnost algoritma, tako da sva povečevala število oseb in štela čas s pomočjo paketa time. Izkaže se, da se porabljen čas res povečuje kvadratno s spremembo števila vhodnih podatkov. Torej, če povečamo število vhodnih podatkov za 2-krat, se bo porabljen čas povečal za 4-krat. Če povečamo število vhodnih podatkov za 4-krat, se bo porabljen čas povečal za 16-krat, itd. Tudi oblika krivulje na spodnjem grafu nakazuje na kvadratno časovno zahtevnost. Rezultati so prikazani v spodnjem grafu:



Slika 8: Časovna zahtevnost

5 Celoštevilski algoritem

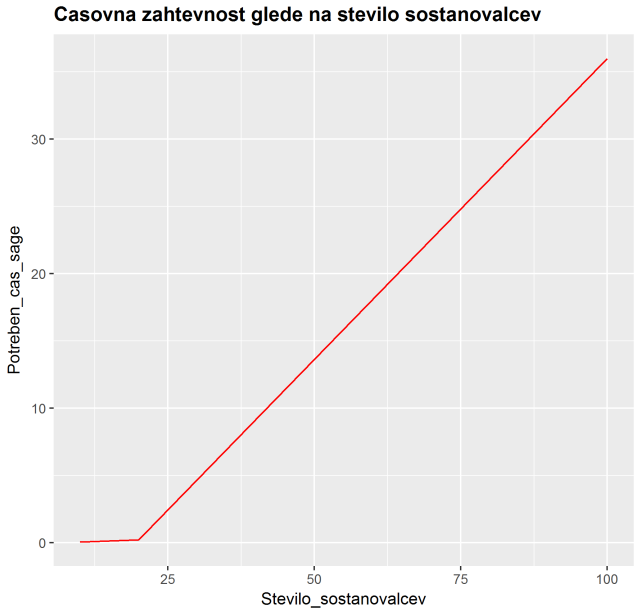
Problem sva tudi poskusila reševati s celoštevilskim linearnim programom v programu Sage. Najprej sva na podoben način kot v pythonu sestavila slovar rangiranj, tako da je naključno generiral rangiranja za vsakega posameznika v odvisnosti od števila oseb n. Program sva reševala brez ciljne funkcije. Najprej sva definirala funkcijo $x_{i,j}$, ki je enaka 1, če sta osebi i, j skupaj ter 0 sicer. Vključila sva štiri pogoje. Zahtevala sva, da je za poljubno osebo $x_{i,i} = 0$, torej, da nobena oseba ne izbere sama sebe. Drugi pogoj je bil simetričnost, kjer sva zahtevala, da za poljubna i, j velja $x_{i,j} = x_{j,i}$. Tretjič sva s pogojem da za vsak i velja da je

$$\sum_{j=1}^{n} x_{i,j} = 1$$

, da ima vsaka oseba natanko enega sostanovalca. Nazadnje pa sva še za vsak $i \neq j$ zahtevala, da je

$$x_{i,j} + \sum_{k<i,j} x_{j,k} + \sum_{k<i,j} x_{i,k} \leq 1$$

. Zadnji pogoj pravi, da ne smeta obstajati dve osebi, ki nista skupaj, pa bi obe rajši bile druga z drugo kot z dodeljeno osebo. Časovna zahtevnost programa Sage je bila veliko večja. Že pri majhnih n je velika razlika glede na reševanje istega problema iste velikosti v pythonu. Recimo za rešitev problema pri $n = 10$ je python v povprečju porabil 0.0001 sekunde, Sage pa 0,041 sekunde. Razlika se samo povečuje, pri $n = 100$, je python še vedno porabil manj kot 1 sekundo, Sage pa v povprečju kar 36 sekund.



Slika 9: Časovna zahtevnost v Sage

6 Viri

Literatura

[1] https://en.wikipedia.org/wiki/Stable_roommates_problem,

[2] Irving, Robert W., *Journal of Algorithms*, Volume 6, Issue 4, Pages 455-598, 1985

[3] https://www.youtube.com/watch?v=DuDvDrAXXbk

[4] https://en.wikipedia.org/wiki/Stable_marriage_problem