Master rad

Statistička fizika epidemija: modeli na kompleksnim mrežama

Darja Cvetković

Smer:Teorijska i eksperimentalna fizika

Mentor: Dr Marija Mitrović Dankulov

Fizički fakultet,Univerzitet u Beogradu,Septembar 2020

Sadržaj

1 Uvod

Prenosive bolesti su neraskidivi deo ljudskih zajednica još od samih početaka civilizacije i čovečanstva. Epidemije koje su se javljale tokom istorije bile su uzrok ne samo velikog broja smrtnih ishoda, već i ekonomskih, političkih i društvenih kriza. Kako bi razumeli uzrok same bolesti, njeno ponašanje i širenje kroz populaciju, a samim tim i našli način njenog kontrolisanja, razvila se posebna naučna disciplina: epidemiologija [8]. Iako se počeci epidemiologije mogu uočiti još od vremena starih Grka(tačnije, Hipokrata) ona je nagli procvat doživela tek posle Drugog svetskog rata. Do tada, doduše, može se izdvojiti par stvarno značajnih doprinosa, od kojih je prvi rad Džon Grant-a(17.vek) koji je prvi prikupio i sistematizovao podatke u vezi sa zaraznim bolestima u svojoj knjizi *"Natural and Political Observations made upon the Bills of Mortality"*. Grant je u njoj kvantifikovao obrasce rođenja, smrti, i javljanja bolesti primećujući razlike između muškaraca i žena, velike smrtnosti novorođene dece, ali i razlike između naseljenih i nenaseljenih regija, pa i sezonalnih varijacija. Drugi značajan doprinos je prvi matematički model koji je razvio Danijel Bernuli(18.vek) u svrhu inokulacije[1] protiv malih boginja,pokazavši da bi univerzalna inokulacja dovela do porasta srednjeg životnog veka. Još jedan veoma značajan doprinos razumevanju zaraznih bolesti je rad anesteziologa Džon Snou-a(19.vek) koji je pre poznavanja samog procesa prenosa bolesti izučavao vremenske i prostorne obrasce javljanja i širenja kolere u Londonu.

U 21.veku epidemiologiju su preoblikovali mnogi faktori - povećanje interdisciplinarnosti i kompleksnosti naučnih istraživanja, pojavljivanje novih metoda i tehnologija za generisanje i analizu podataka (od posebnog značaja je povećanje memorijskih i skladišnih kapaciteta računara), a i pojavljivanje i razvoj novih naučnih disciplina kao što su nauka o kompleksnim mrežama[9] . Ona se bavi proučavanjem kompleksnih mreža različitih vrsta (telekomunikacione mreže, biološke mreže, kompjuterske mreže, socijalne mreže itd.) razmatrajući njihove članove("čvorove") i veze između njih, i time omogućava realniji prikaz pojedinih sistema i odgovarajućih fenomena [2].

Mreže su odavno bile prepoznate kao ključan faktor epidemiološkog modelovanja, ali je tek nedavni priliv obimne količine podataka omogućio razumevanje velikog broja fenomena, što zahteva detaljno teorijsko razumevanje uzajamnog dejstva između epidemioloških procesa i mreža. Veliki broj radova je pokazao da je većina mreža u realnom svetu statistički heterogena i da pokazuju dinamičku samo-organizaciju, što su tipične osobine kompleksnih sistema [15]-[18]. Realne mreže relevantne za širenje epidemije su veoma drugačije od regularnih rešetki: one su hijerarhijski organizovane sa nekoliko izrazito povezanih čvorova koji se mogu ponašati kao *hub*-ovi (čvorišta), dok je većina ostalih čvorova slabo povezana , socijane i mreže infrastrukture su organizovane u "zajednice" blisko povezanih čvorova. Principi organizacije i korelacije u obrascima povezanosti definišu mrežne strukture koje jako utiču na evoluciju i ponašanje epidemijskih i zaraznih procesa. Kompleksnost osobina mreža često se ogleda u njihovim statističkim distribucijama koje su generalno asimetrične, imaju široke repove (*heavy-tailed*), i koje imaju opseg između nekoliko različitih redova veličina.

Dokazano postojanje osobina kao što su klasterovanje, fluktuacije na velikim skalama, javljanje zajednica koje određuju obrasce povezanosti u realnom svetu zahtevale su potrebu za matematičkim pristupom i opisom koji bi omogućio opis kompleksnosti mreža. Međutim, čak ni za najjednostavnije dinamičke procese ne postoji opšte matematičko rešenje, tj. master jednačina, koja bi opisivala ceo sistem. Zbog ovog razloga, istraživački rad fokusiran na matematičkom i kompjuterskom modelovanju epidemioloških i difuzionih procesa krenuo je da se javlja u različitim disciplinama [19]. Izučavanje evoulucije mreža i pojavljivanje makroskopskog kolektivnog

[1] medicinski termin za unošenje nečega što će rasti ili se razmnožavati, a najčešće se odnosi na ubrizgavanje mikroorganizama, seruma, vakcine ili antigena u organizam čoveka ili životinje, u svrhu razvijanja imuniteta

ponašanja u kompleksnim sistemima esencijalno prati principe teorije statističke fizike neravnotežnih faznih prelaza [20] - pa tako statistička fizika kreće da vodi istraživanja vezana za dinamičke procese u kompleksnim mrežama [21] [22].

Ovaj rad će se baviti tzv. "compartmental"(pregradnim) modelima nastalih u 20. veku kao posledica Kermack-McKendrick teorije, koji opisuju transmisiju prenosivih bolesti pod idejom da se populacija može podeliti na različite "odeljke" (kompartmente) koji predstavljaju različite faze bolesti, i iskoristiti njihove relativne veličine za modelovanje dalje vremenske evolucije bolesti. Bolje rečeno, koristićemo jezik ovih modela na realnim kompleksnim mrežama - i time pokazati ključan uticaj strukture mreže na dinamiku epidemije upoređivši ga sa klasičnim "mean-field" predviđanjima originalnih modela. Takođe ćemo pokazati uticaj preventivnih mera - socijalnog distanciranja i nošenja maski na dinamiku epidemije, i to sa početkom u raznim vremenskim trenucima. Na kraju, predstavićemo rezultate simulacije(opisane nešto kompleksnijim kompartmental modelom) za širenje virusa COVID-19 u Srbiji.

2 Pregradni modeli

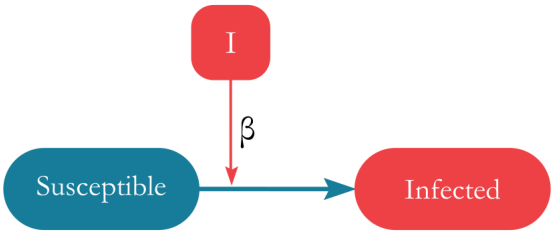
Osnovna ideja pregradnih modela je jednostavna - populacija se može podeliti na "odeljke" koji predstavljaju različite stadijume bolesti (na primer, **S**- Susceptibilni(*'Susceptible'*), **I**- Inficirani(*'Infected'*), **R**- Oporavljeni *'Recovered'*)), i pojedinci mogu prelaziti iz jednog odeljka u drugi. Takođe, glavna pretpostavka je ta da je populacija "dobro izmešana"(*well-mixed*)- svaka osoba može da stupi u kontakt sa svakom drugom osobom. Dakle, ovo je zapravo model srednjeg polja. Dinamiku svakog odeljka, odnosno celog modela, opisuju odgovarajuće linearne diferencijalne jednačine, koje imaju analitička rešenja. Međutim, modeli se mogu rešiti i numerički, i u ovom radu ćemo to učiniti koristeći populaciju predstavljenu kompleksnom mrežom što će dati mnogo realniju sliku problema širenja epidemije. U svakom slučaju, mi na osovu modela pokušavamo da predvidimo kako se bolest prenosi, trajanje epidemije, ukupan broj zaraženih, ali i da izračunamo razne epidemiološke parametre, i da otkrijemo kako određene mere u borbi protiv epidemije utiču na istu. No, pre svega toga osvrnimo se prvo na same pregradne modele.

2.1 SI model

Ovo je najjednostavniji epidemiološki model. Populacija je podeljena na dva odeljka - podložni, tj. susceptibilni **S** i inficirani **I** . Dinamika je takođe jako jednostavna: kada zdrava, odnosno podložna osoba stupi kontakt sa bolesnom, odnosno inficiranom osobom, ona i sama sa nekom verovatnoćom β postaje inficirana (slika 2.1). Simbolično ovo možemo zapisati na sledeći način:

$$S + I \xrightarrow{\beta} I$$

U ovom pojednostavljenom slučaju jasno je da ako se osoba zarazi, ona ostaje zaražena zauvek.Ovaj slučaj imamo kod HIV virusa, citomegalovirusa(CMV), ili herpesa [37].



Slika 2.1: Šematski dijagram SI modela

Matematički sve ovo može se predstaviti sistemom običnih diferencijalnih jednačina:

$$\frac{\partial S_t}{\partial t} = -\beta S_t \frac{I_t}{N} \tag{2.1}$$

$$\frac{\partial I_t}{\partial t} = +\beta S_t \frac{I_t}{N} \tag{2.2}$$

Gde su:

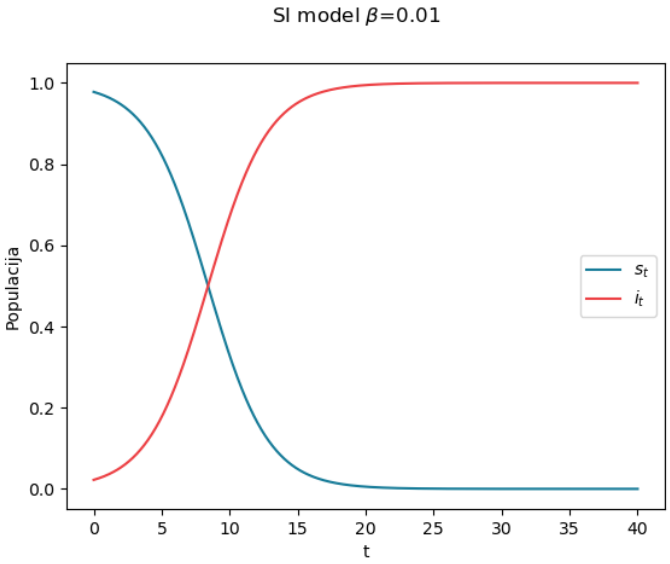
- β - stopa infekcije : srednji broj kontakata po osobi u vremenu pomnožen verovatnoćom prenošenja bolesti između 2 osobe.
- N - veličina populacije,odnosno broj osoba
- S_t - broj podložnih osoba
- I_t/N - udeo inficiranih ljudi,tj. verovatnoća da će podložna osoba sresti zaraženu osobu

Ako obe jednačine podelimo sa N, normiraćemo ih, odnosno,radićemo sa udelima populacije u odeljcima umesto sa ukupnim brojevima u istim:

$$\frac{\partial s_t}{\partial t} = -\beta s_t i_t \tag{2.3}$$

$$\frac{\partial i_t}{\partial t} = +\beta s_t i_t \tag{2.4}$$

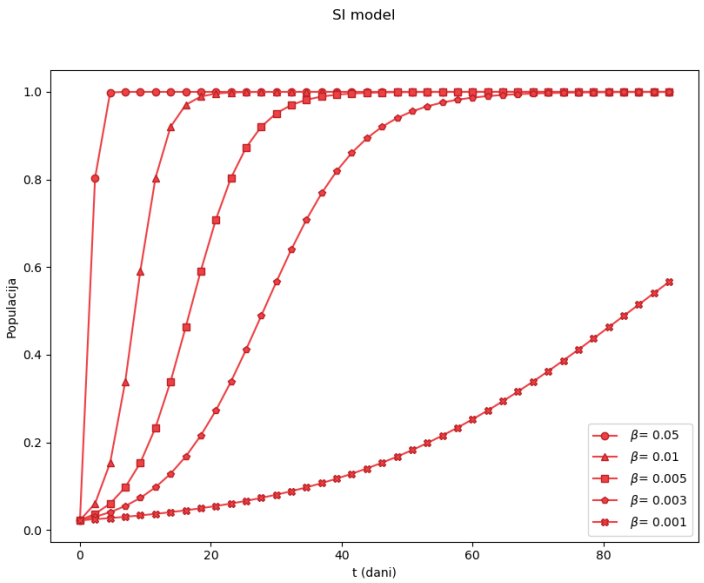
Uglavnom se uzima da je u početnom trenutku približno cela populacija susceptibilna : $s_0 = (1-1/N) \approx 1, i_0 = 1/N \approx 0$. Bez ulaženja u rešavanje ovih jednačina, odnosno u njihovo analitičko rešavanje, očigledno je da nakon dovoljno vremena svi postanu inficirani. Koristeći Python i paket za rešavanje diferencijalnih jednačina ODEINT[2] iz modula *SciPy* na slici 2.2 grafički su predstavljene vremenske evolucije(u danima) zdravog i zaraženog udela populacije veličine $N = 45$ za vrednost $\beta = 0.01$ u slučaju kada u početnom trenutku postoji jedna zaražena osoba. Korišćen kod priložen je u dodatku **B** kao "ODEINT SI model".



Slika 2.2: Vremenska evolucija epidemije SI modela za odgovarajuće parametre

Šta se dešava sa *Infected* populacijom kada se parametar β menja? Parametar β kontroliše verovatnoću inficiranja tokom jednog kontakta, pa što je on veći to je veća šansa da se zaraza prenese i bolest će se širiti brže kroz populaciju, tj. epidemija će pre zahvatiti celu populaciju. Rezultati su predstavljeni na slici 2.3. Vidimo da za $\beta = 0.05$ udeo inficirane populacije naglo raste i da ona za 2 dana postiže maksimimum, dok sa druge strane za $\beta = 0.001$ udeo dostiže maksimum tek posle 80 dana.

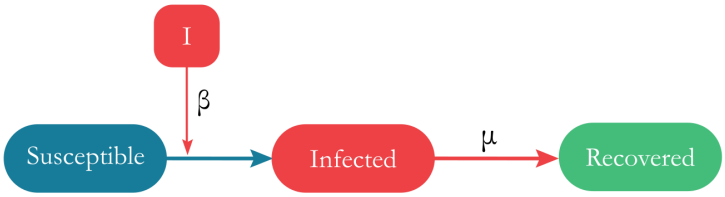
[2]ODEINT koristi LSODE(*Livermore Solver of Ordinary Differential Equations*) algoritam za rešavanje običnih diferencijalnih jednačina



Slika 2.3: Vremenska evolucija epidemije za različite parametre β

SI model je jako grub prikaz širenja epidemije u populaciji, ali nam pruža neke osnovne informacije o uticaju verovatnoće zaraze na širenje infekcije.

2.2 SIR model



Slika 2.4: Šematski dijagram SIR modela

Realniju sliku o evoluciji epidemije dobijamo uvođenjem dodatnih odeljaka. U tu svrhu, na SI model dodajemo novi odeljak: **R**(*Recovered*) koji predstavlja deo populacije koji je preležao bolest i stekao imuninet i više nije podložan zarazi. U tom smislu, često se pored oporavljenih implicitno uključuju i smrtni slučajevi. Dakle, inficirana osoba iz **I** odeljka spontano prelazi u "recovered" **R** nekom stopom μ (slika 2.4) [4, 5] :

$$S + I \xrightarrow{\beta} I$$

$$I \xrightarrow{\mu} R$$

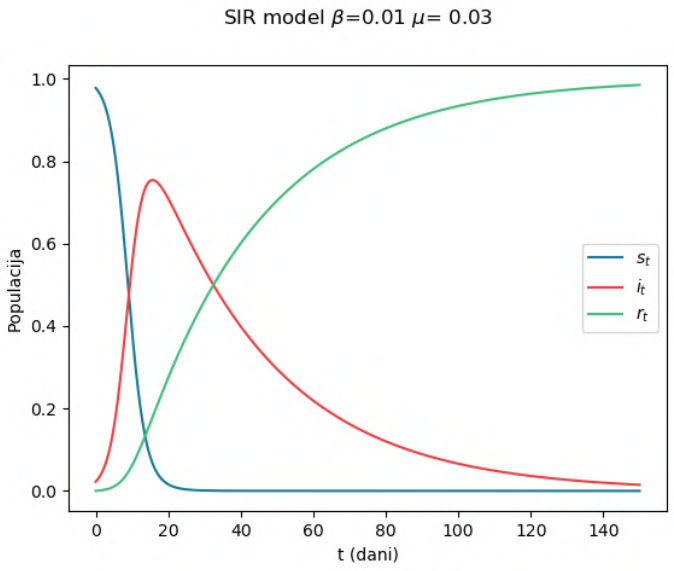
Odgovarajuće jednačine su:

$$\frac{\partial s_t}{\partial t} = -\beta s_t i_t \tag{2.5}$$

$$\frac{\partial i_t}{\partial t} = +\beta s_t i_t - \mu i_t \tag{2.6}$$

$$\frac{\partial r_t}{\partial t} = +\mu i_t \tag{2.7}$$

Ovde, β ima isto značenje kao i u SI modelu, ali kao što smo rekli, javlja se i dodatan parametar μ koji predstavlja stopu, odnosno verovatnoću oporavka. Tačnije, može se reći da je stopa tranzicije iz odeljka **I** u odeljak **R** proporcionalna broju zaraznih osoba μI_t ,što je ekvivalentno pretpostavci da je verovatnoća oporavka zaražene osobe u intervalu dt zapravo μdt. Ako je osoba zarazna neki prosečni vremenski period T onda $\mu = 1/T$, što bi se dobilo iz pretpostavke da je vreme koje je osoba provela u zaraznom stanju slučajna varijabla eksponencijalne raspodele [1]. U razmatranje se može uzeti i vremenski zavisna stopa oporavka, što dalje usložnjava model [7].



Slika 2.5: Vremenska evolucija epidemije SIR modela za odgovarajuće parametre

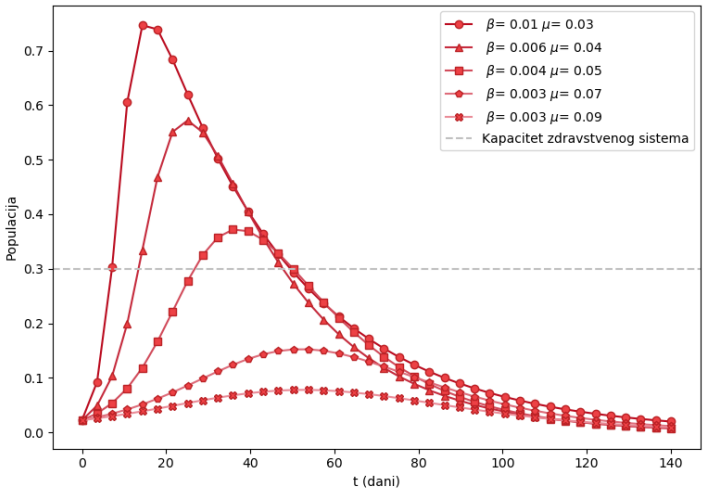
Postojanje *Recovered* odeljka postepeno smanjuje broj inficiranih, što je predstavljeno i na slici 2.5 za populaciju pod identičnim uslovima kao i za SI slučaj ($N = 45$, $\beta = 0.01$,u početnom trenutku postoji jedna zaražena osoba) uz dodatnu verovatnoću $\mu = 0.03$. Ako ove jednačine saberemo vidimo ovako definisan model(kao i SI) implicira konstantnu veličinu populacije:

$$\frac{\partial s_t}{\partial t} + \frac{\partial i_t}{\partial t} + \frac{\partial r_t}{\partial t} = 0 \tag{2.8}$$

Postoje metode koje u razmatranje dodaju i promene u populaciji (u smislu rođenja i smrti), ali u ovom radu o tome neće biti reči - može se samo napomenuti da se tome pristupa u slučaju kada sama priroda ispitivane bolesti to zahteva. Mogu se uočiti sledeće činjenice:

1. Broj *Susceptible* osoba može samo opadati
2. Broj *Recovered* osoba može samo rasti
3. *Infected* populacija u jednom trenutku dostigne maksimum i kreće da opada dok se posle nekog vremena potpuno ne oporavi

Ono što je od velikog značaja i što se može jasno primetiti iz ponašanja *Infected* odeljka je to da veliki deo populacije može postati zaražen u jako uskom vremenskom periodu što može dovesti do preopterećenja zdravstvenog sistema. Variranjem parametra *Infected* kriva se može "spljoštiti" (Slika 2.6) , u realnom svetu to se postiže odgovarajućim preventivnim merama u zavisnosti od prirode bolesti (vakcinacija, socijalno distanciranje, nošenje maski...). Kod koji je upotrebljen za crtanje SIR plotova dat je u dodatku **B** pod nazivom "ODEINT SIR model".



Slika 2.6: Zaravnenje *Infected* krive usled promene parametara sistema

Sa Slike 2.6 uočimo da se simultanom promenom verovatnoća: smanjenjem β i povećanjem μ, postiže ne samo spuštanje maksimuma inficirane populacije već i njegovo pomeranje ka kasnijim vremenskim trenucima - maksimum epidemije će se dostići kasnije i ona će duže trajati, ali će se sprečiti preopterećenje zdravstvenog sistema (čiji je kapacitet predstavljen horizontalnom isprekidanom linijom). Ovo nije slučaj ukoliko se samo jedan od parametara menja - tada dolazi do smanjenja maksimuma ali ne i do njegovog pomeranja. Dakle, odnos parametara igra ključnu ulogu u dinamici epidemije i ovde je zgodno definisati tzv. **osnovni reproduktivni broj** R_0 [3, 5]. Uslov javljanja epidemije u podložnoj populaciji je :

$$\frac{\partial i_t}{\partial t} > 0 \tag{2.9}$$

Pretpostavimo da je u početnom trenutku javljanja epidemije $s_t \approx 1$, i zamenom u izraz 2.9:

$$i_t \approx i_0 e^{(\beta-\mu)t} \tag{2.9}$$

Uporedivši ovaj rezultat sa nejednakošću 2.6 dobija se uslov:

$$R_0 = \frac{\beta}{\mu} > 1 \tag{2.10}$$

Ova veličina određuje da li imamo epidemiju ili ne: ukoliko je $R_0 < 1$ epidemija se "ugasi" , u suprotnom raste eksponencijalno. Jedan od načina interpretacije ovog broja je srednji broj novih infekcija koje nastanu od jedne osobe (tj. ukoliko zaražena osoba uspe da prenese bolest na makar jednu zdravu osobu epidemija se raširi). Osnovni reproduktivni broj zavisi od mnogo faktora koji su vezani za karakteristike same bolesti, i o njemu će biti više reči u narednim sekcijama.

2.3 SEIR model

Jedan od glavnih nedostataka SIR modela je što ne uključuje inkubacioni period, tj. zaraza momentalno širi. Ovaj problem se rešava uvođenjem još jednog odeljka: **E** - *Exposed*. Kada zdrava osoba dođe u kontakt sa inficiranom ona postaje *izložena (exposed)* sa nekom verovatnoćom β , i prelazi u *zaraznu (infected)* nekom

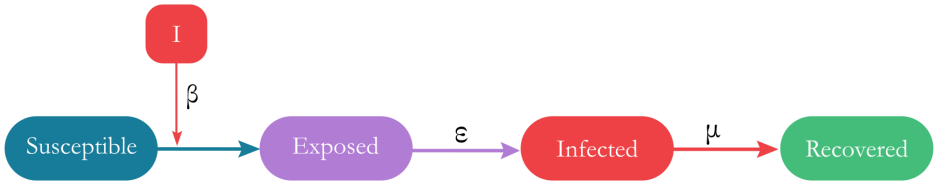
fiksnom stopom ϵ [4] :

$$S + I \xrightarrow{\beta} E$$

$$E \xrightarrow{\epsilon} I$$

$$I \xrightarrow{\mu} R$$

Dok je u *Exposed* odeljku osoba ne može zaraziti druge osobe (iako možda sama razvija simptome bolesti), i u tom smislu može se reći da $1/\epsilon$ predstavlja period latentnosti.



Slika 2.7.: Šematski dijagram SEIR modela

Diferencijalne jednačine u ovom slučaju su:

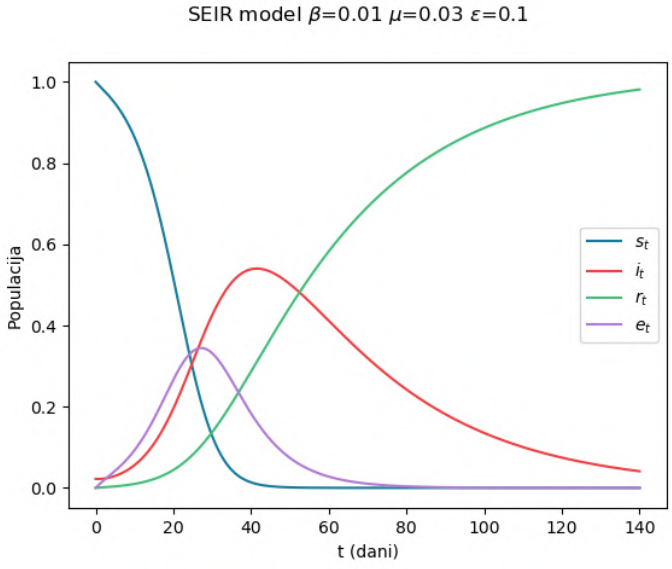
$$\frac{\partial s_t}{\partial t} = -\beta s_t i_t \tag{2.11}$$

$$\frac{\partial e_t}{\partial t} = \beta s_t i_t - \epsilon e_t \tag{2.12}$$

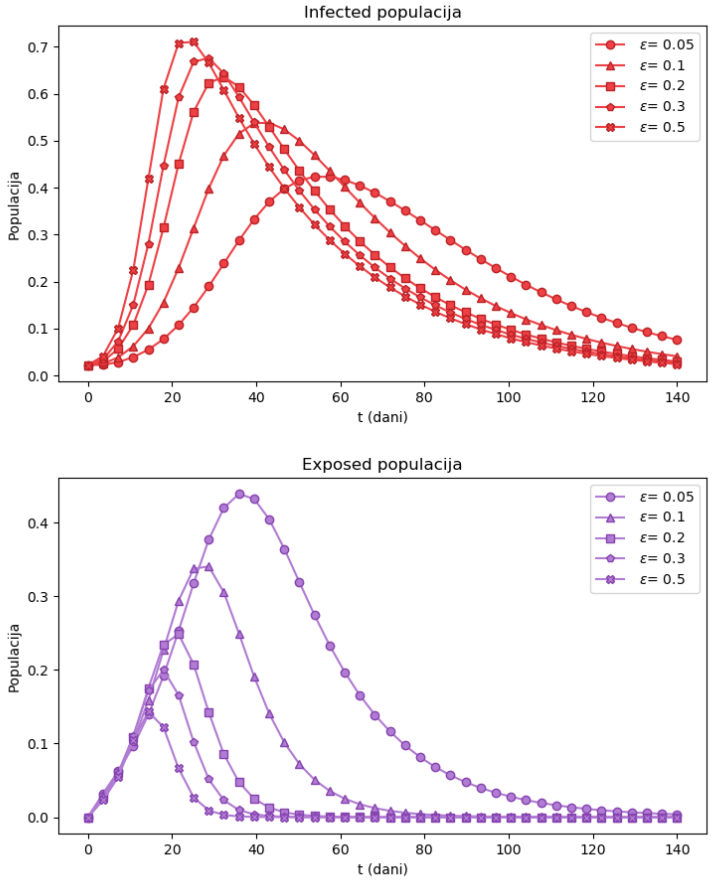
$$\frac{\partial i_t}{\partial t} = \epsilon e_t - \mu i_t \tag{2.13}$$

$$\frac{\partial r_t}{\partial t} = \mu i_t \tag{2.14}$$

Kao i u slučaju SIR modela, zbir parcijalnih izvoda po vremenu daje nulu, odnosno populacija je konstantna. Na slici 2.8 predstavljena je vremenska evolucija epidemije za $N = 45$, $\beta = 0.01$, $\mu = 0.03$, $\epsilon = 0.1$, i za jednu zaraznu osobu u početnom trenutku. Postojanje *Exposed* odeljka je pomerilo i smanjilo pik *Infected* krive, iako je ukupan broj zaraženih ostao isti kao i u SIR modelu.



Slika 2.8: Vremenska evolucija epidemije za SEIR model



Slika 2.9: Vremenska evolucija epidemije za SEIR model za različite vrednosti ϵ

Kako se krive *Infected* i *Exposed* odeljaka menjaju sa promenom parametra ϵ prikazano je na slici 2.9 [3]. Vidi se da se povećanjem parametra ϵ maksimum krive inficiranih pomera ka kasnijim danima, ali se i spušta, odnosno kriva se "zaravnjuje". Ovo ukazuje na važnost uvođenja dodatnog *Exposed* odeljka - direktno se utiče na verovatnoću preopterećenja zdravstvenog sistema ali i na trajanje uvedenih restriktivnih mera kao što je karantin. Manji pik smanjuje pritisak na zdravstveni sistem, dok duže trajanje implicira da period socijalnog distanciranja treba trajati duže. U slučaju ovakvog SEIR modela koji ne uključuje promene u populaciji u vidu stopa rađanja i smrti, izraz za osnovni reproduktivni broj R_0 je isti kao i za SIR model ($R_0 = \beta/\mu$), a do njega se dolazi koristeći pristup matrice sledeće generacije (*"next generation matrix"*) detaljnije opisanog u dodatku **A**.

Daljim dodavanjem odeljaka i veza između njih može se razraditi još modela (npr. SEIRS model koji uključuje "privremeni" imunitet gde osoba posle oporavka ponovo postaje podložna, ili SEIRD- gde se eksplicitno uključuju smrtni slučajevi) [4]. Na sličan način mogu se uključiti i asimptotski slučajevi, kao i hospitalizovani. U par narednih poglavlja zadržaćemo se samo na SIR i SEIR modelima koje ćemo primeniti na "realnu" mrežu kontakata (tada "well-mixed" pretpostavka ne važi) i time pokazati kako struktura mreže utiče na dinamiku epidemije, dok u poslednjem odeljku imamo model koji uključuje asimptotske, umrle i hospitalizovane slučajeve.

[3] "ODEINT SEIR model" u dodatku **B**

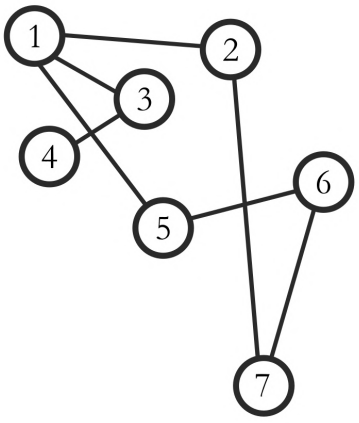
3 Pregradni modeli na kompleksnim mrežama

U ovom odeljku fokusiraćemo se samo na SIR i SEIR modele i njihovoj primeni na kompleksnim mrežama, i to konkretno na mreži dobijenoj iz podataka o interakcijama učesnika na "Godišnjoj Konferenciji o Bolničkim Infekcijama" u Francuskoj [10]. Simulacija modela na mrežama omogućava razmatranje sistema za koje ne važi pretpostavka "dobrog mešanja", odnosno populacija u kojima nije svaka osoba povezana sa svim ostalim, što daje realističniju sliku dinamike epidemije. Pre implementacije na mrežu sa konferencije uvešćemo osnovne pojmove iz teorije mreža i proveriću tačnost koda upoređivanjem rezultata dobijenih korišćenjem diferencijalnih jednačina u prethodnom odeljku, sa rezultatima simulacije na mreži koja zadovoljava uslov "dobrog mešanja". Svi kodovi korišćeni u radu priloženi su u dodatku **B**.

3.1 Osnovni pojmovi i provera koda

3.1.1 Osnovni elementi teorije mreža

Matematički, mreže su opisane kao grafovi: kolekcije tačaka - *čvorova* spojenih setom konekcija - *veza*, a svaka veza predstavlja prisustvo interakcije ili odnosa između čvorova koje spaja. Čvorovi predstavljaju komponente sistema, i u zavisnosti od njegove prirode mogu biti različite stvari. Na primer, u biološkim mrežama čvorovi mogu biti molekuli, proteini, aminokiseline, i tako dalje, ali i procesi, u zavisnosti od prirode problema [38]. Takođe, veze između čvorova mogu biti usmerene ili neusmerene, a mogu imati i znak (pozitivni ili negativni) i težinu [2]. U našem slučaju (socijalne mreže) čvorovi su pojedinci, a veze između njih su njihovi kontakti, sve su neusmerene i nemaju znak, a ni težinu. Grafička reprezentacija ovakve mreže predstavljena je na slici 3.1.



Slika 3.1: Primer neusmerene mreže(grafa) sa 7 čvorova

Veličinu grafa određuje broj čvorova N, i kompaktan način za predstavljanje svih veza koji u njemu postoje je *matrica povezanosti* A čiji su elementi $a_{ij} = 1$ ukoliko postoji veza između čvorova i i j , a $a_{ij} = 0$ u suprotnom[4]. U slučaju neusmerenih grafova matrica povezanosti je simetrična, dok je u slučaju neusmerenih asimetrična. Može se definisati i put $\mathcal{P}_{i_0,i_n}$ koji povezuje čvorove i_0 i i_n nizom različitih veza $\{(i_j, i_{j+1})\}$ gde $j = 0, ..., n-1$; broj veza koji smo prošli da bi spojili i_0 i i_n je n i naziva se dužinom puta [2, 17]. Zatvoren put $i_0 \equiv i_n$ zove se *petlja*. Graf je *povezan* ukoliko postoji put koji povezuje bilo koja dva čvora u njemu. Ukoliko je mreža, odnosno graf nepovezan njen povezani podgraf naziva se *komponentom* $\mathcal{C}$ grafa, a *najvećom komponentom* naziva se komponenta čija se veličina skalira kao broj čvorova u celom grafu. Iz epidemiološke

[4]ako veze imaju težinu $a_{ij} \in [0, 1]$ u zavisnosti od težine veze

perspektive, infekcija koja se javi u najvećoj komponenti može, u principu, zaraziti makroskopski deo grafa, dok infekcija koja se javi izvan najveće komponente može zaraziti samo konačan broj čvorova [6].

Što se tiče puteva i rastojanja između čvorova, može se uvesti još i pojam *dužine najkraćeg puta* l_{ij} između dva čvora i i j koji je definisan kao dužina najkraćeg puta koji ih spaja. Maksimalna vrednost l_{ij} svih parova određuje *prečnik* mreže, a srednji najkraći put $\langle l \rangle$ je srednja vrednost l_{ij} po svim parovima čvorova i, j [2] .

Za neusmerene[5] mreže svakom čvoru može se pripisati tzv. "stepen čvora" k koji predstavlja broj veza koje izlaze iz tog čvora, odnosno $k_i = \sum_j a_{ij}$. Onda možemo definisati *distribuciju stepena* $P(k)$ kao verovatnoću da nasumično izabran čvor ima stepen k, ili u slučaju konačnih mreža, deo čvorova u grafu koji imaju stepen tačno jednak k [2],[17]. Ova veličina nam daje neku opštu sliku o strukturi mreže. Korisno je i definisati momente ove raspodele $\langle k^n \rangle = \sum_k k^n P(k)$: prvi moment $\langle k \rangle = 2L/N$ je dvostruki odnos broja veza L i broja čvorova N daće informaciju o gustini mreže. Mreža je *retka* ukoliko broj veza L raste najviše linearno sa veličinom mreže N, u suprotnom mreža je *gusta*.

Korelacije između stepena dva čvora mogu se opisati verovatnoćom $P(k'|k)$ da je čvor stepena k povezan sa čvorom stepena k' [2]. Mreža je *nekorelisana* ako ova verovatnoća ne zavisi od izvornog stepena k. Međutim, ispostavlja se da je procena ove verovatnoće za realne mreže nezgodna zbog njihove konačne veličine, i jednostavnije je za meru korelacija uzeti srednji stepen najbližih suseda čvorova stepena k: $\overline{k}_{nn}(k) = \sum_{k'} k' P(k'|k)$ [23]. Za nekorelisane mreže $\overline{k}_{nn}(k) = \langle k^2 \rangle / \langle k \rangle$ ne zavisi od k - zavisnost ove veličine od k karakteristično je za mreže gde postoji korelacija stepena. Analiza empirijskih mreža u odnosu na njihove korelacije stepena dovela je do podele na dve grupe korelisanih mreža [17]: *asortativne* i *disortativne*. Kod asortativnih mreža $\overline{k}_{nn}(k)$ je rastuća funkcija što ukazuje na to da čvorovi većeg stepena teže da se povezuju sa drugim čvorovima većeg stepena, dok se čvorovi nižeg stepena povezuju sa drugim čvorovima malog k. Suprotno, disortativne mreže ispoljavaju opadajuće $\overline{k}_{nn}(k)$ što implicira da se čvorovi višeg stepena povezuju sa čvorovima nižeg stepena i obrnuto.

Još jedan od fenomena koji se javlja u kompleksnim mrežama je klasterovanje. Klasterovanje se odnosi na relativnu sklonost da dva čvora budu povezana ako imaju zajedničkog suseda. Koeficijent klasterovanja C daje meru do koje čvorovi u grafu teže da se "klasteruju", i formalno je definisan kao odnos broja petlji dužine 3 (tzv. trougli) u mreži i broja povezanih "trojki" (3 čvora povezanih dvema vezama) [2]. Može se definisati i lokalna mera klasterovanja c_i i to kao odnos broja veza između svih suseda čvora i i maksimalnog mogućeg broja veza, što u stvari daje verovatnoću da su dva suseda čvora i susedi međusobno. Srednja vrednost c_i svih čvorova u mreži daje srednje klasterovanje mreže $\langle c \rangle$, a spektar klasterovanja $\overline{c}(k)$ definiše se kao srednji koeficijent klasterovanja čvorova stepena k i zadovoljava $\langle c \rangle = \sum_k P(k)\overline{c}(k)$ [24].

Poslednji pojam koji ćemo ukratko diskutovati je *centralnost*, koji daje relativnu važnost čvora unutar mreže, što je naročito bitan problem u analizi socijalnih mreža [25]. Postoje različite definicije centralnosti u zavisnosti od različitih indikatora strukturne važnosti čvorova. Jedan, i najjednostavniji od indikatora je stepen čvora koji se zato naziva i *centralnost stepena*: što je veći stepen čvora to on ima veći uticaj(centralnost) u mreži. Mogu se definisati i centralnosti u odnosu na najkraće puteve između čvorova kao npr. takozvana *closeness centrality* $\mathcal{C}_i$ definisana kao inverz srednje vrednosti najkraćih puteva iz čvora i ka svim ostalim čvorovima u mreži - pomoću nje definišemo čvor kao centralni ako se nalazi (u srednjem) blizu svih ostalih čvorova mreže. Još jedan način definicije centralnosti u odnosu na najkraće puteve je tzv. *betweenness centralnost* b_i definisana kao broj najkraćih puteva između bilo koja dva čvora koja prolaze kroz čvor i, i daje potpunu drugačiju perspektivu od $\mathcal{C}_i$:

[5] Za usmerene mreže razlikujemo *in-degree* k_i^{in} kao broj veza koji "ulazi" u i i *out-degree* k_i^{out} kao broj veza koji "izlazi" iz i

ona daje meru centralnosti u smislu kontrole informacija koja teče između različitih čvorova, pretpostavljajući da ova informacija teče kroz najkraće puteve.

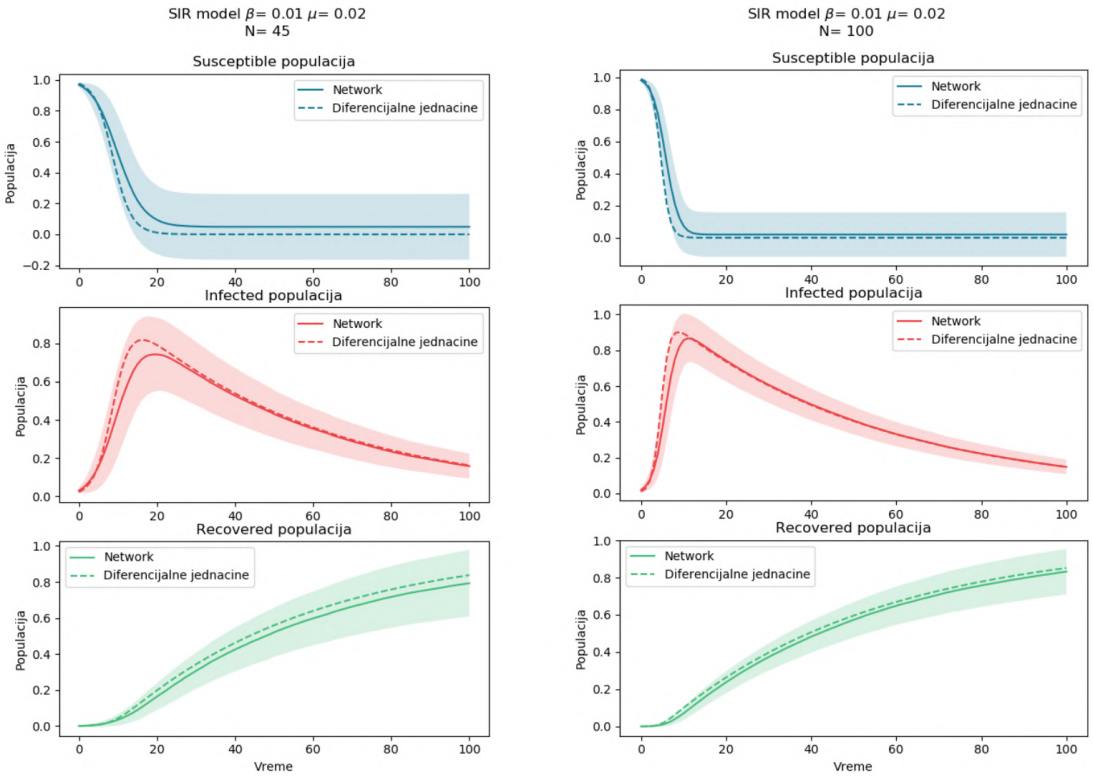
Na kraju, treba napomenuti da realne mreže mogu ispoljiti više nivoe arhitekture koje je teško opisati tačnim brojevima i vrednostima, na primer, mnoge mreže poseduju *strukturu zajednica* u kojoj različiti setovi čvorova koji se nazivaju *zajednicama* ili *modulima* imaju relativno veliku gustinu unutrašnjih konekcija, ali su međusobno slabo povezani [26].

3.1.2 Provera koda

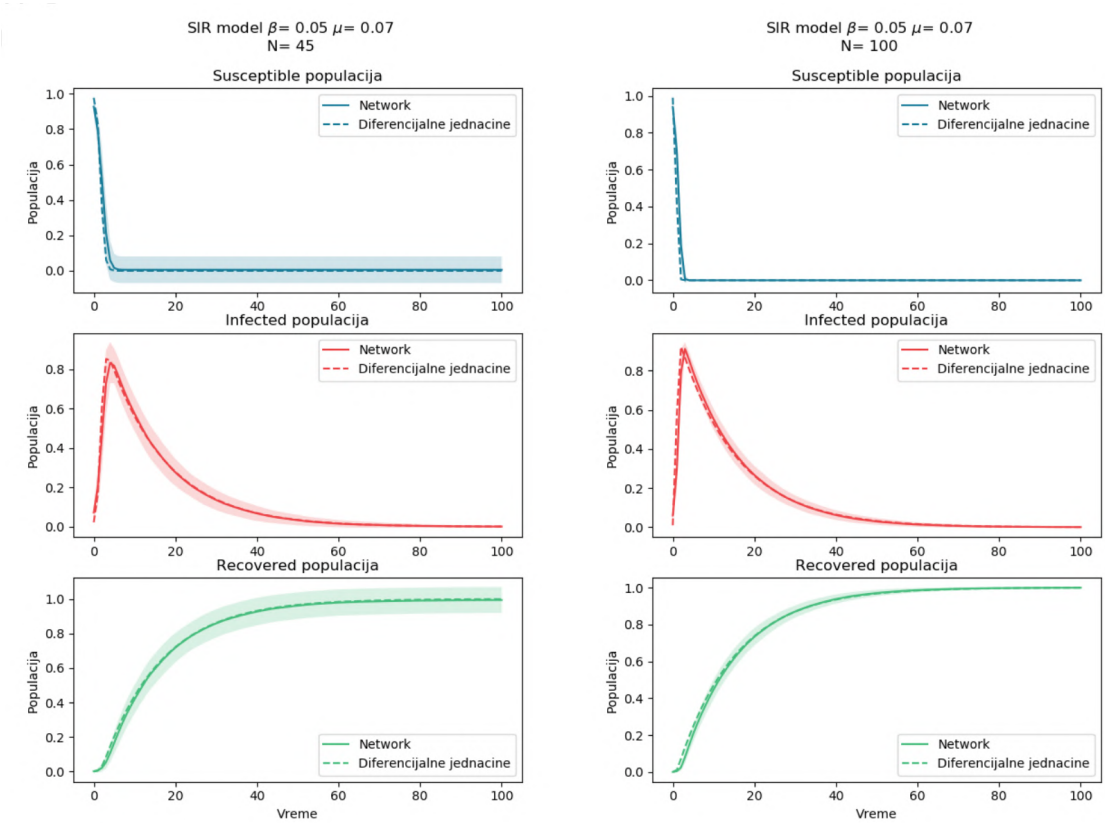
Pre nego što krenemo sa simulacijama na mreži sa konferencije potrebno je testirati kod na mreži koja ispunjava uslov "dobro izmešane" populacije i uporediti ga sa rezultatima dobijenim koristeći diferencijalne jednačine SIR i SEIR modela, kod kojih je, kao što smo rekli na početku, ovo osnovna pretpostavka. U kontekstu naše mreže, gde nam čvorovi predstavljaju osobe, "dobro izmešana" populacija prosto znači da je svaki čvor povezan sa svim ostalim čvorovima (npr. ako imamo 100 čvorova, stepen svakog čvora bio bi $k = 99$). U Python-u se pomoću paketa NetworkX može generisati "Erdős–Rényi" mreža[6] proizvoljne veličine. Mi želimo da ta verovatnoća bude jednaka 1 kako bi svaki čvor bio povezan sa svim ostalim - dakle u simulaciji koristimo $p = 1$. Svi čvorovi sa kojima je čvor i direktno povezan nazivaju se njegovim susedima, i u kontekstu širenja zaraze svaki sused čvora i predstavlja (socijalni) kontakt osobe i. Smatramo da u toku jednog dana svaka osoba bude u kontaktu sa svim svojim "susedima". U početnom trenutku ("nultom danu") zarazimo jednu nasumičnu osobu, tj. čvor, dok su svi ostali čvorovi podložni, tj. *Susceptible* [7]. Od narednog trenutka, tj. sledećeg dana krećemo da proveravamo kontakte čvorova - ako su bili u kontaktu sa zaraženim čvorom oni i sami postaju zaraženi nekom verovatnoćom β (odnosno postaju izloženi verovatnoćom β ako se radi o SEIR modelu, a zaraženi postaju nekom verovatnoćom ϵ u nekom od narednih dana). Ako je čvor u aktuelnom trenutku zaražen postaje *Recovered* nekom verovatnoćom μ. Pratimo stanje za 100 dana i za svaki dan upisujemo koliko čvorova je u kom stanju. Postupak ponovimo za 500 simulacija i krajnji rezultat predstavimo kao srednju vrednost svih simulacija. Kao grešku uzimamo standardnu devijaciju. Rezultati su predstavljeni na slikama 3.2, 3.3 i 3.4, a korišćeni kodovi dati su u dodatku **B** ("SIR/SEIR na mreži").

[6] Erdős–Rényi model daje jedan od načina generisanja *random* mreža kod kojih se veze između čvorova formiraju nasumično. Međutim, kada se priča o *random* mrežama skoro uvek se misli samo na Erdős–Rényi model. Postoje 2 jako slične varijante Erdős–Rényi modela: prvi generiše mrežu,tj graf $G(N, M)$ sa N čvorova i M nasumično postavljenih veza, dok drugi generiše graf $G(N, p)$ gde je svaki par N čvorova povezan verovatnoćom p [2]. NetworkX funkcija koja generiše "Erdős–Rényi" mrežu koristi ovu drugu varijantu.

[7] u Python-u mrežu generisanu pomoću NetworkX paketa "prepišemo" u objekat *dictionary* koji omogućuje jednostavno upisivanje i baratanje svim vrednostima od interesa kao što su stanja čvorova

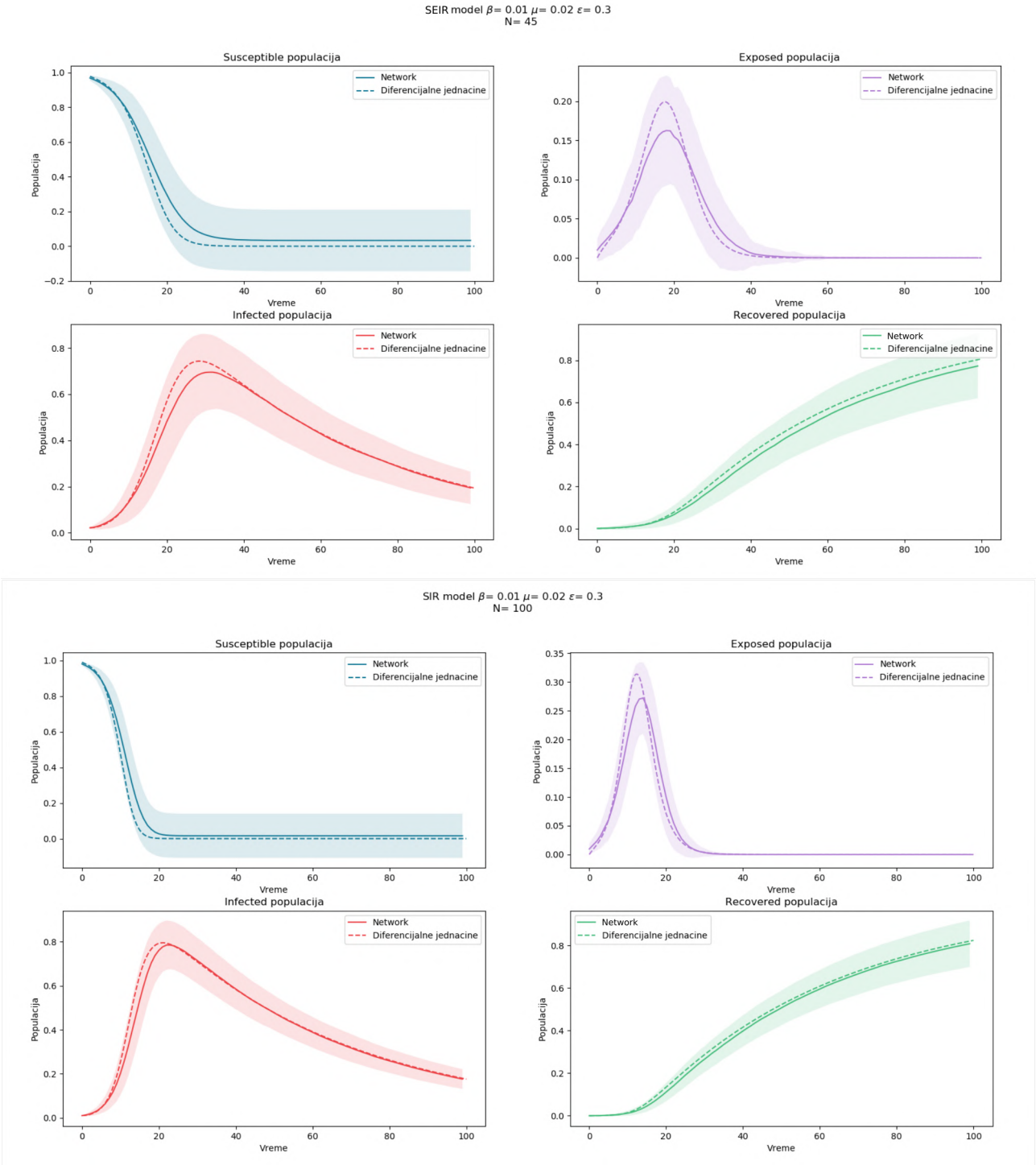


Slika 3.2: Rezultati simulacija na mreži u poređenju sa rešenjima diferencijalnih jednačina SIR modela za broj čvorova $N = 45$ (levo) i $N = 100$ (desno)



Slika 3.3: Rezultati simulacija na mreži u poređenju sa rešenjima diferencijalnih jednačina SIR modela za veće vrednosti parametara i broj čvorova $N = 45$ (levo) i $N = 100$ (desno)

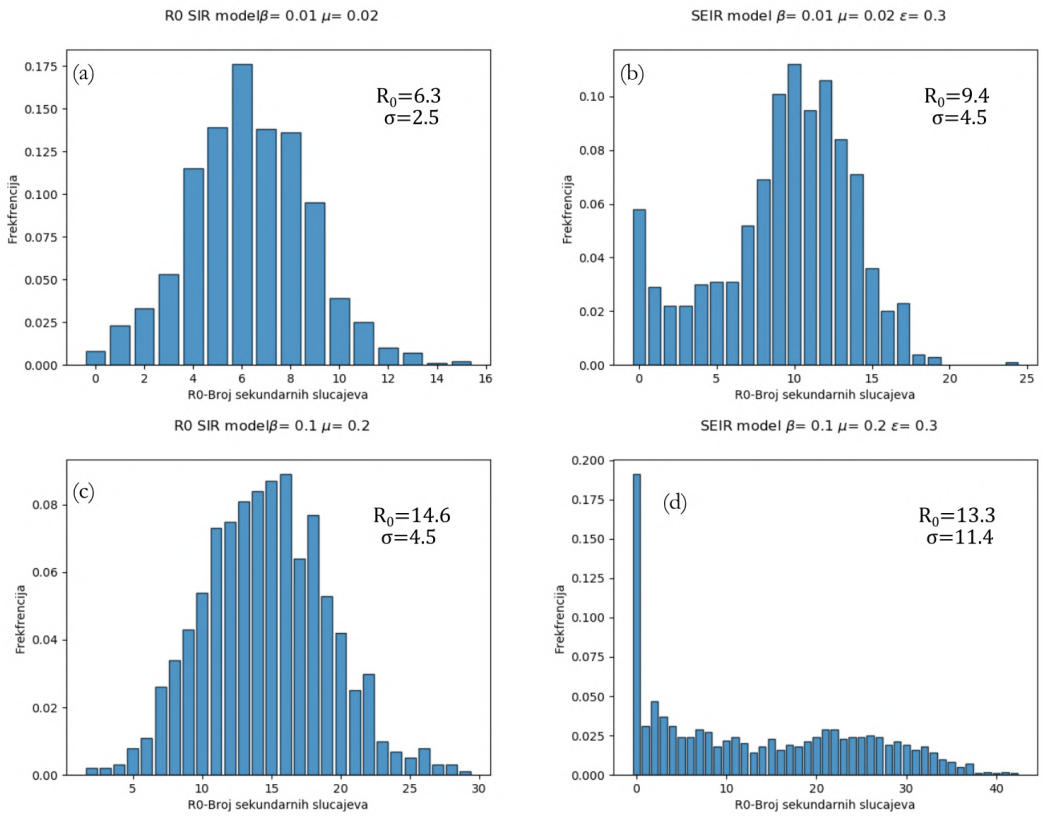
Primećuje se da slaganje nije potpuno, ali da je bolje za što veći broj čvorova N. Razlog tome je što su same diferencijalne jednačine rešenje srednjeg polja i najbolje slaganje se dobija kada $N \to \infty$. Dakle, što je veće N to su bolja slaganja rezultata simulacija sa rešenjem srednjeg polja i standardna devijacija je manja. Uočava se skoro potpuno slaganje za veće vrednosti parametara(slika 3.3) - kada su parametri mali, dolazi do izražaja "nesavršenost" Erdős–Rényi mreže jer se bolest sporije širi i može doći do češćeg prekida lanca širenja. Takođe, povećanje broja simulacija nije dovelo do poboljšanja slaganja ili smanjenja standardne devijacije. Isti argumenti važe i za slučaj SEIR modela, prikazanog na slici 3.4.



Slika 3.4: Rezultati simulacija na mreži u poređenju sa rešenjima diferencijalnih jednačina SEIR modela za broj čvorova $N = 45$ (gore) i $N = 100$ (dole)

3.1.3 Nalaženje R_0 iz simulacije na mreži

Osnovni reproduktivni broj predstavlja jednu od najvažnijih veličina u epidemiologiji, i videli smo da postoji način da se on analitički odredi iz izraza diferencijalnih jednačina - međutim, nas sada interesuje kako bi on mogao da se odredi numerički, iz simulacije. R_0 je definisan kao srednji broj novih slučajeva infekcije nastalih od jedne konkretne osobe u populaciji koja se sastoji samo od podložnih (*Susceptible*) osoba [12]. U skladu s time, u svakoj simulaciji pratimo "nultog pacijenta" i koliko je on ljudi zarazio dok je i sam bio zaražen(tj. dok je u *Infected* odeljku) i na kraju usrednjimo rezultat po svim simulacijama[8]. Rezultati su na prvi pogled neočekivani. Naime, kao što smo videli u prethodnim sekcijama za SIR (i za SEIR model) R_0 zavisi samo od odnosa parametara β, koji predstavlja verovatnoću da se osoba zarazi, i μ koji predstavlja stopu oporavka ($R_0 = \beta/\mu$) - tako da drastično različite vrednosti ovih parametara daju istu vrednost R_0 ukoliko im je odnos isti. Konkretno, dva seta parametara $\beta = 0.01$, $\mu = 0.02$ i $\beta = 0.1$, $\mu = 0.2$ iz analitičkog izraza daju $R_0 = 0.5$. Numeričkim rešavanjem za 1000 simulacija na mreži dobijaju se sledeći rezultati (slika 3.5) : Za set parametara $\beta = 0.01$, $\mu = 0.02$ SIR model dao je $R_0^n = 6.3$ standardne devijacije $\sigma = 2.5$ a za $\beta = 0.1$, $\mu = 0.2$ $R_0^n = 14.6$ $\sigma = 4.5$ što se apsolutno ne poklapa sa analitičkim rezultatom. Dodatno, dobijene vrednosti se čak ni ne slažu sa SEIR modelom (a videli smo da su analitički izrazi R_0 oba modela isti), u njegovom slučaju za $\beta = 0.01$, $\mu = 0.02$ $R_0^n = 9.4$ $\sigma = 2.5$, a $\beta = 0.1$, $\mu = 0.2$ $R_0^n = 13.3$ $\sigma = 11.4$. Pored toga, sa Slike 3.5. primećujemo da su R_0 vrednosti dobijene iz pojedinačnih simulacija za SIR model (Slika 3.5(a),(c)) distribuirane po Gausovoj krivoj, dok to nije slučaj kod SEIR modela kod koga se javlja često nagomilavanje na $R_0 = 0$. Razlog za to je kompleksnija dinamika SEIR modela i postojanje čekanja u *Exposed* fazi.



Slika 3.5 Dobijene vrednosti R_0 iz simulacija (a) SIR model za parametre $\beta = 0.01$, $\mu = 0.02$ (b) SEIR model za parametre $\beta = 0.01$, $\mu = 0.02$, $\epsilon = 0.3$ (c) SIR model za parametre $\beta = 0.1$, $\mu = 0.2$ (d) SEIR model za parametre $\beta = 0.1$, $\mu = 0.2$, $\epsilon = 0.3$

[8]Kodovi u dodatku **B**: "R_0 na mreži SIR/SEIR"

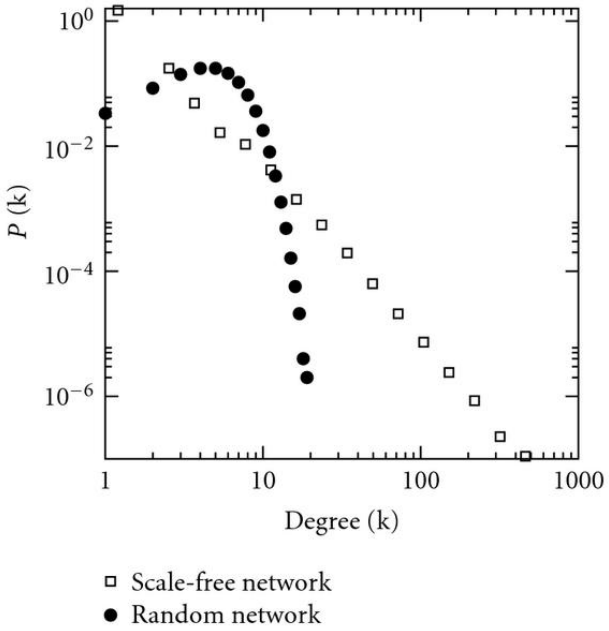
Takođe je uočeno da u ovom slučaju, R_0 zavisi i od broja čvorova u mreži i od vrednosti parametra ϵ, što opet nije bio slučaj kod analitičkog rešenja. Zašto se sve ovo dešava? Ispostavlja se da postoji više metoda za određivanje ovog broja, i da se rezulatati istih mogu međusobno razlikovati [27]. Iako se na prvi pogled R_0 čini kao jednostavna mera za određivanje transmisione dinamike infekcija treba biti pažljiv što se tiče njene interpretacije, i treba imati na umu više mogućih definicija ove veličine. Ovo je kompleksna veličina čije izračunavanje zavisi od velikog broja odluka u procesu epidemiološkog modeliranja, a na kraju krajeva, njena prava vrednost može se znati tek posle medicinskih i mikrobioloških ispitivanja patogena, zajedno sa socijalnim i statističkim istraživanjima ponašanja populacije.

3.2 Evolucija epidemije na realnoj mreži

Za sada smo uradili simulaciju SIR i SEIR modela na *random* mrežama, koje zadovoljavaju uslov homogenog mešanja (verovatnoća formiranja veze između čvorova je $p = 1$). Generalno, raspodela stepena k random mreža ima Poisson-ovu distribuciju (odnosno Gauss-ovu): najveći broj čvorova ima prosečnu povezanost, dok mali broj čvorova ima jako veliku ili jako malu povezanost[9]. Međutim, to nije slučaj u realnim mrežama: one su uglavnom "scale-free" - njihova raspodela stepena čvorova pokorava se nekom stepenom zakonu (makar asimptotski); tj. deo čvorova $P(k)$ koji ima vrednost stepena k ima sledeću zavisnost:

$$P(k) \sim k^{-\gamma}$$

gde je γ tipično u opsegu $2 < \gamma < 3$ [2]. Razlika u raspodeli stepena za *random* i *scale-free* mreže predstavljena je na slici 3.6.



Slika 3.6. Primer distribucije stepena za *random* i *scale-free* mrežu na loglog skali [13]

Ovo jasno implicira veoma drugačiju strukturu mreže koja ne zadovoljava pretpostavku dobro izmešane populacije, što će imati značajan uticaj na evoluciju i dinamiku epidemije. Ono što je karakteristično za ovakve

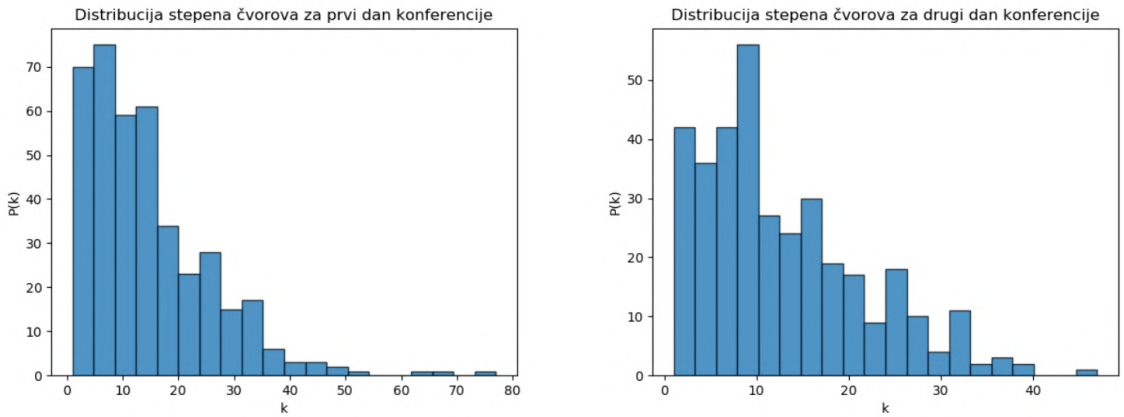
[9]Mreža na kojoj smo proveravali simulaciju bila je *random* graf sa verovatnoćom formiranja veza između čvorova $p = 1$ kako bi ostvarili uslov homogenog mešanja, pa je i svaki čvor imao jednak stepen $k = N - 1$.

mreže je pojava takozvanih *hub*-ova - čvorova sa značajno velikim stepenom povezanosti, što se može primetiti iz same distribucije $P(k)$. Prisustvo *hub*-ova u velikim *scale-free* mrežama ima za posledicu da se patogen može momentalno proširiti kroz celu populaciju ukoliko ovakav čvor postane zaražen; takođe, čak i da je patogen biološki slab (malo infektivan, velika brzina oporavka itd.) ukoliko uspe da zarazi *hub*, on može opstati u populaciji. Ovo se ne dešava u slučajnim mrežama (naročito ne pod pretpostavkom homogenog mešanja) baš zato što svi čvorovi imaju slične stepene povezanosti, odnosno ne javljaju se *hub*-ovi.

Diskusija iznad je bila sasvim generalna, a sada ćemo se fokusirati na konkretan slučaj. Posmatramo realnu mrežu dobijenu iz podataka o interakcijama učesnika na "Godišnjoj Konferenciji o Bolničkim Infekcijama" u Francuskoj. Od 1200 učesnika konferencije 403 je pristalo da nosi posebne bedževe opremljene RFID(radio-frequency identification) uređajima koji kada su na razdaljini od 1-2m međusobno razmenjuju "pakete" informacija koji sadrže specifične identifikatore za svaki uređaj. Svaki učesnik označen je jedinstvenim identifikacionim brojem, tj. ID tagom. Pod ovim uslovima, kada su učesnici okrenuti licem u lice, prate se socijalne interakcije između njih i kasnije se iz prikupljenih podataka može formirati mreža kontakata [10, 11]. Ovo je pogodno za razmatranje bolesti koje se prenose kapljičnim putem poput kijanja ili kašljanja, ili onih koji se prenose direktnim fizičkim kontaktom. Na ovaj način moguće je konstruisati 3 tipa mreža:

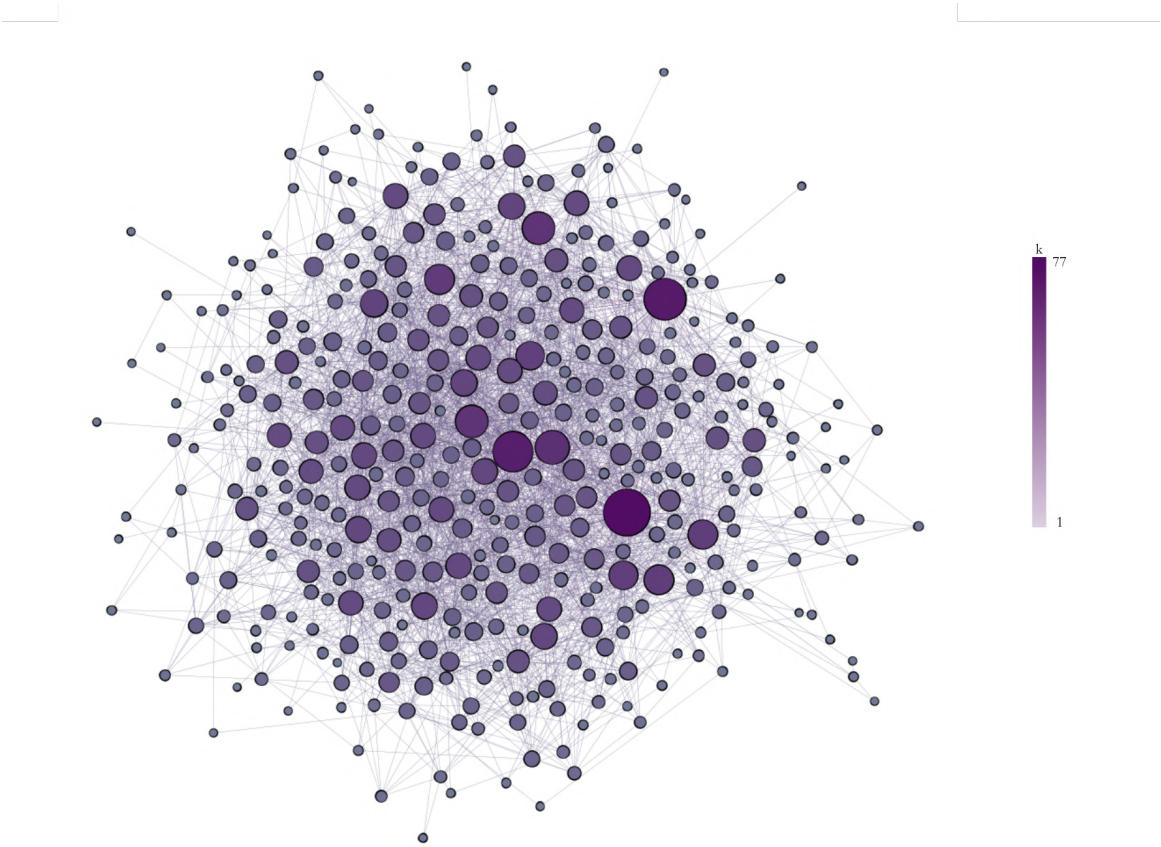
- dinamičku - koja je najrealnija i uključuje težinu veza u zavisnosti od tačnog trajanja interakcija između kontakata, ali i tačan vremenski redosled interakcija što je od posebne važnosti za širenje infekcije [2, 14].
- heterogenu - koja sadrži samo težinu veza, ali ne i vremenski redosled interakcija
- homogenu - sadrži samo težinu veza, ali je svaka veza otežinjena jednakom težinom, što odgovara srednjem trajanju kontakta između 2 osobe koje su se srele istog dana u heterogenoj mreži. Ovakva konstrukcija odgovara situaciji kada bi smo pitali svakog učesnika s kim je sve bio u kontaktu, i koliko je, otprilike, taj kontakt trajao.

U daljoj analizi mi se ograničavamo na homogeni slučaj, a mreža kontakata za prvi dan predstavljena je na slici 3.8, a distribucije stepena za prvi i drugi dan konferecnije na slici 3.7.



(a) (b)

Slika 3.7. Distribucija stepena k za prvi dan konferencije (a) i drugi dan (b).

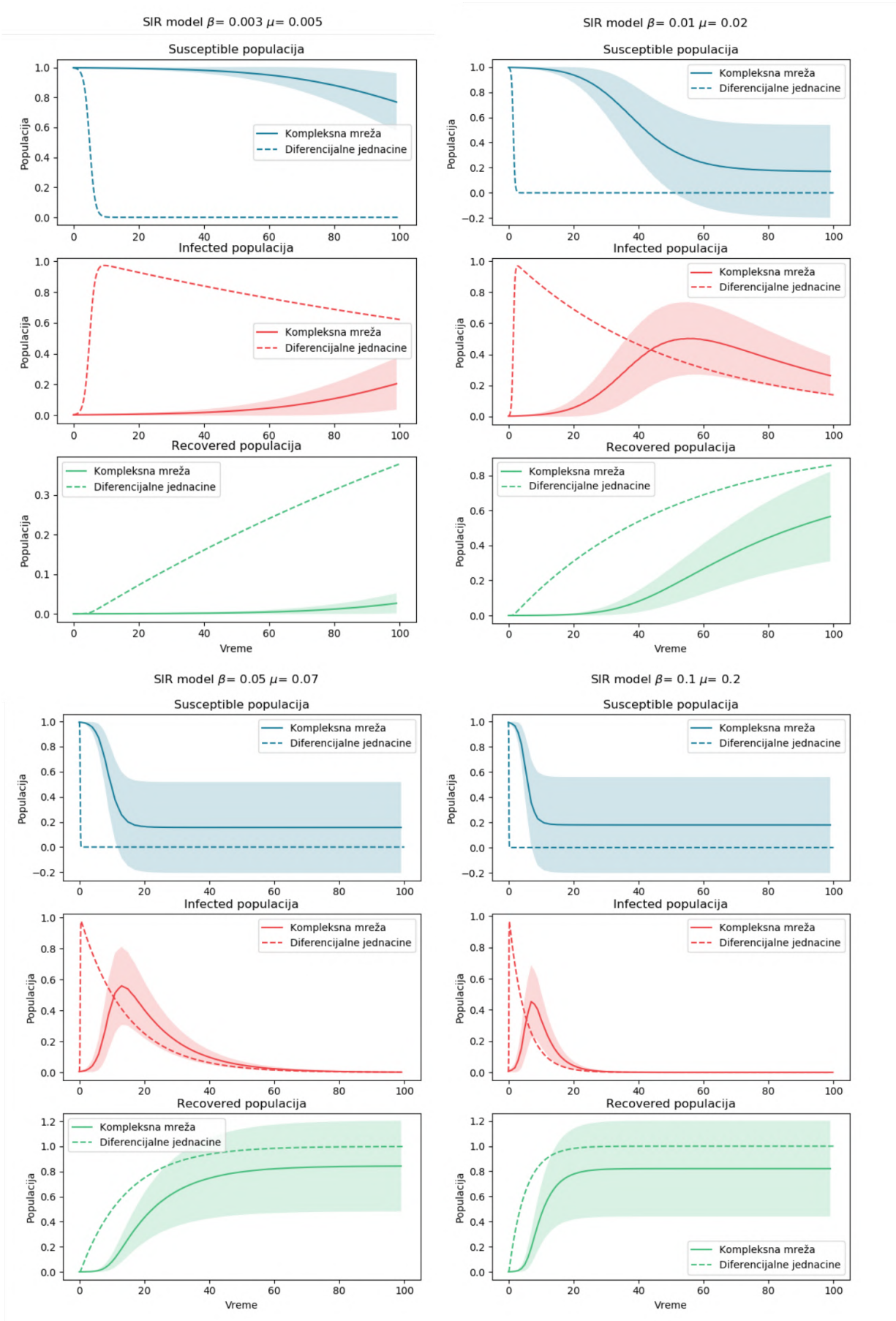


Slika 3.8. Mreža kontakata za prvi dan konferencije. Veličina čvorova i intenzitet njihove boje predstavljaju veću ili manju vrednost njihovog stepena-čvorovi sa velikim k su veći i intenzivnije boje.

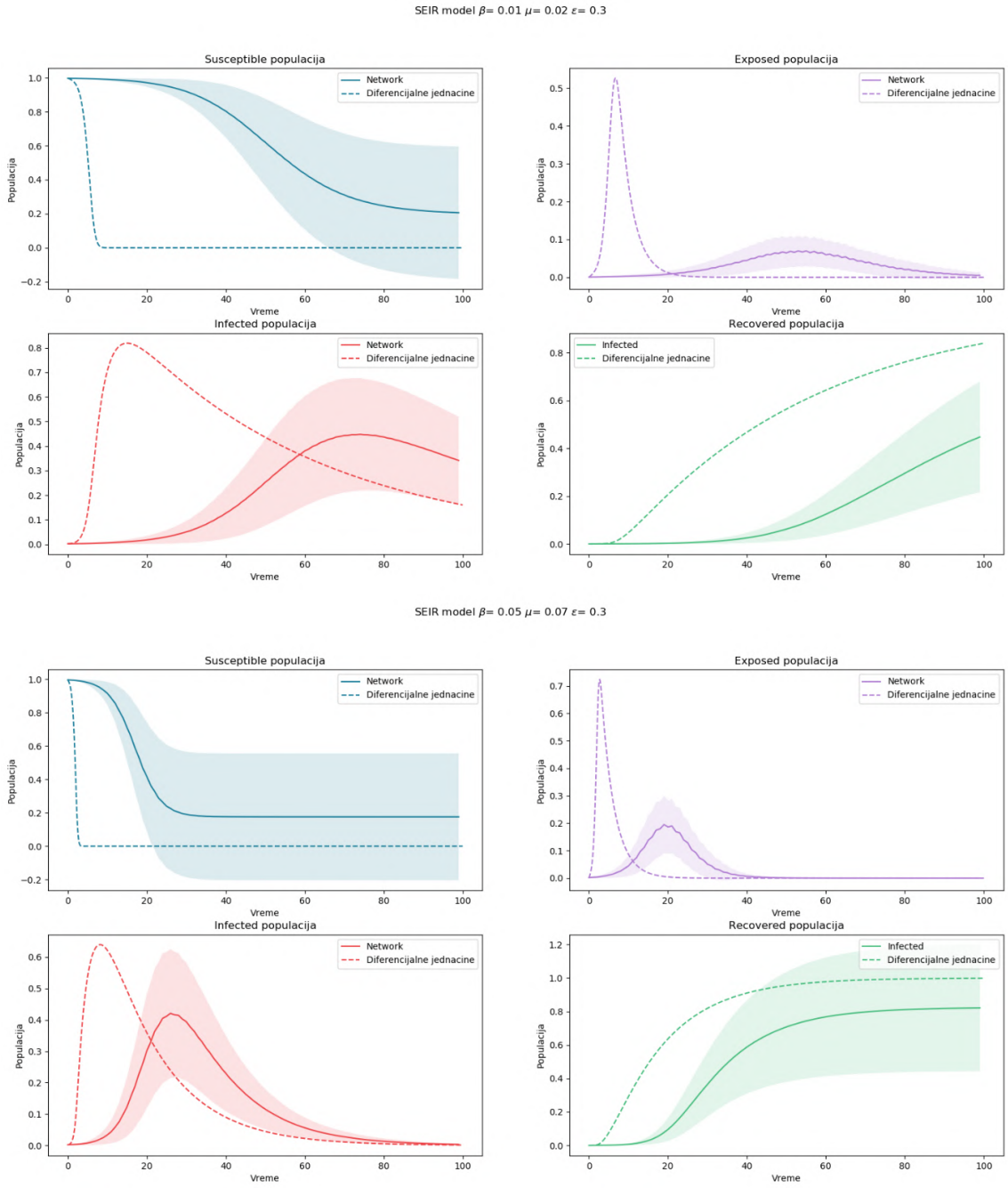
Merenja socijalnih interakcija i konstrukcija mreža izvršena su za 2 dana trajanja konferencije. Kako želimo da pratimo evoluciju epidemije u dužem vremenskom periodu (100dana) možemo iz postojeće 2 mreže sintetički generisati ostalih 98. To se radi tako što izmešamo ID tagove ali tako da očuvamo "preference" učesnika, odnosno korelacije između njihovih socijalnih aktivnosti i obrazaca kontakata. Pored toga što više nije "dobro izmešana", naša mreža je sada i dinamička - svakog dana se menjaju veze između čvorova, odnosno kontakti osoba. Zbog ovakvog načina mešanja distribucije stepena k održavaju se u svim ostalim danima.

Na konstruisanim mrežama sada se može izvršiti simulacija SIR i SEIR modela, i dobijene rezultate predstavljamo zajedno sa rezultatima koji bi dala pretpostavka homogenog mešanja za isti broj čvorova i iste parametre. Na slici 3.9 i slici 3.10 predstavljeni su usrednjeni rezultati za 500 simulacija.

Sada se tačno vidi kako to struktura mreže utiče na dinamiku epidemije: konkretno u ovom slučaju, usporava njeno širenje, pik *Infected* odeljka je manji i javlja se dosta kasnije. Drastična razlika primećuje se u slučaju jako malih parametara β i γ gde potpuno homogena populacija od 403 osobe dostigne pik epidemije skoro trenutno, dok "realna" dostiže blagi porast tek posle dva meseca od trenutka pojave "nultog pacijenta". Ova razlika se postepeno smanjuje što su parametri veći, ali je pik i dalje pomeren i niži u slučaju mreže sa konferencije - uzrok tome je što je tada verovatnoća zaraze dovoljno velika da se ona dovoljno brzo proširi i kroz heterogenu mrežu. Uočava se i ranije pomenuto pomeranje i smanjenje pika infekcije u odnosu na SIR model za isto β i γ zbog postojanja *Exposed* odeljka.



Slika 3.9. SIR model na simulaciji epidemije na mreži sa konferencije za različite vrednosti parametara



Slika 3.10. SEIR model na simulaciji epidemije na mreži sa konferencije za različite vrednosti parametara.

Sa priloženih grafika se primećuje izuzetno velika standardna devijacija rezultata simulacija, i obrazloženje za to je sledeće: mreža sa konferencije je vrlo heterogena, a epidemiološke krive se dobijaju kada se usrednje rezultati simulacija sa različitim početnim uslovima - koji su u ovom slučaju čvorovi, tj. ljudi od kojih kreće epidemija ("nulti pacijenti"). Mreža se menja svakog dana i u nekim slučajevima dešava se da putanja bolesti od nultog pacijenta ka ostatku mreže ima tzv. "uska grla" (putanje sa malim brojem čvorova, kratke putanje i sl.) zbog čega ne uspe da se proširi na ostatak mreže i praktično se ugasi u samom početku. Sa druge strane, postoje i *hub*-ovi, koji ako se zaraze mogu dalje zaraziti veoma veliki broj čvorova i tako jako brzo proširiti epidemiju. To će sve dovesti do toga da imamo raspodelu procenata zaraženih u svakom danu koja nije Gausova, što dalje dovodi do velike standardne devijacije, koja sada više ne može bite interpretirana kao greška.

Primećujemo da struktura mreže može imati dramatičan uticaj na obim i brzinu širenja epidemije. Ovo pokazuje da je prilikom modeliranja širenja bolesti neophodno uzeti u obzir heterogenost mreže kontakata.

4 Uticaj restriktivnih mera na dinamiku epidemije na mreži

U ovoj sekciji predstavićemo uticaj restriktivnih mera u vidu socijalnog distanciranja i nošenja maski na dinamiku mreže sa konferencije iz prethodnog odeljka. Restriktivne mere su bitne u sprečavanju širenja zaraze, i što su intenzivnije to je efekat veći, ali je takođe bitan i trenutak njihovog uvođenja, odnosno dan od koga kreću da se poštuju. Ispitaćemo ove dve varijante i na SIR i na SEIR modelu za različite procente populacije koji se pridržavaju mera, a i kako izgleda epidemiološka situacija u zavisnosti od dana njihovog uvođenja.

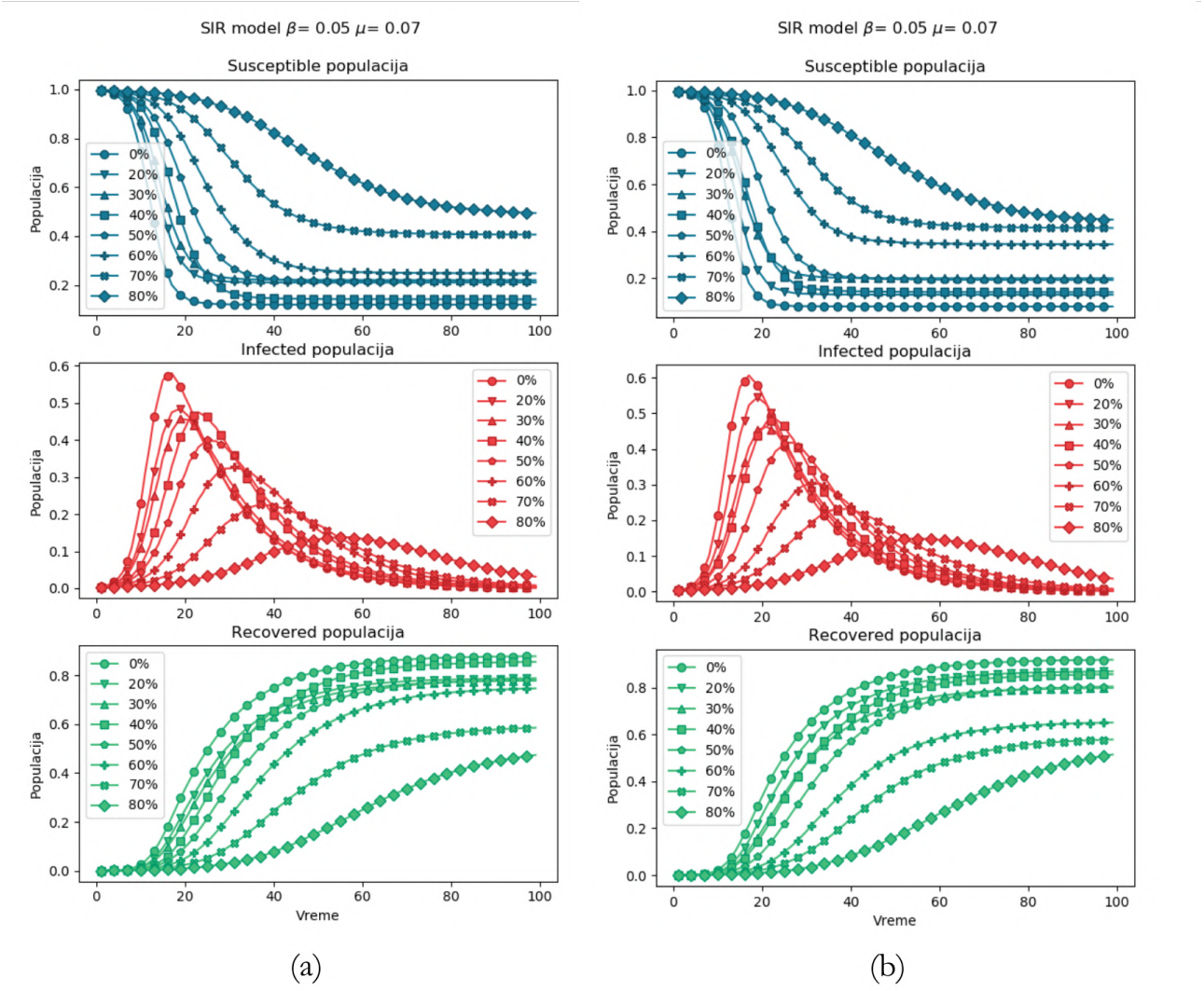
4.1 Socijalno distanciranje

Osnovna ideja za implementiranje mera socijalnog distanciranja u našu simulaciju je smanjenje kontakata između osoba, odnosno "brisanje" nekih veza između čvorova. U kodu, svaki čvor ima listu svojih suseda u svakom trenutku; ono što ćemo mi uraditi kako bi stekli neki kvalitativnu sliku o efektima socijalnog distanciranja je uklanjanje određenog broja elemenata iz tih lista suseda svakog čvora u zavisnosti od procenta "izolacije". Dakle, ako želimo da vidimo šta se desi kada je procenat socijalnog distanciranja, 20% , mi ćemo svakom čvoru izbrisati nasumičnih 20% suseda iz liste, od trenutka kada distanciranje stupa na snagu. Videli smo da se naša mreža menja iz dana u dan, tj. kontakti osoba se menjaju, i da je ona jako heterogena - postoje i jako konektovani čvorovi, ali i jako slabo povezani. Naizgled, postoje 2 varijante simuliranja ovakve situacije i njenog analiziranja: prva je da se prvo izvrši redukovanje mreže u zavisnosti od procenta izolacije, pa da se na toj novoj mreži izvrši željeni broj simulacija (a u svakoj simulaciji se inicijalno zarazi novi, nasumično izabrani čvor); a druga je da se u svakoj simulaciji genriše nova mreža na opisani način (ovde će takođe svaki put biti drugačiji nulti pacijent) i na kraju prikažu usrednjeni rezultati. Drugi način je statistički ispravan - upravo zbog heterogenosti naše mreže, ako bi smo inicirali samo na početku izgubili bi smo "raznovrsnost" mogućih ishoda - mi bi se time ograničili samo na samo jednu nasumičnu, nepoznatu konfiguraciju i takva informacija dobijena iz nje, bez obzira na različite početne uslove(u smislu različitog nultog pacijenta) nije dovoljna. Mi ćemo ipak prikazati rezultate obe varijante. Prvo razmatramo šta se dešava ako se sa socijalnim distanciranjem krene odmah [10], od pojave nultog pacijenta za procente uklanjanja kontakata od 20% - 80% u koracima od po 10%. Prikazane su i epidemiološke krive kada nema distanciranja. Rezultati za 100 simulacija su za SIR model prikazani na slici 4.1., a za SEIR na slici 4.2. Vremenska skala je standardno u danima.

Na prvi pogled, ne primećuju se velike razlike između pomenuta dva načina ubacivanja redukovane mreže kontakata u program. Međutim, ako se bolje pogleda: u slučaju SIR modela , epidemija kao da je imala isti intenzitet maksimuma za procenat redukcije kontakata od 40% od onog za 20% (Slika 4.1(a)) , što je suprotno intuiciji ali svedoči o uticaju heterogenosti mreže, i da u ovom slučaju, uklanjanje nekih kontakata uz odgovarajuće početne uslove može dovesti do većeg proširenja epidemije. Jedan od mogućih razloga što se to dešava je efektivno "skraćenje" puteva širenja epidemije kroz mrežu, što može dovesti do zaražavanja povezanijih čvorova i time do povećanog širenja infekcije na ostale čvorove. Slična situacija uočava se i u slučaju SEIR modela gde redukcija kontakata za 20% i 30% dovodi do višeg pika u odnosu na slučaj kada uopšte nema socijalnog distanciranja. Svakako, smanjenje socijalnih kontakata dovodi do smanjenja *Infected* pika i njegovog pomeranja na kasnije vremenske trenutke, odnosno dolazi do efektivnog "zaravnjenja" krive zaražene populacije (što kao što smo diskutovali na početku značajno utiče na smanjenje preopterećenja zdravstvenog sistema), i to intenzivnije što je veći procenat redukovanja kontakata.

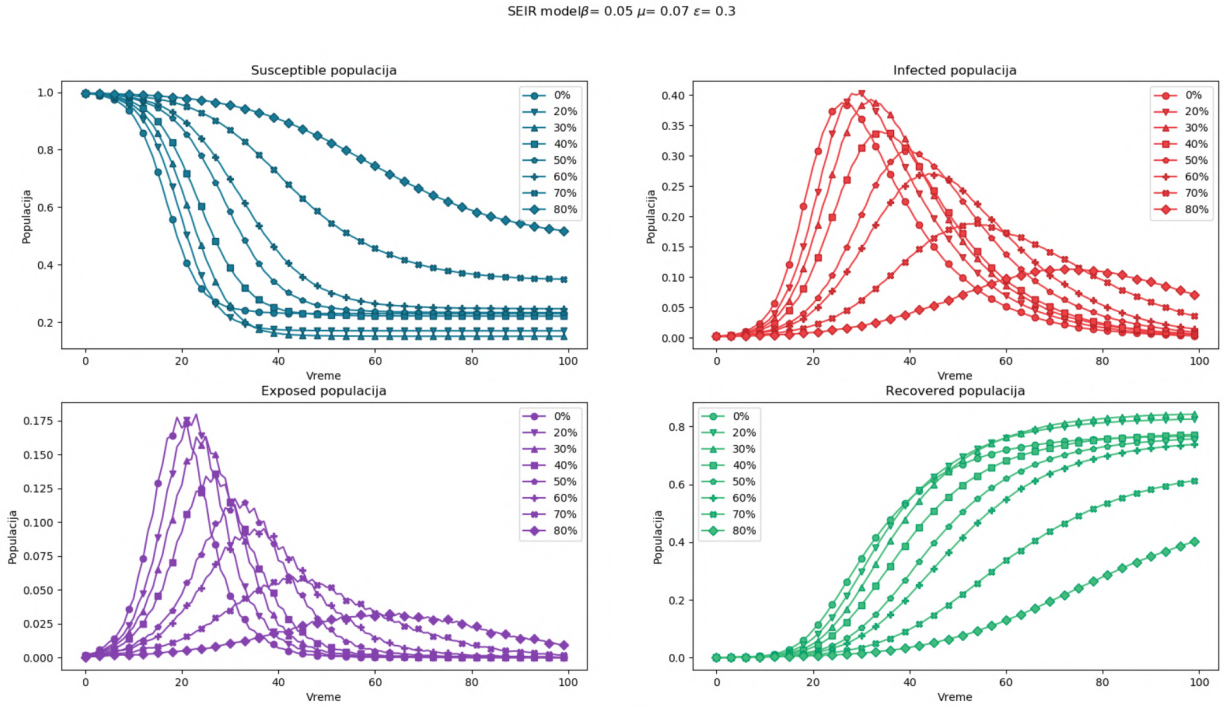
[10] dodatak **B** "Socijalno distanciranje od prvog dana SIR/SEIR"

Na slici 4.3. i slici 4.4. predstavljeni su rezultati simulacija za SIR i SEIR model respektivno, u situacijama kada bi se sa socijalnim distanciranjem krenulo u kasnijim vremenskim trenucima [11]. Zarad uštede prostora, a i kako su krive svih odeljaka u suštini povezane, predstavljeni su rezultati samo za *Infected* odeljak. U ovom slučaju, pored sličnih "odstupanja" od intuitivnih pretpostavki obrazloženih u tekstu iznad, da se uočiti još par dodatnih stvari. Postoji trenutak kada primena restriktivnih mera nema efekta - ako se sa socijalnim distanciranjem krene kada je epidemija već dostigla pik, ili kasnije, to praktično neće ublažiti situaciju, ma koliko ono bilo intenzivno (čak i da se ukine 80% kontakata ne dolazi do smanjenja broja obolelih). Takođe, što se kasnije krene to je uticaj mera manji, odnosno maksimum krive se ne pomera toliko ka kasnijim danima, i teže se smanjuje. Na ovom mestu se takođe uočava i razlika između samih pregradnih modela - u slućaju SEIR modela, efekat restriktivnih mera se javlja postepenije i postoji veći vremenski interval kada je moguće efikasno reagovati na širenje epidemije, uostalom i sam pik SEIR modela bez uvođenja ikakvih mera se javlja kasnije u odnosu na SIR za iste parametre β i μ kao što smo videli i ranije - baš zbog prisustva dodatnog *Exposed* odeljka koji omogućava sistemu da postepeno prolazi kroz promene.

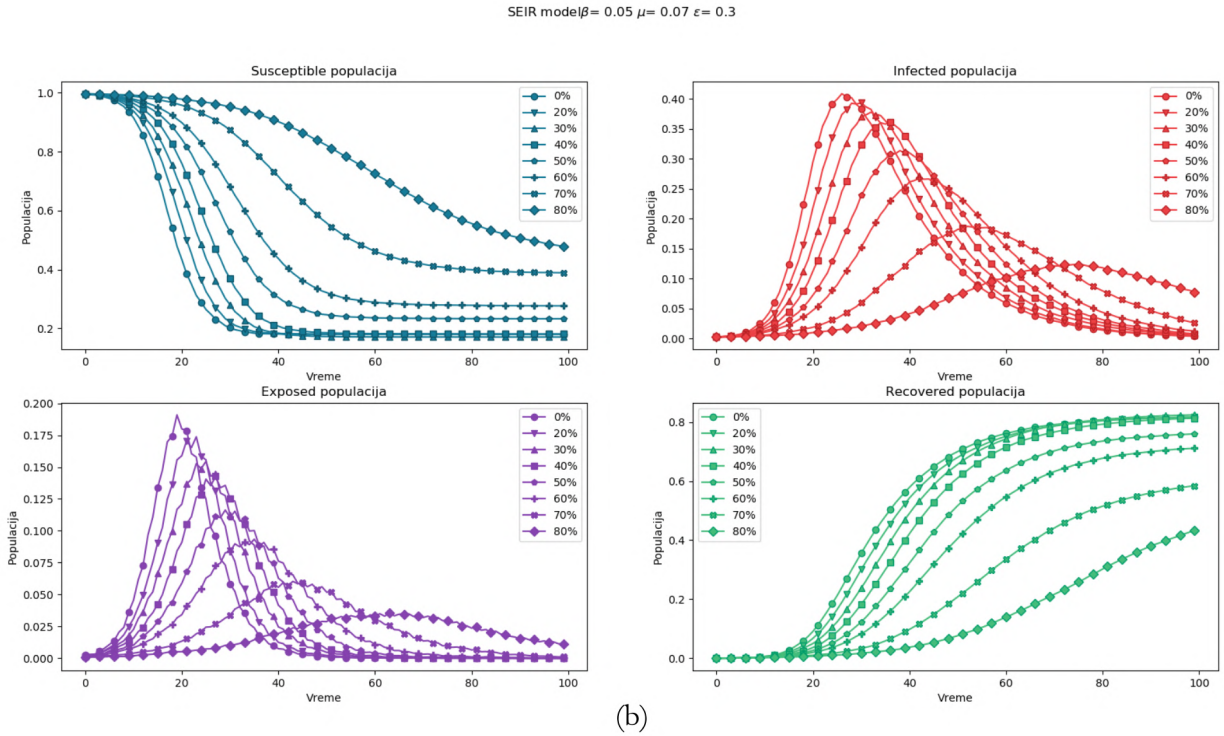


Slika 4.1. Uticaj socijalnog distanciranja (u zavisnosti od njegovog procenta) koje počinje od $t = 0$ za SIR model (a) generisanje nove "redukovane" mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije

[11] dodatak **B** "Socijlano distanciranje (od t_m-tog dana SIR/SEIR)"

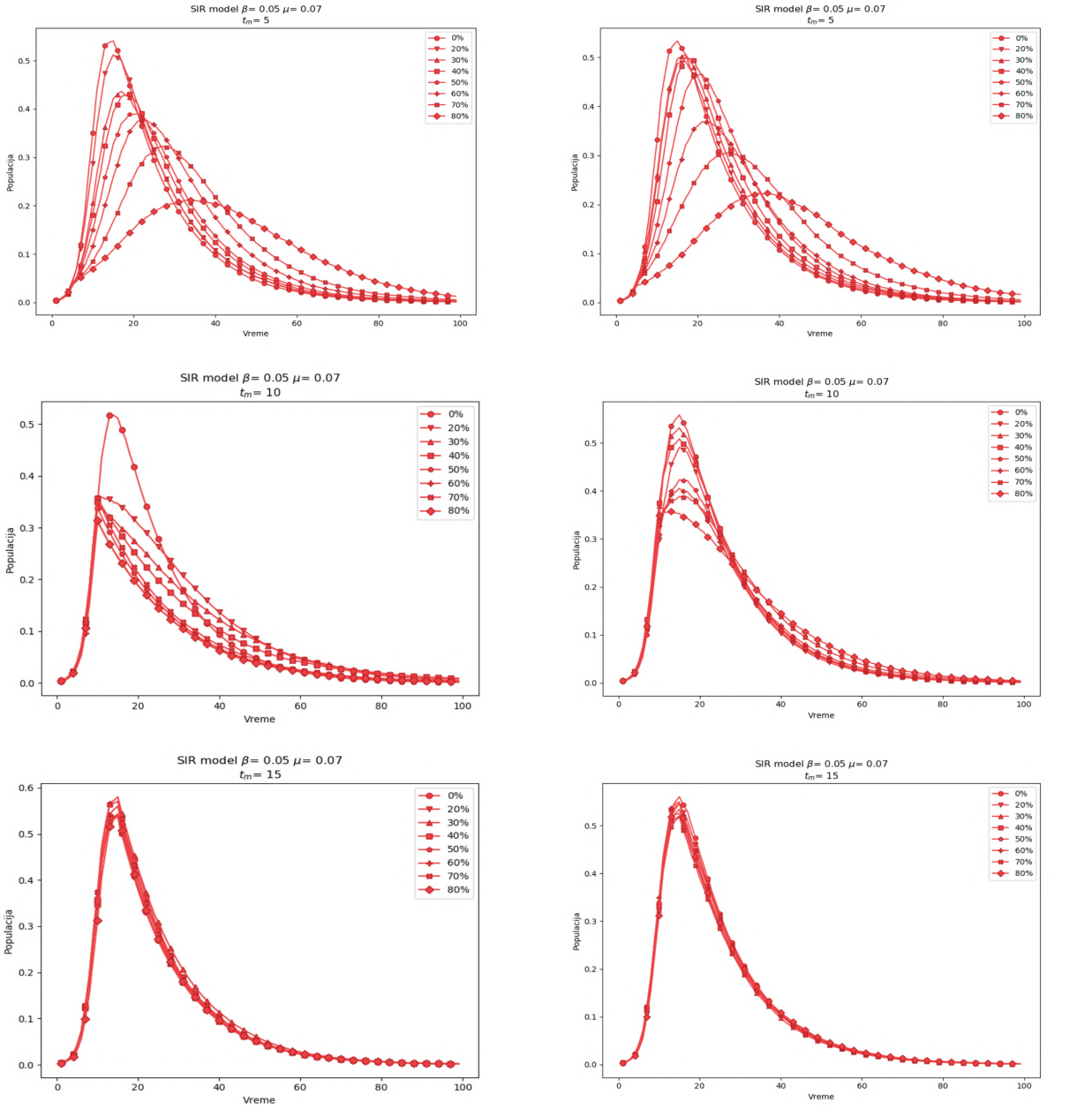


(a)

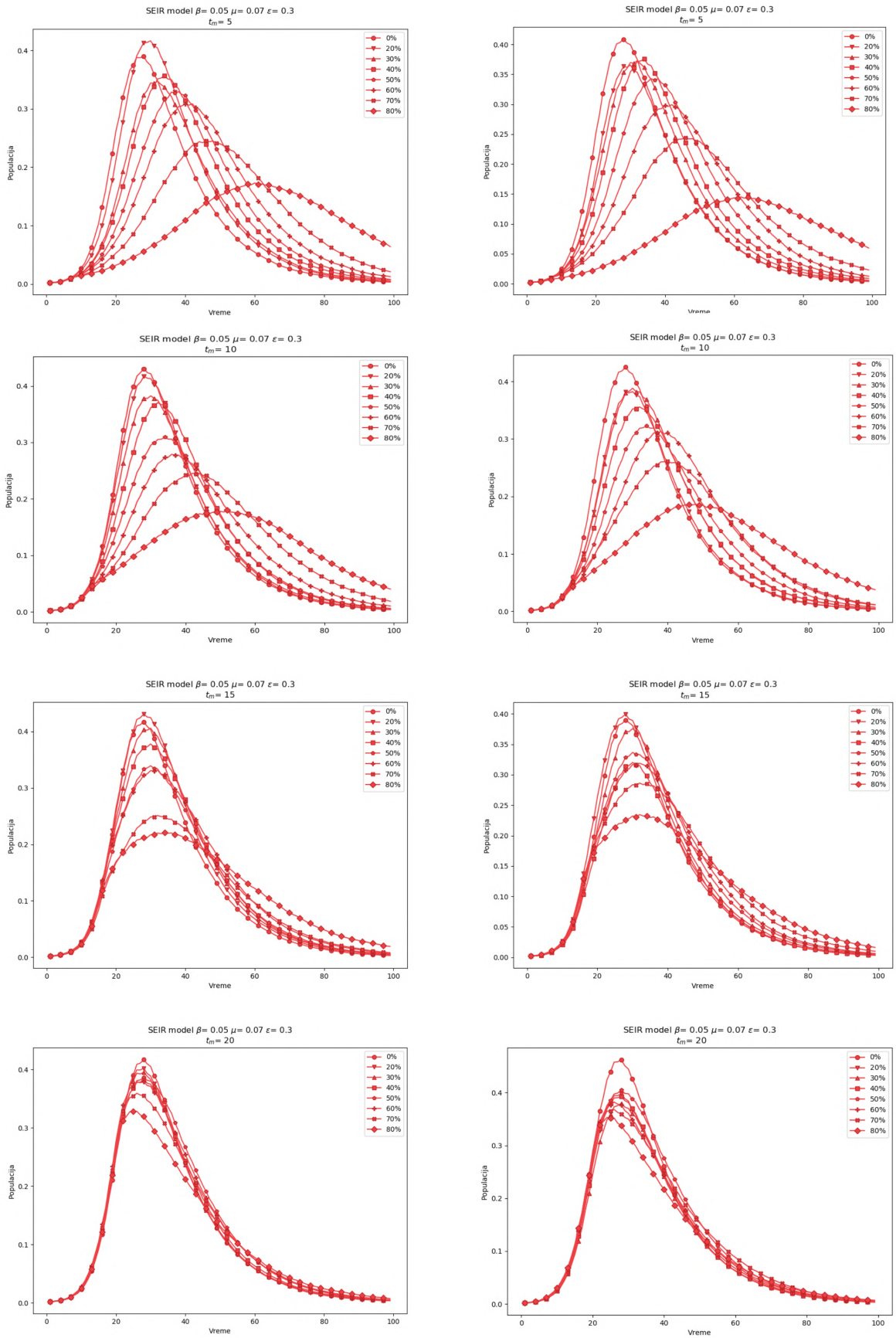


(b)

Slika 4.2. Uticaj socijalnog distanciranja (u zavisnosti od njegovog procenta) koje počinje od $t = 0$ za SEIR model (a) generisanje nove "redukovane" mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije



Slika 4.3. Uticaj socijalnog distanciranja (u zavisnosti od njegovog procenta) koje počinje od $t = 5, 10, 15$ dana za SIR model (a) generisanje nove "redukovane" mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije



(a) (b)

Slika 4.4. Uticaj socijalnog distanciranja (u zavisnosti od njegovog procenta) koje počinje od $t = 5, 10, 15, 20$ dana za SEIR model (a) generisanje nove "redukovane" mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije

4.2 Nošenje maski

Kod infekcija koje se šire vazdušnim ili kapljičnim putem nošenje zaštitnih maski predstavlja ključ u borbi protiv širenja bolesti [28]. Analogno prethodnom slučaju i u ovom odeljku želimo dati kvalitativnu sliku o tome kako bi nošenje zaštitnih maski uticalo na širenje epidemije na mreži sa konferencije u zavisnosti od procenta ljudi koji ih nosi, i od trenutka kada se sa nošenjem krene.

Ideja je sledeća: u zavisnosti od procenta ljudi koji kreću da nosi maske, dodeliti nasumičnno izabranim čvorovima varijablu koja će označavati da li ih nose ili ne[12], i sa tako modifikovanom mrežom izvršiti simulacije. U svakom trenutku pored stanja čvora proverimo da li on nosi masku ili ne, ali i da li njegovi susedi to isto rade - verovatnoća transmisije infekcije nije ista ukoliko samo jedan od njih nosi masku ili ukoliko je oboje nose, i to je predstavljeno u tabeli 1. Treba napomenuti da ne postoje egzaktne procene procentualnog udela u smanjenju verovatnoće transmisije u slučaju nošenja zaštitnih maski, osim kvalitativnih koje oslikavaju njihov dokazani značaj. S tim u vezi treba biti svestan da takve vrednosti iz tabele 1. služe najviše za sticanje nekog opšteg utiska o važnosti pridržavanja mera ovakve prirode. I ovog puta predstavljamo rezultate kao usrednjene po svim simulacijama, i to za ista dva načina implementacije kao i za slučaj socijalnog distanciranja. Rezultati za SIR i SEIR model [13] prikazani su na slikama 4.5-4.8.

Zaražena	Nezaražena	Verovatnoća infekcije se smanjuje za:
Nosi masku	Ne nosi masku	95%
Nosi masku	Nosi masku	98.5%
Ne nosi masku	Nosi masku	70%

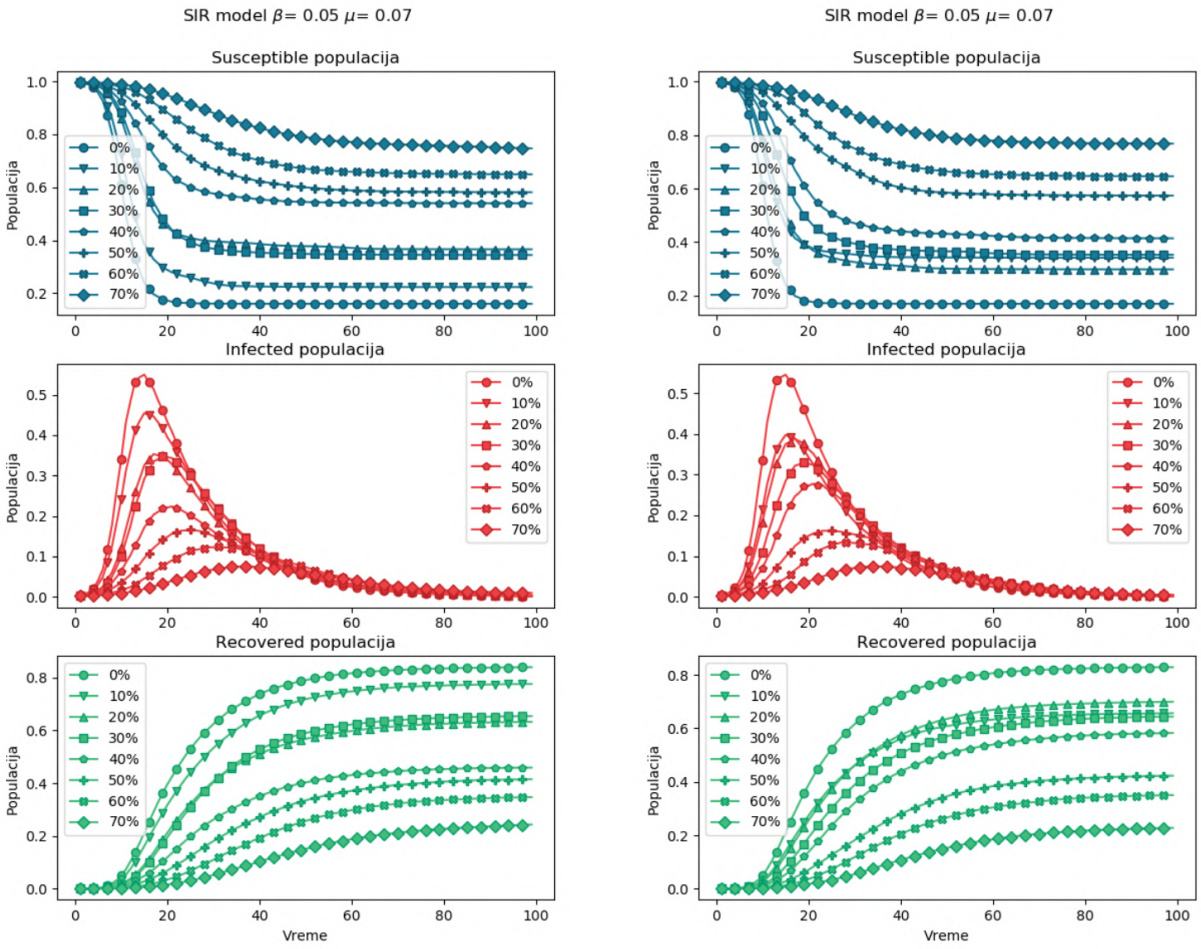
Tabela 1. Procenti smanjenja verovatnoće infekcije u zavisnosti od stanja osobe (čvora) i toga da li on i njegovi susedi nose maske

Na osnovu priloženih grafika, može se zaključiti da nošenje maski ima isti uticaj na ponašanje epidemiološke krive kao i socijalno distanciranje - maksimum se pomera ka kasnijim danima i smanjuje se, epidemija se usporava. U slučaju primene mera od prvog dana na SEIR modelu (slika 4.6(a)) može se uočiti da se za slučajeve kada 10% i 20%, odnosno 30% i 40%, odnosno 60% i 70% populacije nosi maske krive skoro poklapaju - kao da ta razlika od 10% u ovim parovima procenata ne utiče značajno na dinamiku epidemije, ali i da se pri prelasku s jednog na drugi par (sa 20% na 30%, i sa 30% na 40%) javlja značajna razlika u broju zaraženih (pikovi se razlikuju za ≈ 0.1 udeo populacije u *Infected* odeljku). Ovo je još jedan primer uticaja heterogenosti mreže na evoluciju epidemije. Rezultati su na prvi pogled skoro identični onima iz prethodnog odeljka, tj. čini se da u ovom slučaju nošenje maski ima istu "težinu" kao i socijalno distaciranje, ako ne i veću. Vidi se , na primer, da u svakoj varijanti sa priloženih grafika u slučaju da 60% i 70 % ljudi nosi masku zakrivljenje i pomeranje krive je intenzivnije čak i od situacije kada bi se 80% ljudi distanciralo. Jedan od mogućih razloga za to leži u činjenici da su ovi procenti smanjenja transmisije toliki da efektivno kao da uklanjaju kontakte. Iz tabele 1. se vidi da su oni preko 90% u čak dve situacije.

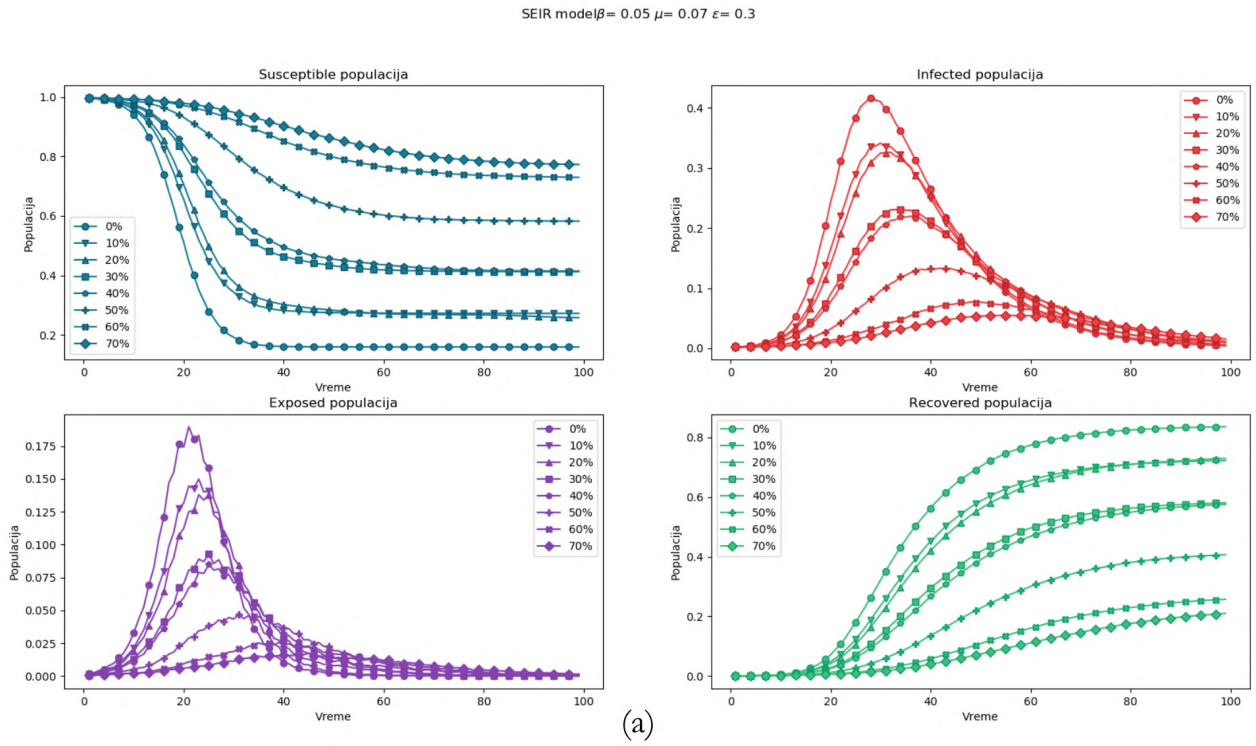
I u slučaju nošenja maski, i u slučaju socijalnog distanciranja ono što je sigurno je da se pridržavanjem ovih mera, ako se sa njima krene na vreme, utiče na epidemiju tako što se njen "intenzitet" smanjuje, što će dovesti do rasterećenja zdravstvenog sistema, i time omogućiti prijem većeg broja pacijenata na lečenje i normalno funkcionisanje bolnica i odgovarajućih ustanova.

[12] konkretno,u kodu se u dictionary čvorova doda još jedna vrednost *bool*-ovog tipa (True,False) koja ima tu funkciju (npr. ako 30% populacije nosi, nasumično izaberemo 30% čvorova i setujemo varijablu kao "True")

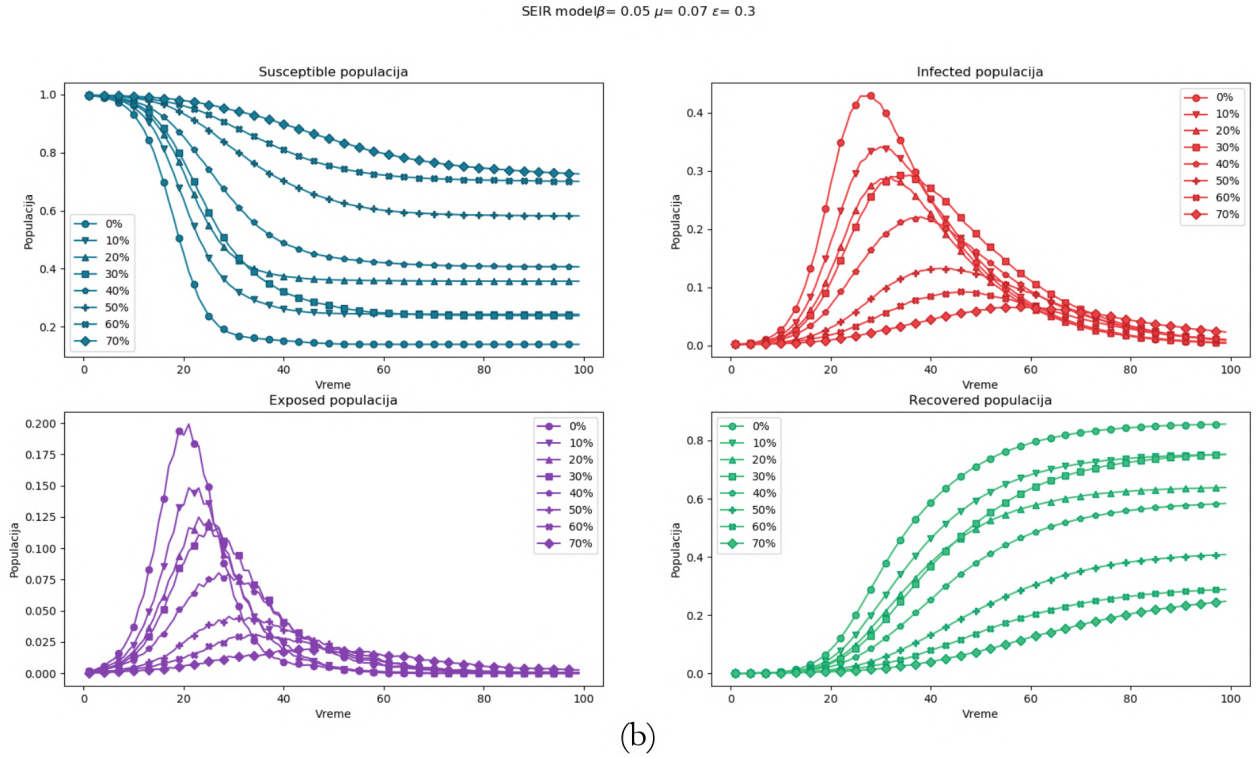
[13] kodovi se nalaze u dodatku **B** pod nazivima "Nošenje maski od prvog dana SIR/SEIR", "Nošenje maski od t_m-tog dana SIR/SEIR"



Slika 4.5. Uticaj nošenja zaštitnih maski (u zavisnosti od procenta populacije koji ih nosi) koje počinje od $t = 0$ dana za SIR model (a) generisanje nove mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije

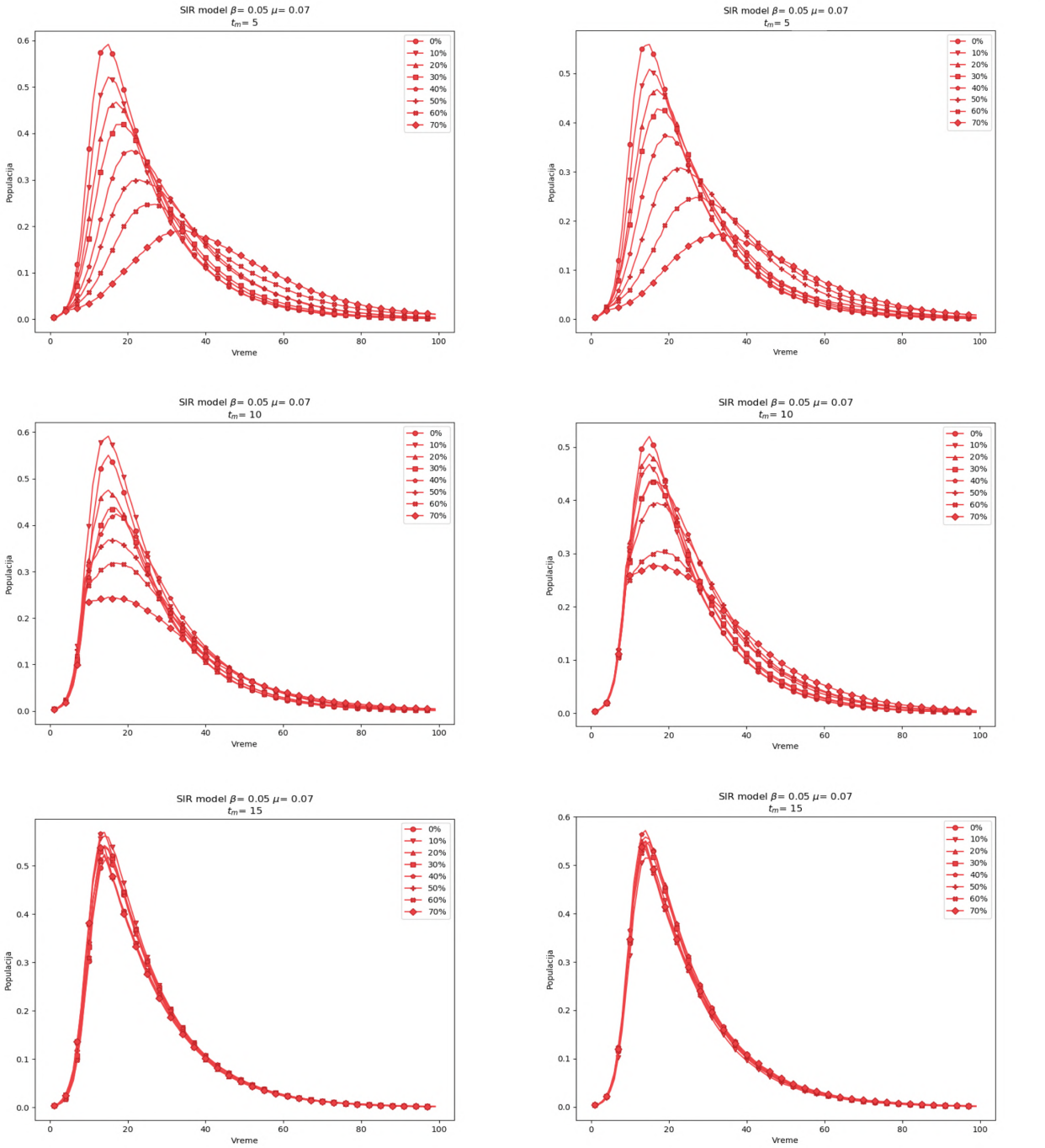


(a)

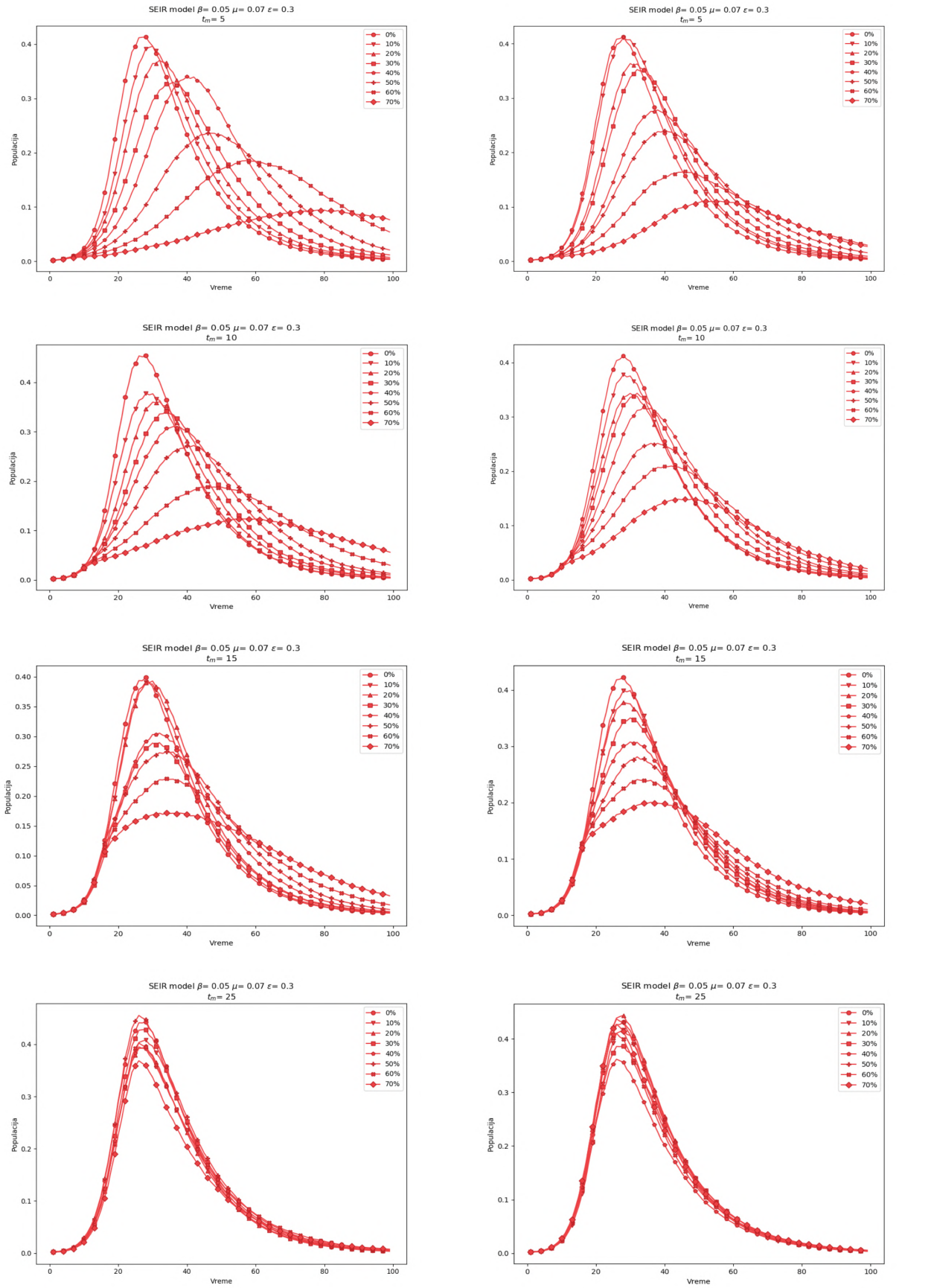


(b)

Slika 4.6. Uticaj nošenja zaštitnih maski (u zavisnosti od procenta populacije koji ih nosi) koje počinje od $t = 0$ dana za SEIR model (a) generisanje nove mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije



Slika 4.7. Uticaj nošenja zaštitnih maski (u zavisnosti od procenta populacije koji ih nosi) koje počinje od $t = 5, 10, 15$ dana za SIR model (a) generisanje nove mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije



(a) (b)

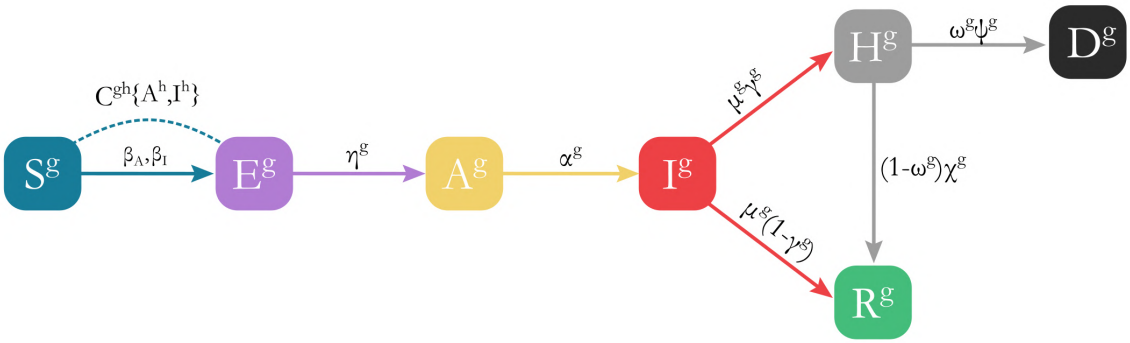
Slika 4.8. Uticaj nošenja zaštitnih maski (u zavisnosti od procenta populacije koji ih nosi) koje počinje od $t = 5, 10, 15$ dana za SEIR model (a) generisanje nove mreže u svakoj simulaciji (b) generisanje nove mreže pre svake simulacije

5 Matematički model širenja epidemije COVID19 u Srbiji

Prvi slučaj zaraze novim koronavirusom SARS-CoV-2 registrovan je decembra 2019. u Kini. Virus se od tada za par meseci proširio na ceo svet i do ovog trenutka registrovano je skoro 30 miliona slučajeva, od kojih je skoro 1 milion preminulih. Ovaj deo rada zasniva se na radu [29] u kome je predložen usložnjen pregradni model za prostorno-vremensku evoluciju epidemije korona virusa u Španiji. On uključuje karakteristične osobine virusa ali i glavne epidemiološke, kliničke i bihevioralne razlike populacije: uključuje 7 epidemioloških odeljaka, jedan od kojih je *Asimptomatski* koji reflektuje činjenicu da veliki broj registrovanih slučajeva nema, ili ima jako blage simptome [30]; deli demografiju na 3 starosne grupe - mlade, odrasle i stare kako bi se razmatrale njihove različite epidemiološke i kliničke osobine ali kako bi se video i uticaj demografskog ponašanja i interakcija svake grupe. Mi ćemo pomoću ovog modela i njegovih parametara,ali i podataka iz popisa republike Srbije iz 2011.godine predstaviti rezultate za evoluciju epidemije COVID19 u Srbiji.
Treba još dodati, da predstavljamo samo rezulate modela koji se za slučaj Srbije ne mogu uporediti sa zvaničnim prijavljenim podacima zbog njihove nepouzdanosti.

5.1 Definicija modela

Kako bi razumeli geografsku difuziju bolesti (koja je posledica međuljudske interakcije na manjim geografskim površinama) na ovako velikim sistemima moramo ukombinovati proces zaražavanja sa "dugodometnom" propagacijom bolesti prouzrokovanom ljudskom mobilnošću na različitim prostornim skalama. U tu svrhu, pretpostavka je sledeća: populacija je distribuirana po opštinama države, u kojima je populacija dobro izmešana, tj. infekcija se sa zaražene osobe može preneti na bilo koju zdravu osobu u opštini sa istom verovatnoćom. Međutim, osobe se mogu kretati između opština i tako proširiti zarazu na veće geografske skale sistema. Ovaj argument je u vezi sa *mobilnošću* osoba o kojoj će biti reči kasnije. Uglavnom, mobilnost se predstavlja pomoću mreže u kojoj su čvorovi opštine (sa njihovim lokacijama) gde veza između 2 opštine predstavlja verovatnoću kretanja između njih.
Pored ovoga, moramo definisati i dodatnu demografsku podelu stanovništva, i u ovom slučaju to je podela po starosnim dobima na mlade (stanovništvo do 25 godina), odrasle (od 26 do 65 godina) i stare (preko 65). Ovo je važno iz više razloga. Prvo, priroda same bolesti je takva da je mali broj zaraženih iz mlade populacije, dok oni u staroj imaju tešku kliničku sliku što će uticati na bolničke kapacitete; drugo, mobilnost svake grupe je drugačija (videćemo da se smatra da samo odrasle osobe putuju između opština) ali i količine interakcija između njih su različite.



Slika 5.1. Šematski prikaz pregradnog modela kojeg razmatramo. g i h označavaju starosnu grupu

Na kraju, moramo proširiti standardan SEIR model dodatnim odeljcima - u ovom slučaju imamo 7 epidemioloških odeljaka i to (Slika 5.1) :

- **S** : *Susceptible* - podložni
- **E** : *Exposed* - izloženi
- **A** : *Asymptomatic infectious* - asimptomatski zaraženi (osobe bez simptoma ali koje i dalje mogu preneti patogen)
- **I** : *Infected* - zaraženi
- **H**: *Hospitalised* - hospitalizovani, osobe premeštene u bolničke jedinice
- **D**: *Death* - preminuli
- **R** : *Recovered* - oporavljeni

Definišemo evoluciju udela agenata u stanju $m \in \{S, E, A, I, H, D, R\}$ svake starosne grupe $g \in \{1, N_G\}, N_G = 3$ u određenoj opštini $i \in \{1, ..N\}$ i označavamo je $\rho_i^{m,g}(t)$. Vremenska evolucija ovih veličina data je jednačinama:

$$\rho_i^{S,g}(t+1) = \rho_i^{S,g}(t)(1 - \Pi_i^g(t)) \tag{5.1}$$

$$\rho_i^{E,g}(t+1) = \rho_i^{S,g}(t)\Pi_i^g(t) + (1-\eta^g)\rho_i^{E,g}(t) \tag{5.2}$$

$$\rho_i^{A,g}(t+1) = \eta^g\rho_i^{E,g}(t) + (1-\alpha^g)\rho_i^{A,g}(t) \tag{5.3}$$

$$\rho_i^{I,g}(t+1) = \alpha^g\rho_i^{A,g}(t) + (1-\mu^g)\rho_i^{I,g}(t) \tag{5.4}$$

$$\rho_i^{H,g}(t+1) = \mu^g\gamma^g\rho_i^{I,g}(t) + \omega^g(1-\psi^g)\rho_i^{H,g}(t) + (1-\omega^g)(1-\chi^g)\rho_i^{H,g}(t) \tag{5.5}$$

$$\rho_i^{D,g}(t+1) = \omega^g\psi^g\rho_i^{H,g}(t) + \rho_i^{D,g}(t) \tag{5.6}$$

$$\rho_i^{R,g}(t+1) = \mu^g(1-\gamma^g)\rho_i^{I,g}(t) + (1-\omega^g)\chi^g\rho_i^{H,g}(t) + \rho_i^{R,g}(t) \tag{5.7}$$

Ove jednačine zasnovane su na pristupu mikroskopskoga lanca Markova (MMCA) modelizaciji epidemijskih procesa [31] i odgovaraju diskretnim vremenskim koracima, i kao i do sada, jedan vremenski korak predstavlja jedan dan. Interpretacija modela je sledeća. Podložne osobe postaju zaražene u kontaktu sa asimptomatskim ili zaraženim agentima verovatnoćom Π_i^g i postaju izložene (*Exposed*). Izložene osobe prelaze u asimptomatske stopom η^g a potom zaražene stopom α^g. Kada su zaražene, postoje dve mogućnosti, koje se ostvaruju stopom "izlaska" μ^g: prva mogućnost je da budu hospitalizovane, i za to je verovatnoća γ^g , u suprotnom postaju zdrave (oporave se). Kada su u bolnici u jedinicama intenzivne nege,odnosno u *Hospitalised* odeljku, osobe imaju verovatnoću smrtnosti ω^g koja se dostiže stopom ψ^g, ali se mogu i oporaviti i to stopom otpuštanja χ^g.

Veličina $\Pi_i^g(t)$ predstavlja verovatnoću da će se podložna (*Susceptible*) osoba koja pripada grupi g i opštini i zaraziti. To je složena veličina koja se izračunava na sledeći način:

$$\Pi_i^g(t) = (1-p^g)P_i^g(t) + p^g\sum_{j=1}^{N} R_{ij}^g P_j^g(t) \tag{5.8}$$

gde p^g označava stepen mobilnosti pojedinaca iz grupe g , $P_i^g(t)$ verovatnoću da se te te osobe zaraze u opštini i, a R_{ij}^g je matrica mobilnosti, tako da sada prvi član gornje jednačine predstavlja verovatnoću da se osoba inficira u kontaktu sa osobama iz svoje opštine, a drugi član zbog kontakata sa osobama iz drugih opština.
Pretpostavimo da se broj kontakata povećava sa gustinom svake opštine kao monotona funkcija f :

$$f(x) = 1 + (1 - e^{-\xi x}) \tag{5.9}$$

gde je ξ faktor gustine.
Uvedimo i matricu kontakata C čiji elementi C^{gh} predstavljaju deo kontakata koji pojedinci iz starosne grupe g ostvare sa pojedincima starosne grupe h [39].
Sada možemo napisati $P_i^g(t)$ kao :

$$P_i^g(t) = 1 - \prod_{h=1}^{N_G}\prod_{j=1}^{N}(1-\beta_A)^{z^g\langle k^g\rangle f(\frac{n_i^{eff}}{s_i})C^{gh}\frac{n_{j\to i}^{A,h}(t)}{(n_i^h)^{eff}}}(1-\beta_I)^{z^g\langle k^g\rangle f(\frac{n_i^{eff}}{s_i})C^{gh}\frac{n_{j\to i}^{I,h}(t)}{(n_i^g)^{eff}}} \tag{5.10}$$

Eksponenti izraza 5.10 predstavljaju broj kontakata koji agent iz starosne grupe g i opštine i ostvari sa zaraznim osobama iz odeljaka **A** i **I** grupe h i opštine j, stoga dupli proizvod predstavlja verovatnoću da se osoba iz starosne grupe g **ne** zarazi dok je opštini i.
Član $z^g\langle k^g\rangle f(\frac{n_i^{eff}}{s_i})$ predstavlja ukupan broj kontakata (zaraznih i nezaraznih) koji raste sa gustinom opštine i prateći funkciju 5.9, uključujući i normalizacioni faktor z^g:

$$z^g = \frac{N^g}{\sum_{i=1}^{N} f(\frac{n_i^{eff}}{s_i})(n_i^g)^{eff}} \tag{5.11}$$

gde je s_i površina opštine, a n_i^{eff} efektivna populacija u opštini i:

$$n_i^{eff} = \sum_{g=1}^{N_G}(n_i^g)^{eff} \tag{5.12}$$

i ona je podeljena u starosne grupe veličine:

$$(n_i^g)^{eff} = \sum_j[(1-p^g)\delta_{ij} + p^g R_{ij}^g]n_j \tag{5.13}$$

Poslednji član u eksponentu sadrži verovatnoću da su kontakti zarazni što je proporcionalno veličini $n_{j\to i}^{m,h}$: očekivanom broju osoba starosne grupe h zaraznog stanja $m \in \{A, I\}$ koji su prešli iz opštine j u opštinu i:

$$n_{j\to i}^{m,h}(t) = n_j^h \rho_j^{m,h}(t)[(1-p^h)\delta_{ij} + p^h R_{ji}^h] \tag{5.14}$$

Svi parametri modela predstavljeni su u tabeli 2. Svi parametri koji se tiču virusa preuzeti su iz rada [29] koji su izvučeni iz zvaničnih podataka o epidemiološkoj situaciji u Španiji, a svi demografski parametri izvučeni su iz popisa stanovništva 2011.godine (https://popis2011.stat.rs/). Popis sadrži relevantne informacije kao što su broj ljudi u opštini, njihova starost, ali i da li putuju zbog posla u neku drugu opštinu. Matrica kontakata za Srbiju je direktno uzeta iz rada [39], a matrica mobilnosti procenjena je na osnovu podataka popisa o tome koliko ljudi putuje zbog posla u druge opštine metodom opisanom u narednom odeljku, za eksponent mobilnosti

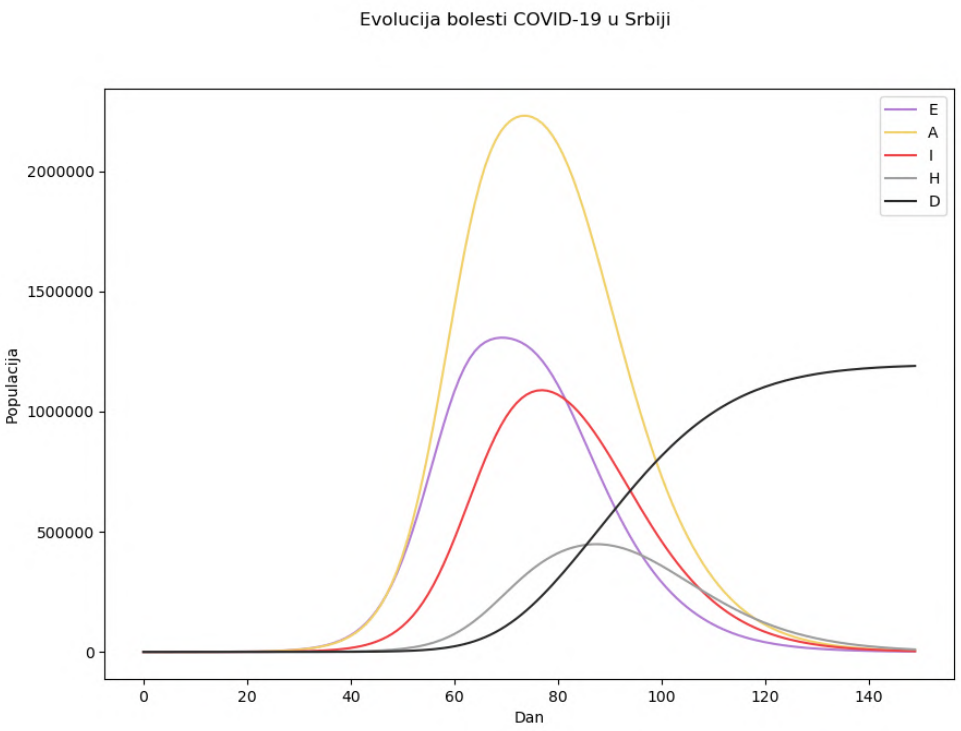
$\lambda = 2$. Za početne uslove uzeti su prijavljeni slučajevi iz zvaničnih podataka 18. marta 2020 godine i stavljeni su u asimptomatski odeljak.

Simbol	Opis	Vrednost
β_A	Infektivnost asimptomatskih agenata	0.06
β_I	Infektivnost zaraženih agenata	0.06
$\langle k^g \rangle$	Srednji broj kontakata	(12,14,7)
η^g	Stopa latentnosti	$\frac{1}{2.34}$
α^g	Stopa prelaska u *Infected*	$(\frac{1}{5.06}, \frac{1}{2.84}, \frac{1}{2.84})$
μ^g	Stopa izlaska	$(\frac{1}{1.0}, \frac{1}{3.2}, \frac{1}{3.2})$
γ^g	Deo slučajeva koji zahteva hospitalizaciju(intenzivnu negu)	(0.002,0.05,0.36)
ω^g	Stopa smrtnosti hospitalizovanih pacijenata u intenzivnoj nezi	0.42
ψ^g	Stopa smrtnosti	$\frac{1}{7.0}$
χ^g	Stopa otpuštanja iz jedinica intenzivne nege	$\frac{1}{10.0}$
n_i^g	Populacija u opštini	Iz popisa
R_{ij}^g	Matrica mobilnosti	Iz popisa
C^{gh}	Matrica kontakata po starosnim grupama	$\begin{pmatrix} 0.631 & 0.351 & 0.017 \\ 0.276 & 0.689 & 0.034 \\ 0.261 & 0.526 & 0.213 \end{pmatrix}$
ξ	Faktor gustine	0.01
p^g	Faktor mobilnosti	(0.0,1.0,0.0)

Tabela 2. Parametri modela sa kratkim opisima i vrednostima

Rezultati modela[14] predstavljeni su na slici 5.2. Pored velikog broja stanovnika koji bi bili zaraženi (i simptomatski i asimptomatski) , uočava se i veliki broj preminulih i hospitalizovanih stanovnika - oko pola miliona stanovnika bi bilo hospitalizovano, a preko milion preminulo nakon 5 meseci od početka epidemije. To su pretpostavke ovog modela za slučaj kada se ne bi uvodile nikakve restriktivne mere, pored toga, broj smrtnih slučajeva je toliko veliki zbog velikog broja starijeg stanovništva (koje je najugroženije ovim virusom) u republici Srbiji. Nije na odmet prokomentarisati i činjenicu da krive zaraženih odeljaka praktično ne ispoljavaju nikakav rast prvih mesec dana od registrovanja prvog slučaja, potom imaju blagi porast od par dana, nakon čega sledi nagli eksponencijalni rast. Ovo treba služiti kao neka vrsta upozorenja, odnosno da treba reagovati što pre od trenutka javljanja prvih slučajeva kako bi sprečili eksponencijalni skok u porastu broja zaraženih.

[14]Kod u dodatku **B** "MODEL"



Slika 5.2. Rezultati modela za $\lambda = 2$. *Susceptible* i *Infected* krive nisu prikazane zbog velikog opsega koje zauzimaju, a i predstavljeni odeljci su epidemiološki najbitniji.

5.2 Uticaj mobilnosti na evoluciju epidemije modela

Model je postavljen tako da čvorove mreže predstavljaju opštine države, pod pretpostavkom da je unutar svake opštine ispunjen uslov homogenog mešanja. Ljudi se mogu kretati iz jedne opštine u drugu, ali verovatnoća kretanja između svih parova opština neće biti jednaka - što su one udaljenije, to je manja verovatnoća da ljudi putuju između njih. To reprezentuje matrica mobilnosti R_{ij} koju ćemo proceniti pomoću popisa stanovništva, u kome postoje podaci koliko osoba putuje (zbog posla) iz opštine i u opštinu j. Ovi podaci su postoje samo za radno sposobno stanovništvo, tako da je u ovom slučaju "mobilan" samo odrastao deo stanovništva, odnosno osobe između 26 i 65 godina, što je reflektovano i u faktoru mobilnosti starosnih grupa $p^g = (0.0, 1.0, 0.0)$. Dakle, samo odrasle osobe putuju između opština, mladi i stari ostaju u svojoj izvornoj opštini. Ova mreža dnevnih migracija između opština je jedan od glavnih elemenata našeg modela, i nas zanima kako i koliko njena struktura utiče na širenje epidemije.

Mobilnost dakle predstavlja neku količinu kretanja,putovanja između lokacija, i informacija o njoj se može dobiti ili prikupljanjem podataka (npr. prikupljanjem podataka o lokaciji mobilnih telefona) ili procenama iz modela [32]. Takvi modeli zasnovani su na poznavanju veličine populacije u svakom mestu. Jedan od najpoznatijih modela je tzv. gravitacioni model koji se,između ostalog, koristi za predviđanje kretanja populacije [33, 34, 35]. U njemu intenzitet interakcija x_{ij} između dve lokacije određen je veličinom njihovih populacija N_i, N_j i rastojanjem između njih d_{ij}:

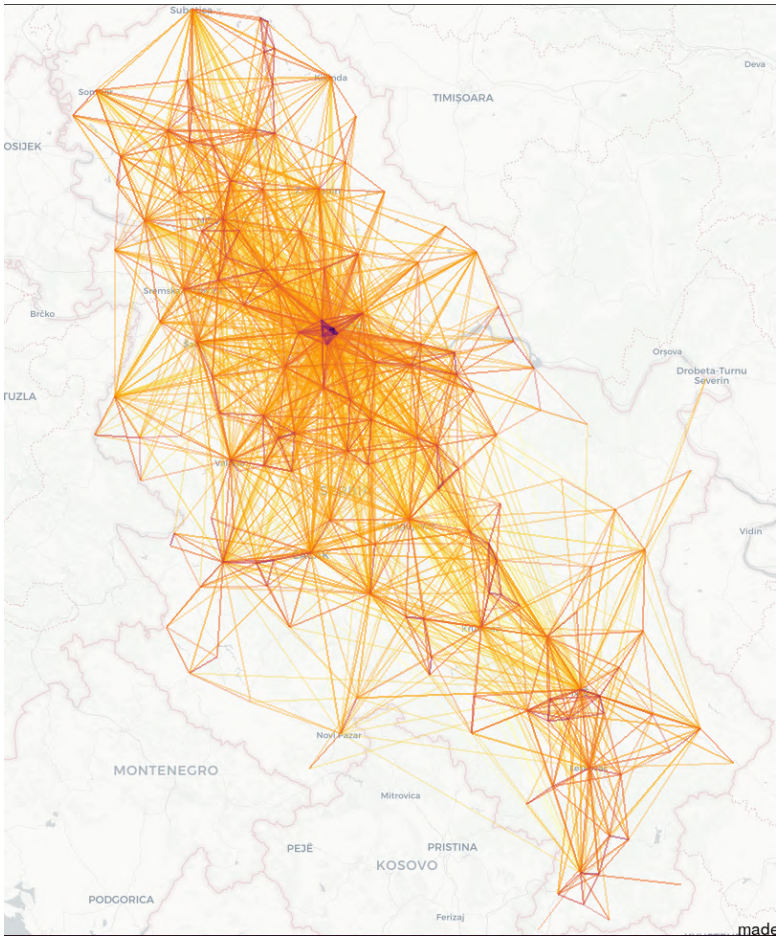
$$x_{ij} \propto \frac{N_i N_j}{d_{ij}^{\alpha}}$$

gde eksponent α određuje zavisnost od rastojanja. Mi smo, za svaku opštinu iskoristili informaciju o broju ljudi koji putuju na posao van svoje izvorne opštine. Ova informacija nam je poslužila da odredimo broj radno

sposobnih građana koji iz opštine i zbog posla odlazi u druge opštine j : M_i. Raspodelu ove veličine za Srbiju odredili smo na osnovu verovatnoće:

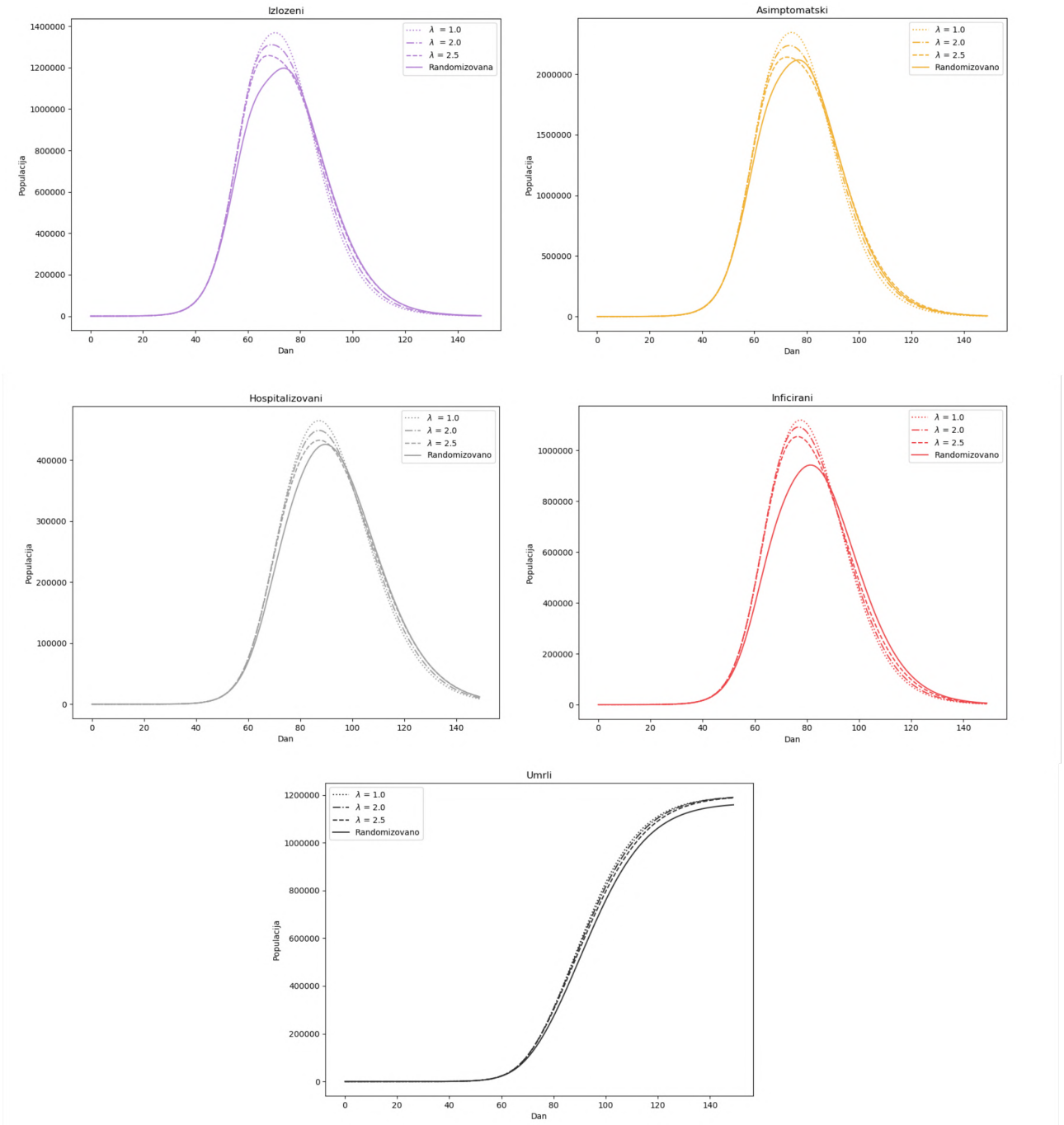
$$P_{ij} = \frac{N_j}{d_{ij}^{\lambda}}$$

gde je N_j ukupna populacija opštine j a d_{ij} udaljenost između opština i i j. Dnevna migracija iz opštine i ka opštini j jednaka je broju građana koji radi u opštini j a živi u opštini i, i ona je proporcionalna veličini opštine j a obrnuto proporcionalna njihovoj udaljenosti. Parametar λ određuje koliki je uticaj udaljenosti na mobilnost između dve opštine - što je veći to udaljenost između opština ima veći uticaj na mobilnost između njih. Dosadašnja istraživanja na podacima dobijenim od mobilnih operatera pokazala su da je $\lambda \approx 2$. Kako mi nažalost nemamo podatke za Srbiju, variraćemo vrednost ovog parametra i generisati različite mreže mobilnosti i posmatrati kako se u odnosu na vrednost ovog parametra menja dinamika epidemije. Tako smo generisali set mreža za vrednosti parametra $\lambda = 1$, $\lambda = 2$, $\lambda = 2.5$. Vizuelna reprezentacija mreže na Srbiji za $\lambda = 2$ predstavljena je na slici 5.3, a oblik epidemioloških kriva za tu vrednost predstavljen je i u prethodnom odeljku na slici 5.2. Dodatno, generisali smo i set mreža gde imamo homogeno izmešanu populaciju i između opština, tj. na nivou cele države. To praktično znači da je broj građana M_i jednako raspodeljen između svih opština, i da veličina opština i njihova udaljenost nemaju uticaj na dnevne migracije. Rezultati za sva 3 slučaja[15] predstavljeni su na slici 5.4.



Slika 5.3. Vizuelna reprezentacija mreže modela za Srbiju za $\lambda = 2$. Prikazane su samo veze težine veće od 10 (kada više od 10 osoba putuje u druge opštine) da bi se izbegla prenatrpanost.

[15] za svaki slučaj generisano je po 100 mreža, pa su predstavljeni usrednjeni rezultati, standardna devijacija se ispostavilo da je izuzetno mala i praktično se nije videla na grafiku dok se nije zumiralo oko pikova u opsegu od samo par dana, što znači da su generisane mreže jako slične. Iz tog razloga se ne vide na priloženim plotovima.

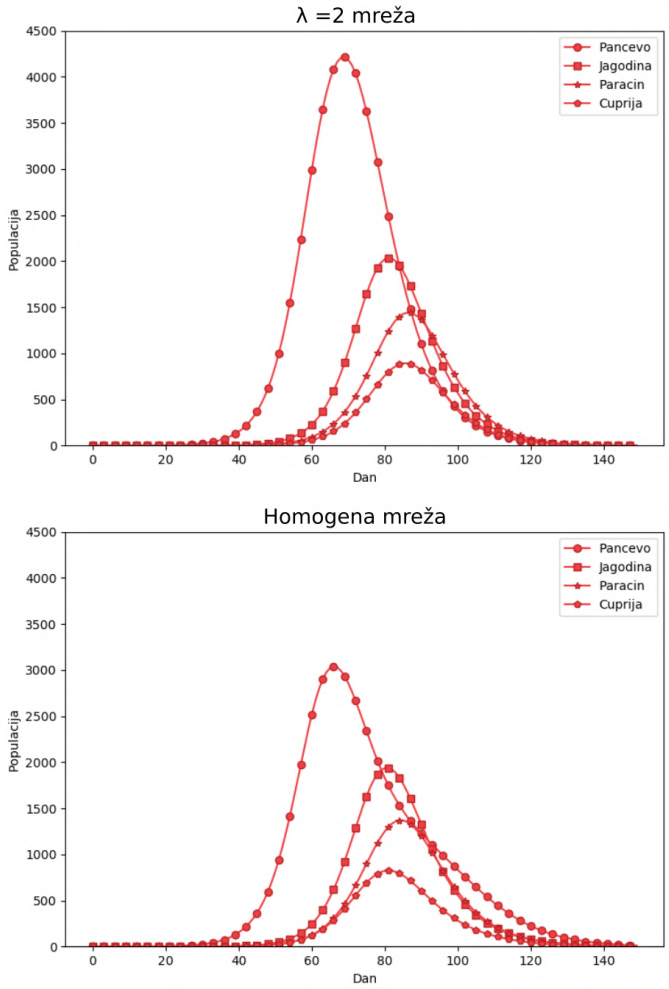


Slika 5.4. Rezultati modela za generisane mreže sa eksponentom mobilnosti $\lambda = 1$, $\lambda = 2$, $\lambda = 2.5$ i potpuno homogene mreže.

Vidi se da mobilnost modelovana gravitacionim modelom za različite eksponente λ nema prevelikog uticaja na dinamiku epidemije, pikovi su nešto niži i pomereniji ka ranijim danima za veće eksponente. Ono što je jako bitno je to, da u ovom slučaju, potpuno homogena mreža usporava i "smiruje" širenje epidemije - pik je niži i javlja se u kasnijim danima u odnosu na sve heterogene mreže različitih λ. Razlog je taj, što u ovom slučaju epidemija kreće iz (u realnosti) jako prometnih opština a mi smo randomizacijom praktično uklonili tu prometnost, odnosno uklonili smo *hub*-ove, i time praktično usporili epidemiju u startu. Dakle, ovde heterogenost mreže ubrzava širenje epidemije. Podsetimo se, da smo u prethodnim odeljcima, za heterogenu mrežu sa konferencije u Francuskoj imali suprotnu situaciju - heterogenost je usporavala širenje epidemije. Ovo

jasno ukazuje na važnost koju ima struktura mreže, ali i sami početni uslovi.

Zbog strukture mreže i početnih uslova (tj. mesta gde su se pojavili prvi zaraženi) pikovi kriva se javljaju kasnije na lokalnom nivou. Kao primer, na slici 5.5 predstavljene su *Infected* krive za 4 opštine: Pančevo, Ćupriju, Jagodinu i Paraćin, i to za homogenu mrežu i mrežu sa eksponentom $\lambda = 2$.



Slika 5.5. Evolucija epidemije za opštine: Pančevo, Ćuprija, Jagodina i Paraćin

Pančevo i Jagodina imaju jako sličan broj stanovnika - oko 70 000, ali broj inficiranih u Pančevu je duplo veći (u heterogenom slučaju) od onog u Jagodini - jedan od razloga je njegova blizina Beogradu (a samim tim i velika mobilnost između njih) koji je bio jedan od najpogođenijih gradova, pa je pik bio veći i desio se ranije. Takođe, uočava se primetno manji maksimum zaraženih u Pančevu za homogeni slučaj. Paraćin, Jagodina i Ćuprija su međusobno blizu i veliki broj ljudi iz ovih opština putuje iz jedne u drugu na dnevnoj bazi. Jagodina je najveća i nalazi se u blizini drugih većih gradova pa se pik javlja nešto ranije, i intenzivniji je. U svakom slučaju, vidi se da epidemija ne dostiže svoj maksimum u svakom gradu u isto vreme - i zbog heterogenosti sistema ali i zbog početnih uslova (čemu svedoče pomereni pikovi čak i kada je mreža potpuno homogena).

5.3 Izračunavanje R_0 modela

Za ovaj model može se analitički odrediti osnovni reproduktivni broj pomoću pristupa matrice sledeće generacije. Kao i do sada, za to je potrebno linearizovati jednačine 5.1-5.8 na stabilno stanje bez infekcija. U takvom stanju udeo susceptibilne populacije je mnogo veći od svih ostalih, i dosadašnje jednačine možemo razviti u prvi red za male vrednosti $\epsilon \sim \rho_i^m << \rho_j^S \quad \forall i,j$ i $\rho_i^m << 1 \quad \forall i \quad m \in \{E, A, I, H, D, R\}$. Ovakav razvoj transformiše naše diskretne Markov-Chain jednačine po vremenu, u kontinualne diferencijalne jednačine po vremenu. Polazimo od razvoja verovatnoća P_i^g:

$$P_i^g = \sum_{h=1}^{N_G}\sum_{j=1}^{N} z^g \langle k^g \rangle f_i C^{gh} \frac{n_j^h[(1-p^h)\delta_{ij} + p^h R j i^h]}{(n_i^h)^{eff}} \left(b^A \rho_j^{A,h} + b^I \rho_j^{I,h}\right) + O(\epsilon^2) \tag{5.15}$$

gde smo definisali:

$$b^m = ln\left[(1-\beta_m)^{-1}\right], \quad m \in \{A, I\} \tag{5.16}$$

Kada se ovo vrati u Π_i^g dobija se:

$$\Pi_i^g = \sum_{h=1}^{N_G}\sum_{j=1}^{N} \left((M_1)_{ij}^{gh} + (M_2)_{ij}^{gh} + (M_3)_{ij}^{gh} + (M_4)_{ij}^{gh}\right)\left(b^A \rho_j^{A,h} + b^I \rho_j^{I,h}\right) + O(\epsilon^2) \tag{5.17}$$

gde su $M_{1,2,3,4}$ tenzori definisani kao:

$$(M_1)_{ij}^{gh} = \delta_{ij}(1-p^g) z^g \langle k^g \rangle f_i C^{gh} \frac{(1-p^h) n_j^h}{(n_i^h)^{eff}} \tag{5.18}$$

$$(M_2)_{ij}^{gh} = (1-p^g) z^g \langle k^g \rangle f_i C^{gh} \frac{R_{ji}^h p^h n_j^h}{(n_i^h)^{eff}} \tag{5.19}$$

$$(M_3)_{ij}^{gh} = p^g R_{ij}^g z^g \langle k^g \rangle f_j C^{gh} \frac{(1-p^h) n_j^h}{(n_j^h)^{eff}} \tag{5.20}$$

$$(M_4)_{ij}^{gh} = \sum_{k=1}^{N} p^g R_{ik}^g z^g \langle k^g \rangle f_k C^{gh} \frac{R_{jk}^g p^h n_j^h}{(n_k^h)^{eff}} \tag{5.21}$$

i predstavljaju četiri različita načina na koji se epidemiološke interakcije mogu dogoditi:

- M_1-interakcije između osoba koje pripadaju istoj opštini $i = j$
- M_2-interakcija u opštini i sa osobama koje dolaze iz opštine j
- M_3- interakcija u opštini j sa osobama koje dolaze iz opštine i
- M_4- interakcije osoba koje pripadaju i i j u bilo kojoj drugoj opštini k

Kako u pogledu matrice sledeće generacije posmatramo samo odeljke koji se odnose na zaražena stanja, odgovarajuće diferencijalne jednačine sada postaju:

$$\dot{\rho}_i^{E,g} = -\eta_g \rho_i^{E,g} + \sum_{h=1}^{N_G}\sum_{j=1}^{N} M_{ij}^{gh} (b^A \rho_j^{A,h} + b^I \rho_j^{I,h}) \tag{5.22}$$

$$\dot{\rho_i^{A,g}} = \eta_g \rho_i^{E,g} - \alpha^g \rho_i^{A,g} \tag{5.23}$$

$$\dot{\rho_i^{I,g}} = \alpha_g \rho_i^{A,g} - \mu^g \rho_i^{I,g} \tag{5.24}$$

u kojima je tenzor M dat kao $M = \sum_{l=1}^{4} M_l$.

Ove jednačine možemo prepisati u matričnom obliku ako definišemo vektor $(\rho^g)^T = (\rho^{E,g}, \rho^{A,g}, \rho^{I,g})$ kao:

$$\dot{\rho^g} = \sum_{h=1} (F^{gh} - V^{gh})\rho^h \tag{5.25}$$

$$F^{gh} = \begin{pmatrix} \mathbb{O}_{N\times N} & b^A M^{gh} & b^I M^{gh} \\ \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} \\ \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} \end{pmatrix} \tag{5.26}$$

$$V^g = \begin{pmatrix} \eta^g & 0 & 0 \\ -\eta^g & \alpha^g & 0 \\ 0 & \alpha^g & \mu^g \end{pmatrix} \tag{5.27}$$

a $V^{gh} = V^g \delta^{gh} \otimes 1_{N\times N}$. Onda se osnovni reproduktivni broj može izračunati kao $R_0 = \Lambda_{max}(FV^{-1})$ gde:

$$(FV^{-1})^{gh} = \begin{pmatrix} (\frac{b^A}{\alpha^g} + \frac{b^I}{\mu^g})M^{gh} & (\frac{b^A}{\alpha^g} + \frac{b^I}{\mu^g})M^{gh} & \frac{b^I}{\mu^g}M^{gh} \\ \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} \\ \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} & \mathbb{O}_{N\times N} \end{pmatrix} \tag{5.28}$$

pa se onda možemo ograničiti samo na vektorski prostor koji pripada *Exposed* odeljku (drugi i treći red matrice iz 5.28 koji odgovaraju odeljcima **A** i **I** su praktično 0) :

$$R_0 = \Lambda_{max}(Z) \tag{5.29}$$

gde su elementi matrice $Z^{gh} = \left(\frac{b^A}{\alpha^g} + \frac{b^I}{\mu^g}\right)M^{gh}$:

Prag epidemije se dobija iz uslova $R_0 > 1$.

U tabeli 3. predstavljene su dobijene vrednosti za osnovni reproduktivni broj bolesti COVID19 u zavisnosti od eksponenta mobilnosti λ[16]. Vrednosti su izračunate pomoću koda iz dodatka **B** pod nazivom "R_0 MODEL".

	R_0	Standardna devijacija
$\lambda = 1$	4.0080	0.0005
$\lambda = 2$	4.0664	0.0005
$\lambda = 2.5$	4.0925	0.0004
Randomizovana	4.0976	0.0009

Tabela 3. Vrednosti osnovnog reproduktivnog broja za bolest COVID19 u zavisnosti od eksponenta λ

Vidimo da mobilnost ne utiče značajno ni na vrednost R_0. R_0 je najmanje za $\lambda = 1$ koje odgovara najmanjem uticaju distance.

[16]Trenutne procene R_0 za COVID19 kreću se u opsegu 3.9-8.9 [36]

6 Zaključak

U ovom radu upoznali smo se sa osnovnim pregradnim epidemiološkim modelima, videli njihovu originalnu formulaciju u vidu diferencijalnih jednačina koje predstavljaju promenu udela populacije u odeljcima po vremenu, kada važi pretpostavka dobro izmešane populacije. U tom slučaju smo videli i kako se epidemiološke krive menjaju u zavisnosti od vrednosti parametara modela, ali i kako uvođenje dodatnih odeljaka utiče na evoluciju epidemije. Uveli smo potom i pojam osnovnog reproduktivnog broja R_0 koji daje prag za javljanje epidemije (i daje sliku o njenom intenzitetu), i pokazali kako se nalazi njegov analitički izraz u slučaju komplikovanijih pregradnih modela. Kasnije smo videli da različiti metodi procene ovog broja mogu dati jako različite rezultate, i kako se mora biti pažljiv prilikom njegovog definisanja i tumačenja. Ovo je potvrđeno i rezultatima SIR i SEIR modela u odeljku 3.1.3.

Nakon uvoda, upoznali smo se sa osnovnim elementima i pojmovima teorije mreža, kako bi uveli metod za simuliranje epidemije na realnim kompleksnim mrežama pomoću SIR i SEIR pregradnih modela. Pokazali smo da struktura mreže značajno utiče na dinamiku epidemije, i da se za realne situacije ne može uvek koristiti pretpostavka homogeno izmešane populacije. U slučaju simulacija na mreži sa konferencije u Francuskoj videli smo da njena struktura usporava širenje epidemije, za razliku od homogene meže iste veličine. Primetili smo i velike standardne devijacije u rezultatima što nam je dodatno ukazalo na uticaj koji ima heterogena struktura mreže - u zavisnosti od početnih uslova, odnosno inicijalno zaraženog čvora epidemija se može nekad proširiti skoro momentalno kroz celu mrežu (npr. ako je zaražen visoko konektovan čvor-*hub*), ali se može i ugasiti u samom početku. Dali smo i kvalitativne rezultate koje bi dale restriktivne mere u vidu socijalnog distanciranja i nošenja maski (na mreži sa konferencije), i to u zavisnosti od trenutka njihovog uvođenja. Tu smo još jednom posvedočiili uticaju heterogenosti strukture na dinamiku procesa, ali generalni zaključak je taj da su mere efikasnije što se ranije krene sa njihovim uvođenjem, i da praktično, ako se krene previše kasno one ne vrše nikakvu funkciju.

Na kraju, predstavili smo usložnjeni epidemiološki pregradni model i pomoću podataka iz popisa stanovništva, ranijih radova, i trenutnih procena parametara, predstavili smo evoluciju bolesti COVID19 na teritoriji Srbije. Takođe smo našli analitički izraz R_0 ovog modela za bolest COVID19, i izračunali njegovu vrednost, koja je bila u skladu sa aktuelnim procenama. Videli smo i kako mobilnost, odnosno "šabloni" kretanja i putovanja građana utiču na dinamiku epidemije. Ovog puta, kada je mreža bila potpuno homogena infekcjia se širila sporije nego kada je usled različite mobilnosti između gradova mreža bila heterogena.

Epidemiološko modelovanje na kompleksnim mrežama u poslednjih par godina, dovelo je do velikog napretka u razumevanju uzajamnog dejstva između osobina mreža i zaraznih procesa, ali je i otvorilo vrata za nova pitanja i probleme, što je često praćeno sve većim prilivom novih podataka. Odgovori na pitanja kao što su koji su limiti u predviđanju epidemija na mrežama, kako naše razumevanje zavisi od načina prikupljanja podataka, kakav je uticaj početnih uslova na dinamiku problema, itd. zahtevaju uključenje i rad velike interdisciplinarne istraživačke zajednice. Konačni cilj je ne samo razumevanje epidemijskih procesa, već predviđanje i kontrola njihovog ponašanja, što je od krucijalnog značaja za sprečavanje nastajanja mogućih kriza - zdravstvenih, društvenih, i ekonomskih.

A

Određivanje R_0 metodom matrice sledeće generacije

Pregradni modeli dele populaciju na konačan broj diskretnih kategorija koje uglavnom opisuju zdravstveno stanje pojedinaca u njima. U tom slučaju možemo definisati matricu koja povezuje brojeve novozaraženih osoba u raznim kategorijama u uzastopne generacije. Ova matrica naziva se *matricom sledeće generacije* (MSG)[17], a vrednost R_0 se dobija kao njena dominantna svojstvena vrednost.[12]

Dinamiku sistema opisuju nelinearne diferencijalne jednačine koje opisuju promenu udela populacije u svakom odeljku u jedinici vremena. Za računanje R_0 potrebni su nam samo odeljci koji se odnose na zaražena stanja, odnosno posmatraćemo samo "zaraženi podsistem" jednačina. Prvi korak je linearizacija zaraženog podsistema na stabilno stanje bez zaraze, koje po pravilu postoji. Ova linearizacija u epidemiološkom smislu reflektuje to da R_0 karakteriše potencijal za početno širenje infektivnog agenta kada se on uvede u poptuno podložnu populaciju, i da pretpostavljamo da je promena u podložnoj populaciji zanemarljiva tokom inicijalnog širenja. Svaki linearni sistem običnih diferencijalnih jednačina može se opisati matricom (Jakobijanom). Struktura ove matrice ima jasnu epidemiološku interpretaciju : ona se može razložiti na zbir dve matrice : $T + \Sigma$,gde je T *transmisioni* deo koji opisuje nastajanje novih infekcija, a Σ *tranzicioni* deo, koji opisuje promenu stanja. Dominantna svojstvena vrednost matrice $K_L = -T\Sigma^{-1}$ predstavlja traženi osnovni reproduktivni broj R_0. Matrica K_L nije striktno matrica sledeće generacije K, ali se ona može dobiti iz K_L pogodnom linearnom transformacijom, tako da ćemo nazvati K_L *MSG velikog domena.* Spektralni radijusi ove dve matrice su isti i dominantne ajgenvrednosti obe daju R_0 s tim što je za složenije modele pogodnije koristiti K zbog njene manje dimenzije. Iskusni epidemiolozi mogu i bez korišćenja linearne algebre odmah zapisati tačan oblik matrice K zbog biološkog značenja njenih elemenata: K_{ij} je očekivani broj novih slučajeva *epidemioloških rođenja*[18] u stanju i nastalih jednom osobom koja se tek *epidemiološki rodila* u stanju j. Sada ćemo odmah postupno izračunati R_0 za SEIR model u našem slučaju,za kompleksnije modele pogledati [12].

Diferencijalne jednačine SEIR modela date su izrazima 2.11-2.14. Zaraženi podsistem čine jednačine:

$$\frac{\partial e_t}{\partial t} = \beta s_t i_t - \epsilon e_t$$

$$\frac{\partial i_t}{\partial t} = \epsilon e_t - \mu i_t$$

U stabilnom stanju bez infekcija $E = I = R = 0$ i $S = N$,odnosno $s = 1$, što će linearizovati gornje jednačine na:

$$\frac{\partial e_t}{\partial t} = \beta i_t - \epsilon e_t$$

$$\frac{\partial i_t}{\partial t} = \epsilon e_t - \mu i_t$$

Ovo vektorski možemo zapisati u obliku:

$$\dot{\vec{x}} = (T + \Sigma)\vec{x}$$

[17] eng. *next generation matrix*

[18] Epidemiološka rođenja predstavljaju stanja koja pojedinci mogu zauzeti odmah nakon njihove infekcije.

gde $\vec{x} = (e_t, i_t)$. Svi epidemiološki događaji koji vode do novih infekcija se nalaze u T, a svi ostali događaji u Σ. Dakle:

$$T = \begin{pmatrix} 0 & \beta \\ 0 & 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} -\epsilon & 0 \\ \epsilon & -\mu \end{pmatrix}$$

Nalaženjem dominantne svojstvene vrednosti matrice $K_L = -T\Sigma^{-1}$ dobićemo R_0. Iako ovo nisu velike matrice, i operacije sa njima nisu komplikovane, zgodno je objasniti kako bi se dobila MSG K iz K_L. Definišimo matricu E čije su kolone jedinični vektori koji se odnose samo na nenulte redove matrice T. U našem slučaju samo prvi red matrice transmisije T je nenulti,i E je onda zapravo samo vektor:

$$E = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Da smo,na primer,imali 3 zarazna odeljka,i matricu T dimenzije 3×3 čija su prva 2 reda nenulta, matrica E bi bila:

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Uglavnom, matrica sledeće generacije K dobija se sledećom transformacijom matrice K_L :

$$K = E^T K_L E = -E^T T \Sigma^{-1} E$$

Nalaženje dominantne svojstvene vrednosti daće za SEIR model:

$$R_0 = \beta/\mu$$

što je potpuno identično SIR slučaju.

B

Kodovi

ODEINT SI model

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

b=0.1 # koeficijent beta
def sir(x,t):

    s=x[0]
    i=x[1]
    #diferencijalne jednacine modela
    dsdt=-b*s*i
    didt=b*s*i

    return [dsdt,didt]


x0=[44,1] #pocetni uslovi St(t=0)=44 I(t=0)=1
t=np.linspace(0,40,1000)

x=odeint(sir,x0,t) #resavanje jednacina

s=x[:,0]
i=x[:,1]

plt.suptitle('SI model '+r'$\beta$='+str(b))
plt.plot(t,s/45,'#177c99',label=r'$s_t$')
plt.plot(t,i/45,'#ed4145',label=r'$i_t$')

plt.xlabel('t')
plt.ylabel('Populacija')
plt.legend(loc='best')
plt.show()
```

ODEINT SIR model

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt


beta = 0.05
mu = 0.07

def sir(x,t):

    s=x[0]
    i=x[1]
    r=x[2]

    dsdt=-beta*s*inf
    didt=beta*s*inf-mu*inf
    drdt=mu*inf

    return [dsdt,didt,drdt]

x0=[44,1,0]
t=np.linspace(0,100,1000)

x=odeint(sir,x0,tm)

plt.suptitle('SIR model '+r'$\beta$='+str(beta)+ r" $\mu$= " +str(mu))
plt.plot(t,s/45,'#177c99ff',label=r'$s_t$')
plt.plot(t,i/45,'#ed4145ff',label=r'$i_t$')
plt.plot(t,r/45,'#44bd7bff',label=r'$r_t$')
plt.xlabel('t (dani)')
plt.ylabel('Populacija')
plt.legend(loc='best')
plt.show()
```

ODEINT SEIR model

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

b = 0.01
mu = 0.02
ep = 0.2

def seir(x,tm):

    s=x[0]
    i=x[1]
    e=x[2]
    r=x[3]
    dsdt=−b∗s∗inf
    dedt=b∗s∗i−e∗ep
    didt=e∗ep−mu∗i
    drdt=mu∗i

    return [dsdt,didt,dedt,drdt]

x0=[45,1,0,0]
t=np.linspace(0,140,1000)

x=odeint(seir,x0,tm)

s=x[:,0]/45
i=x[:,1]/45
e=x[:,2]/45
r=x[:,3]/45

plt.suptitle('SIR␣model␣'+r'$\beta$='+str(b)+ r"␣$\mu$=" +str(mu)
+r'␣$\epsilon␣$='+str(ep))
plt.plot(t,s,'#177c99ff',label=r'$s_t$')
plt.plot(t,i,'#ed4145ff',label=r'$i_t$')
plt.plot(t,r,'#44bd7bff',label=r'$r_t$')
plt.plot(t,e,'#b17cd3ff',label=r'$e_t$')
plt.xlabel('t␣(dani)')
plt.ylabel('Populacija')
plt.legend(loc='best')
plt.show()
```

SIR na mreži

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import json
import networkx as nx
import random as rd
import numpy.random as nrd

def susedi(i): #daje listu suseda cvora,cisto zbog lakseg i pedantnijeg zapisa
    sus=list(g.neighbors(i))
    return (sus)


beta = 0.01
mu = 0.02

pb = [1 − beta, beta] # verovatnoca prelaska s−i
pm = [1 − mu, mu] # verovatnoca prelaska i−r

N=45
g=nx.erdos_renyi_graph(N,1) #generise graf date velicine i verovatnoce formiranja veza

node_states={}

timestep=101 #vremenski korak


li=[] #lista za infected stanja
ls=[] #lista za susceptible stanja
```

```
lr=[] #lista za recovered stanja

'''
Stanje Vrednost koja ce beleziti stanje
Susceptible 0
Infected 1
Recovered 2
'''

#definisem dictionary sa podacima o cvorovima mreze kroz koje cu da pratim susede i stanja
for i in g.nodes():
    node_states[i]={}
for i in node_states:
    node_states[i]['t']=0 #stanje cvora i u prethodnom trenutku
    node_states[i]['t+1']=0 #stanje cvora iu aktuelnom trenutku
    node_states[i]['susedi']=susedi(i) #susedi cvora i

nsim=100 #broj simulacija
for sim in range(nsim):
    St = {} # bice formata S={Vremenski trenutak : [susceptible stanja]}
    It = {}
    Rt = {}
    print(sim)

    #zarazimo nasumicnog nultog pacijenta
    infected = rd.choice(list(node_states.keys()))
    node_states[infected]['t'] = 1

    for t in range(1,timestep):
        # print ('timestep ',t)
        St[t] = []
        It[t] = []
        Rt[t] = []

        for i in node_states:

            if node_states[i]['t'] == 1: # if infected
                node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0] # verovatnoca za oporavak

                for j in node_states[i]['susedi']: # za njegove susede
                    if node_states[j]['t+1'] == 0: # if stanje suseda susceptible moze se zaraziti nekom verovatnocom
                        node_states[j]['t+1']= nrd.choice([0, 1], 1, p=pb)[0]

            elif node_states[i]['t'] == 2: # ako je recovered ne moze ponovo da se inficira
                node_states[i]['t+1'] = 2

        for i in node_states:
            node_states[i]['t'] = node_states[i]['t+1']
            if node_states[i]['t+1'] == 0:
                St[t].append(0)
            elif node_states[i]['t+1'] == 1:
                It[t].append(1)
            elif node_states[i]['t+1'] == 2:
                Rt[t].append(2)

    S = {}
    I = {}
    R = {}
    for t in St.keys():
        S[t] = len(St[t])
        I[t] = len(It[t])
        R[t] = len(Rt[t])

    # print(I,'\n',list(I.values()))

    ls.append(list(S.values()))
    li.append(list(I.values()))
    lr.append(list(R.values()))

    for k in node_states: # cisti posle svake simulacije
        node_states[k]['t'] = 0
        node_states[k]['t+1'] = 0


#plotovanje
ys = np.mean(ls, axis=0) /len(list(node_states.keys())) # racuna srednje vrednosti svake kolone
yr = np.mean(lr, axis=0)/len(list(node_states.keys()))
yi = np.mean(li, axis=0)/len(list(node_states.keys()))
```

```
xs = xr = xi = list(range(1,timestep))
#res = list(map(abs, test_list))
std_i=np.std(li,axis=0)/len(list(node_states.keys()))
std_s=np.std(ls,axis=0)/len(list(node_states.keys()))
std_r=np.std(lr,axis=0)/len(list(node_states.keys()))


fig, (ax1,ax2,ax3) = plt.subplots(3)
fig.suptitle('SIR model '+r'$\beta$='+str(beta)+r' $\mu$='+str(mu)+'\n'+'N='+str(N))
ax1.plot(xs,ys,'#177c99ff',label="Network")
ax1.fill_between(xs, ys + std_s,ys - std_s, facecolor='#177c99ff', alpha=0.2)
ax1.set_title('Susceptible populacija')
ax1.legend()

ax2.plot(xi,yi,'#ed4145ff',label="Network")
ax2.fill_between(xi, yi + std_i,yi - std_i, facecolor='#ed4145ff', alpha=0.2)
ax2.set_title('Infected populacija')
ax2.legend()

ax3.plot(xs,yr,'#44bd7bff',label="Network")
ax3.fill_between(xs, yr + std_r,yr - std_r, facecolor='#44bd7bff', alpha=0.2)
ax3.set_title('Recovered populacija')
ax3.legend()

ax1.set( ylabel='Populacija')
ax2.set( ylabel='Populacija')
ax3.set(xlabel='Vreme', ylabel='Populacija')

plt.show()
```

SEIR na mreži

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd

def susedi(i): #daje listu suseda cvora,cisto zbog lakseg i pedantnijeg zapisa
    sus=list(graph.neighbors(i))
    return (sus)

beta = 0.05
mu = 0.07
ep = 0.3

N=45
timestep = 100

graph=nx.fast_gnp_random_graph(N,1)

'''
Stanje Vrednost koja ce beleziti stanje
Susceptible 0
Infected 1
Exposed 2
Recovered 3
'''

node_states = {}
for i in graph.nodes():
    node_states[i] = {}
for i in node_states:
    node_states[i]['t'] = 0
    node_states[i]['t+1'] = 0
    node_states[i]['susedi'] = susedi(i)

pb = [1 - beta, beta] # verovatnoca prelaska e-i
pm = [1 - mu, mu] # verovatnoca prelaska i-r
pe = [1 - ep, ep] #verovatnoca prelaska s-e

ls = []
lr = []
li = []
```

```
le = []

for sim in range(100):
    print(sim)
    St = {} # bice formata S={Vremenski trenutak : [susceptible stanja]}
    Et = {}
    It = {}
    Rt = {}

    infected = rd.choice(list(node_states.keys()))
    node_states[infected]['t'] = 1
    for t in range(timestep):
        St[t] = []
        It[t] = []
        Et[t] = []
        Rt[t] = []
        for i in node_states:
            # prvo proverimo jel ima susede uopste

            if node_states[i]['t'] == 1: # if infected
                node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0] # verovatnoca za oporavak

                for j in node_states[i]['susedi']: # za njegove susede
                    if node_states[j]['t+1']== 0: # if stanje suseda susceptible moze se zaraziti nekom verovatnocom i
                        ↪ postati exposed
                        node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=pb)[0]

            elif node_states[i]['t'] == 2:
                node_states[i]['t+1'] = nrd.choice([2, 1], 1, p=pe)

            elif node_states[i]['t'] == 3: # ako je recovered ne moze ponovo da se inficira
                node_states[i]['t+1'] = 3

        for i in node_states:
            node_states[i]['t'] = node_states[i]['t+1']
            if node_states[i]['t+1'] == 0:
                St[t].append(0)
            elif node_states[i]['t+1'] == 1:
                It[t].append(1)
            elif node_states[i]['t+1'] == 2:
                Et[t].append(2)
            elif node_states[i]['t+1']== 3:
                Rt[t].append(3)

    S = {}
    I = {}
    E = {}
    R = {}
    for t in St.keys():
        S[t] = len(St[t])
        I[t] = len(It[t])
        R[t] = len(Rt[t])
        E[t] = len(Et[t])

    ls.append(list(S.values()))
    li.append(list(I.values()))
    lr.append(list(R.values()))
    le.append(list(E.values()))

    for k in node_states: # cisti posle svake simulacije
        node_states[k]['t'] = 0
        node_states[k]['t+1'] = 0

#plotovanje
ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
yi = np.mean(li, axis=0) / len(list(node_states.keys()))
ye = np.mean(le, axis=0) / len(list(node_states.keys()))

xs = xr = xi = xe =list(range(1,timestep))
print(len(list(node_states.keys())))
std_i=np.std(li,axis=0)/len(list(node_states.keys()))
std_s=np.std(ls,axis=0)/len(list(node_states.keys()))
std_e=np.std(le,axis=0)/len(list(node_states.keys()))
std_r=np.std(lr,axis=0)/len(list(node_states.keys()))

fig, axs = plt.subplots(2, 2)
fig.suptitle('SEIR␣model␣'+r'$\beta$=␣'+str(beta)+r'␣$\mu$=␣'+str(mu) +r'␣$\epsilon$=␣'+str(ep) + '\n' +'N=␣
```

```
    ↪ '+str(N))
axs[0, 0].plot(xs,ys,color='#177c99ff',label="Network")
axs[0, 0].fill_between(xs, ys + std_s,ys − std_s, facecolor='#177c99ff', alpha=0.15)
axs[0, 0].set_title('Susceptible populacija')
axs[0,0].legend()

axs[0, 1].plot(xe,ye,color='#b17cd3ff',label='Network')
axs[0, 1].fill_between(xe, ye + std_e,ye − std_e, facecolor='#b17cd3ff', alpha=0.15)
axs[0, 1].set_title('Exposed populacija')
axs[0,1].legend()

axs[1, 0].plot(xi,yi,color='#ed4145ff',label="Network")
axs[1, 0].fill_between(xi, yi + std_i,yi − std_i, facecolor='#ed4145ff', alpha=0.15)
axs[1, 0].set_title('Infected populacija')
axs[1,0].legend()

axs[1, 1].plot(xr,yr,color='#44bd7bff',label="Network")
axs[1, 1].fill_between(xr, yr + std_r,yr − std_r, facecolor='#44bd7bff', alpha=0.15)
axs[1, 1].set_title('Recovered populacija')
axs[1,1].legend()
for ax in axs.flat:
    ax.set(xlabel='Vreme', ylabel='Populacija')

plt.show()
```

R_0 na mreži SIR

```
import math
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd
from collections import Counter

def susedi(i):
    sus=list(g.neighbors(i))
    return (sus)

N=100
beta = 0.1
mu = 0.2

g=nx.erdos_renyi_graph(N,1)
#print(g.degree())
node_states={}

pb = [1 − beta, beta] # verovatnoca prelaska s−i
pm = [1 − mu, mu] # verovatnoca prelaska i−r

timestep=101

for i in g.nodes():
    node_states[i]={}
for i in node_states:
    node_states[i]['t']=0
    node_states[i]['t+1']=0
    node_states[i]['susedi']=susedi(i)

nsim=1000
R0 = []

for sim in range(nsim):

    print(sim)
    infected = rd.choice(list(node_states.keys()))
    node_states[infected]['t'] = 1
    r0=0
    stop=0
    for t in range(1,timestep):
        for i in node_states:

            if node_states[i]['t'] == 1: # if infected
                node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0] # verovatnoca za oporavak
```

```
                for j in node_states[i]['susedi']: # za njegove susede
                    if node_states[j]['t+1'] == 0: # if stanje suseda susceptible moze se zaraziti nekom verovatnocom
                        st=nrd.choice([0, 1], 1, p=pb)[0]
                        node_states[j]['t+1']= st
                        if i==infected and st == 1:
                            r0=r0+1

            elif node_states[i]['t'] == 2:
                node_states[i]['t+1'] = 2

        if node_states[infected]['t+1']==2:
            stop=1
        if stop==1:
            break
        for i in node_states:
            node_states[i]['t'] = node_states[i]['t+1']
    R0.append(r0)
    for k in node_states: # cisti posle svake simulacije
        node_states[k]['t'] = 0
        node_states[k]['t+1'] = 0


fr=Counter(R0)

for f in fr:
    fr[f]=fr[f]/nsim

print(np.mean(R0))
print (np.std(R0))

print()
lists = sorted(fr.items())

x, y = zip(*lists)

plt.suptitle('R0 SIR model'+r'$\beta$= '+str(beta)+r' $\mu$= '+str(mu))
plt.bar(x,y,color='#267bb8cf',edgecolor='#001321cf')
plt.xlabel('R0-Broj sekundarnih slucajeva')
plt.ylabel('Frekfrencija')
plt.show()
```

R_0 na mreži SEIR

```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
import networkx as nx
import random as rd
import numpy.random as nrd

def susedi(i):
    sus=list(graph.neighbors(i))
    return (sus)

N=100

beta = 0.1
mu = 0.2
ep = 0.3

timestep = 1000

graph=nx.fast_gnp_random_graph(N,1)
node_states = {}
for i in graph.nodes():
    node_states[i] = {}
for i in node_states:
    node_states[i]['t'] = 0
    node_states[i]['t+1'] = 0
    node_states[i]['susedi'] = susedi(i)

pb = [1 - beta, beta]
pm = [1 - mu, mu]
pe = [1 - ep, ep]

R0=[]
```

```
nsim=1000
for sim in range(nsim):
    print(sim)

    infected = rd.choice(list(node_states.keys()))
    node_states[infected]['t'] = 1
    r0=0
    exposed=[]
    stop=0
    for t in range(1,timestep):

        for i in node_states:
            if node_states[i]['t'] == 1: # if infected
                node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0] # verovatnoca za oporavak

                for j in node_states[i]['susedi']: # za njegove susede
                    if node_states[j]['t+1']== 0: # if stanje suseda susceptible moze se zaraziti nekom verovatnocom i
                        ↪ postati exposed
                        st1= nrd.choice([0, 2], 1, p=pb)[0]
                        node_states[j]['t+1'] = st1
                        if i==infected and st1==2:
                            exposed.append(j)

            elif node_states[i]['t'] == 2:
                st2=nrd.choice([2, 1], 1, p=pe)
                node_states[i]['t+1'] = st2

                if i in exposed and st2==1:
                    r0=r0+1

            elif node_states[i]['t'] == 3:
                node_states[i]['t+1'] = 3

        if node_states[infected]['t+1']==3:
            stop=1

        if stop==1:
            break
        for i in node_states:
            node_states[i]['t'] = node_states[i]['t+1']

    R0.append(r0)

    for k in node_states: # cisti posle svake simulacije
        node_states[k]['t'] = 0
        node_states[k]['t+1'] = 0


fr=Counter(R0)

for f in fr:
    fr[f]=fr[f]/nsim

print(np.mean(R0))
print (np.std(R0))

lists = sorted(fr.items())

x, y = zip(*lists)

print(y)
plt.suptitle('SEIR␣model␣' +r'$\beta$=␣'+str(beta)+r'␣$\mu$=␣'+str(mu) + r'␣$\epsilon$=␣' + str(ep))
plt.bar(x, y,color='#267bb8cf',edgecolor='#001321cf')
plt.xlabel('R0−Broj␣sekundarnih␣slucajeva')
plt.ylabel('Frekfrencija')
plt.show()
```

Socijalno distanciranje (od prvog dana) SIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import random as rd
import numpy.random as nrd


def rmsoc(sus,proc): #funkcija koja uklanja kontakte u zavisnosti od procenta
    rd=int(len(sus)*proc/100)
    newsus=sus
    if rd!=len(sus):
        newsus=np.ndarray.tolist(nrd.choice(sus,len(sus)-rd,replace=False))
    else:
        newsus=newsus
    return newsus


proc=[0,20,30,40,50,60,70,80]

timestep = 100

beta=0.05
mu=0.07

pb=[1-beta,beta] #verovatnoca prelaska s-i
pm=[1-mu,mu] #verovatnoca prelaska i-r

node_states = json.load(open("2_RHOM_dict.txt")) #otvara fajl sa podacima mreze
ns_copy = json.load(open("2_RHOM_dict.txt"))
ips={}

for pr in proc:
    ips[pr]={}
    print(pr)
    ls = []
    lr = []
    li = []

    #node_states=ns_copy
    ''' # koristiti ovo parce ako se mreza generise pre simulacije
    for i in node_states:
        for t in range(timestep):
            node_states[i]['susedi'][t] = ns_copy[i]['susedi'][t]
            node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t], pr)
    '''


    for sim in range(100):
        print('//////////'+str(sim))
        St = {} # bice formata S={Vremenski trenutak : [susceptible stanja]}
        It = {}
        Rt = {}

            #koristiti ovo parce ako se mreza generise u svakoj simulaciji
        for i in node_states:
            for t in range(timestep):
                node_states[i]['susedi'][t]=ns_copy[i]['susedi'][t]
                node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t],pr)

        infected = rd.choice(list(node_states.keys()))
        node_states[infected]['t'] = 1

        for t in range(1,timestep):

            St[t] = []
            It[t] = []
            Rt[t] = []
            for i in node_states:
                if node_states[i]['susedi'][t] != []: #ako ima susede u tom trenutku,za svaki slucaj

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0]

                        for j in node_states[i]['susedi'][t - 1]:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=pb)[0]
```

```
                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = 2

                elif node_states[i]['susedi'][t] == []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = 2

                node_states[i]['t'] = node_states[i]['t+1']


            for i in node_states:
                if node_states[i]['t+1'] == 0:
                    St[t].append(0)
                elif node_states[i]['t+1'] == 1:
                    It[t].append(1)
                elif node_states[i]['t+1'] == 2:
                    Rt[t].append(2)

        S = {}
        I = {}
        R = {}
        for t in St.keys():
            S[t] = len(St[t])
            I[t] = len(It[t])
            R[t] = len(Rt[t])

        ls.append(list(S.values()))
        li.append(list(I.values()))
        lr.append(list(R.values()))

        for k in node_states: # cisti posle svake simulacije
            node_states[k]['t'] = 0
            node_states[k]['t+1'] = 0


    ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
    yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
    yi = np.mean(li, axis=0) / len(list(node_states.keys()))

    ips[pr]['S']=ys
    ips[pr]['I']=yi
    ips[pr]['R']=yr


markers=['o','v','^','s','p','P','X','D']
x=list(range(1,timestep))
fig, (ax1,ax2,ax3) = plt.subplots(3)
fig.suptitle('SIR model '+r'$\beta$='+str(beta)+r' $\mu$='+str(mu) )


for pr in range(8):
    ax1.plot(x,ips[proc[pr]]['S'],'#177c99ff',marker=markers[pr],markeredgecolor="#0e5468ff",markevery=3,label=str(
        ↪ proc[pr])+'%')
    ax2.plot(x,ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr],markeredgecolor="#bb2226ff",markevery=3, label=
        ↪ str(proc[pr]) + '%')
    ax3.plot(x, ips[proc[pr]]['R'], '#44bd7bff', marker=markers[pr],markeredgecolor="#009d6bff",markevery=3, label=
        ↪ str(proc[pr]) + '%')

ax1.set( ylabel='Populacija')
ax1.set_title('Susceptible populacija')
ax1.legend()
ax2.set_title('Infected populacija')
ax2.legend()
ax2.set( ylabel='Populacija')
ax3.set_title('Recovered populacija')
ax3.legend()
ax3.set(xlabel='Vreme', ylabel='Populacija')

plt.show()
```

Socijalno distanciranje (od prvog dana) SEIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd

def rmsoc(sus,proc): #funkcija koja uklanja kontakte u zavisnosti od procenta
    rd=int(len(sus)*proc/100)
    newsus=sus
    if rd!=len(sus):
        newsus=np.ndarray.tolist(nrd.choice(sus,len(sus)-rd,replace=False))
    else:
        newsus=newsus
    return newsus

proc=[0,20,30,40,50,60,70,80]

timestep = 100

beta=0.05
mu=0.07
ep=0.3
pb=[1-beta,beta]
pm=[1-mu,mu]
pe=[1-ep,ep]
node_states = json.load(open("2_RHOM_dict.txt"))
ns_copy = json.load(open("2_RHOM_dict.txt"))

ips={}
for pr in proc:
    ips[pr]={}
    print(pr)
    ls = []
    lr = []
    li = []
    le = []

    ''' #koristiti ovo parce ako se mreza generise pre simulacija
    for i in node_states:
        for t in range(timestep):
            node_states[i]['susedi'][t] = ns_copy[i]['susedi'][t]
            node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t], pr)
    '''

    for sim in range(100):
        print('//////////'+str(sim))
        St = {}
        Et = {}
        It = {}
        Rt = {}

        for i in node_states:
            for t in range(timestep):
                node_states[i]['susedi'][t] = ns_copy[i]['susedi'][t]
                node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t], pr)

        infected = rd.choice(list(node_states.keys()))
        node_states[infected]['t'] = 1
        for t in range(1,timestep):
            St[t] = []
            It[t] = []
            Et[t] = []
            Rt[t] = []
            for i in node_states:

                if node_states[i]['susedi'][t] != []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                        for j in node_states[i]['susedi'][t - 1]:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=pb)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = nrd.choice([2, 1], 1, p=pe)
```

```
                    elif node_states[i]['t'] == 3:
                        node_states[i]['t+1'] = 3

                elif node_states[i]['susedi'][t] == []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                    elif node_states[i]['t'] == 3:
                        node_states[i]['t+1'] = 3

            for i in node_states:
                node_states[i]['t'] = node_states[i]['t+1']
                if node_states[i]['t+1'] == 0:
                    St[t].append(0)
                elif node_states[i]['t+1'] == 1:
                    It[t].append(1)
                elif node_states[i]['t+1'] == 2:
                    Et[t].append(2)
                elif node_states[i]['t+1'] == 3:
                    Rt[t].append(3)

        S = {}
        I = {}
        E = {}
        R = {}
        for t in St.keys():
            S[t] = len(St[t])
            I[t] = len(It[t])
            R[t] = len(Rt[t])
            E[t] = len(Et[t])

        ls.append(list(S.values()))
        li.append(list(I.values()))
        lr.append(list(R.values()))
        le.append(list(E.values()))

        for k in node_states: # cisti posle svake simulacije
            node_states[k]['t'] = 0
            node_states[k]['t+1'] = 0

    ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
    yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
    yi = np.mean(li, axis=0) / len(list(node_states.keys()))
    ye = np.mean(le, axis=0) / len(list(node_states.keys()))

    ips[pr]['S']=ys
    ips[pr]['I']=yi
    ips[pr]['E']=ye
    ips[pr]['R']=yr


markers=['o','v','^','s','p','P','X','D']
x=list(range(timestep))
fig, axs = plt.subplots(2, 2)
fig.suptitle('SEIR model' +r'$\beta$='+str(beta)+r' $\mu$='+str(mu) + r' $\epsilon$=' + str(ep))
for pr in range(8):
    axs[0, 0].plot(x,ips[proc[pr]]['S'],'#177c99ff',marker=markers[pr],markeredgecolor="#0e5468ff",markevery=3,label=
        ↪ str(proc[pr])+'%')
    axs[0, 1].plot(x,ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr],markeredgecolor="#bb2226ff",markevery=3,
        ↪ label=str(proc[pr]) + '%')
    axs[1, 0].plot(x, ips[proc[pr]]['E'], '#8643b0ff', marker=markers[pr], markeredgecolor="#8341acff", markevery=3,
        ↪ label=str(proc[pr]) + '%')
    axs[1, 1].plot(x, ips[proc[pr]]['R'], '#44bd7bff', marker=markers[pr],markeredgecolor="#009d6bff",markevery=3,
        ↪ label=str(proc[pr]) + '%')


axs[0, 0].set_title('Susceptible populacija')
axs[0, 0].legend()
axs[0, 1].set_title('Infected populacija')
axs[0, 1].legend()
axs[1, 0].set_title('Exposed populacija')
axs[1, 0].legend()
axs[1, 1].set_title('Recovered populacija')
axs[1, 1].legend()

for ax in axs.flat:
```

```
    ax.set(xlabel='Vreme', ylabel='Populacija')

plt.show()
```

Socijalno distanciranje (od t_m-tog dana) SIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import random as rd
import numpy.random as nrd

def rmsoc(sus,proc): #funkcija koja uklanja kontakte u zavisnosti od procenta
    rd=int(len(sus)*proc/100)
    newsus=sus
    if rd!=len(sus):
        newsus=np.ndarray.tolist(nrd.choice(sus,len(sus)-rd,replace=False))
    else:
        newsus=newsus
    return newsus

proc=[0,20,30,40,50,60,70,80]

timestep = 100
tm=5 #trenutak u kome krece redukcija mreze
beta=0.05
mu=0.07

pb=[1-beta,beta] #verovatnoca prelaska s-i
pm=[1-mu,mu] #verovatnoca prelaska i-r

node_states = json.load(open("2_RHOM_dict.txt"))
ns_copy = json.load(open("2_RHOM_dict.txt"))
ips={}

for pr in proc:
    ips[pr]={}
    print(pr)
    ls = []
    lr = []
    li = []

    #koristiti ovo parce ako je u svakoj simulaciji ista mreza
    for i in node_states:
        for t in range(timestep):
            node_states[i]['susedi'][t] = ns_copy[i]['susedi'][t]
            if t >= tm:
                node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t], pr)


    for sim in range(100):
        print('//////////'+str(sim))
        St = {} # bice formata S={Vremenski trenutak : [susceptible stanja]}
        It = {}
        Rt = {}

        '''
        # koristiti ovo parce ako je u svakoj simulaciji nova mreza
        for i in node_states:
            for t in range(timestep):
                node_states[i]['susedi'][t]=ns_copy[i]['susedi'][t]
                if t>=tm:
                    node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t],pr)
        '''

        infected = rd.choice(list(node_states.keys()))
        node_states[infected]['t'] = 1
        for t in range(1,timestep):
            St[t] = []
            It[t] = []
            Rt[t] = []

            for i in node_states:
                if node_states[i]['susedi'][t] != []: # prvo proverimo jel ima susede uopste
```

```
                    if node_states[i]['t'] == 1: # if infected
                        node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0] # verovatnoca za oporavak

                        for j in node_states[i]['susedi'][t - 1]: # za njegove susede
                            if node_states[j]['t+1'] == 0: # if stanje suseda susceptible moze se zaraziti nekom
                                ↪ verovatnocom i postati exposed
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=pb)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = 2

                elif node_states[i]['susedi'][t] == []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = 2


            for i in node_states:
                node_states[i]['t'] = node_states[i]['t+1']
                if node_states[i]['t+1'] == 0:
                    St[t].append(0)
                elif node_states[i]['t+1'] == 1:
                    It[t].append(1)
                elif node_states[i]['t+1'] == 2:
                    Rt[t].append(2)

        S = {}
        I = {}
        R = {}
        for t in St.keys():
            S[t] = len(St[t])
            I[t] = len(It[t])
            R[t] = len(Rt[t])

        ls.append(list(S.values()))
        li.append(list(I.values()))
        lr.append(list(R.values()))

        for k in node_states: # cisti posle svake simulacije
            node_states[k]['t'] = 0
            node_states[k]['t+1'] = 0

    ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
    yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
    yi = np.mean(li, axis=0) / len(list(node_states.keys()))

    ips[pr]['S']=ys
    ips[pr]['I']=yi
    ips[pr]['R']=yr


markers=['o','v','^','s','p','P','X','D']
x=list(range(1,timestep))

for pr in range(8):
    plt.plot(x,ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr],markeredgecolor="#bb2226ff",markevery=3, label=str
        ↪ (proc[pr]) + '%')


plt.title('SIR␣model␣'+r'$\beta$=␣'+str(beta)+r'␣$\mu$=␣'+str(mu)+'\n'+r'$t_m$=␣'+str(tm)+ "␣Pre" )
plt.legend()
plt.xlabel('Vreme')
plt.ylabel('Populacija')

plt.show()
```

Socijalno distanciranje (od t_m-tog dana) SEIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd

def rmsoc(sus,proc): #funkcija koja uklanja kontakte u zavisnosti od procenta
    rd=int(len(sus)*proc/100)
    newsus=sus
    if rd!=len(sus):
        newsus=np.ndarray.tolist(nrd.choice(sus,len(sus)-rd,replace=False))
    else:
        newsus=newsus
    return newsus

proc=[0,20,30,40,50,60,70,80]

timestep = 100
tm=20
beta=0.05
mu=0.07
ep=0.3
pb=[1-beta,beta]
pm=[1-mu,mu]
pe=[1-ep,ep]

node_states = json.load(open("2_RHOM_dict.txt"))
ns_copy = json.load(open("2_RHOM_dict.txt"))

ips={}
for pr in proc:
    ips[pr]={}
    print(pr)
    ls = []
    lr = []
    li = []
    le = []

    for i in node_states:
        for t in range(timestep):
            node_states[i]['susedi'][t] = ns_copy[i]['susedi'][t]
            if t >= tm:
                node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t], pr)

    for sim in range(100):
        print('//////////'+str(sim))
        St = {} # bice formata S={Vremenski trenutak : [susceptible stanja]}
        Et = {}
        It = {}
        Rt = {}

        '''
        for i in node_states:
            for t in range(timestep):
                node_states[i]['susedi'][t] = ns_copy[i]['susedi'][t]
                if t >= tm:
                    node_states[i]['susedi'][t] = rmsoc(node_states[i]['susedi'][t], pr)
        '''

        infected = rd.choice(list(node_states.keys()))
        node_states[infected]['t'] = 1
        for t in range(1,timestep):
            St[t] = []
            It[t] = []
            Et[t] = []
            Rt[t] = []

            for i in node_states:
                if node_states[i]['susedi'][t] != []: # prvo proverimo jel ima susede uopste

                    if node_states[i]['t'] == 1: # if infected
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0] # verovatnoca za oporavak

                        for j in node_states[i]['susedi'][t - 1]: # za njegove susede
                            if node_states[j]['t+1'] == 0: # if stanje suseda susceptible moze se zaraziti nekom
```

```
                                ↪ verovatnocom i postati exposed
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=pb)[0]

                        elif node_states[i]['t'] == 2:
                            node_states[i]['t+1'] = nrd.choice([2, 1], 1, p=pe)

                        elif node_states[i]['t'] == 3:
                            node_states[i]['t+1'] = 3

                    elif node_states[i]['susedi'][t] == []:

                        if node_states[i]['t'] == 1:
                            node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                        elif node_states[i]['t'] == 3:
                            node_states[i]['t+1'] = 3

                for i in node_states:
                    node_states[i]['t'] = node_states[i]['t+1']
                    if node_states[i]['t+1'] == 0:
                        St[t].append(0)
                    elif node_states[i]['t+1'] == 1:
                        It[t].append(1)
                    elif node_states[i]['t+1'] == 2:
                        Et[t].append(2)
                    elif node_states[i]['t+1'] == 3:
                        Rt[t].append(3)

            S = {}
            I = {}
            E = {}
            R = {}
            for t in St.keys():
                S[t] = len(St[t])
                I[t] = len(It[t])
                R[t] = len(Rt[t])
                E[t] = len(Et[t])

            ls.append(list(S.values()))
            li.append(list(I.values()))
            lr.append(list(R.values()))
            le.append(list(E.values()))

            for k in node_states: # cisti posle svake simulacije
                node_states[k]['t'] = 0
                node_states[k]['t+1'] = 0

        ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
        yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
        yi = np.mean(li, axis=0) / len(list(node_states.keys()))
        ye = np.mean(le, axis=0) / len(list(node_states.keys()))

        ips[pr]['S']=ys
        ips[pr]['I']=yi
        ips[pr]['E']=ye
        ips[pr]['R']=yr

markers = ['o', 'v', '^', 's', 'p', 'P', 'X', 'D']
x = list(range(1, timestep))

for pr in range(8):
    plt.plot(x, ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr], markeredgecolor="#bb2226ff", markevery=3, label=
        ↪ str(proc[pr]) + '%')
plt.title('SEIR␣model␣' +r'$\beta$=␣'+str(beta)+r'␣$\mu$=␣'+str(mu) + r'␣$\epsilon$=␣' + str(ep)+'\n'+r'
    ↪ $t_m$=␣'+str(tm)+'␣Pre')
plt.legend()
plt.xlabel('Vreme')
plt.ylabel('Populacija')

plt.show()
```

Nošenje maski od prvog dana SIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd

def newprob(prob,mp): #prob ce biti pb a mp verovatnoca za odgovarajucu situaciju maske
    x=prob-prob*mp/100
    newprob=[1-x,x]
    return newprob

def maskpr(nstates,proc): #funkcija koja stavlja maske
    mpop=int(len(nstates.keys())*proc/100)
    masked=np.ndarray.tolist(nrd.choice(list(nstates.keys()),mpop,replace=False))
    for m in masked:
        nstates[m]['mask']=True
    return nstates

proc=list(range(0,80,10))
timestep = 100
beta=0.05
mu=0.07

pb=[1-beta,beta] #verovatnoca prelaska s-i
pm=[1-mu,mu] #verovatnoca prelaska i-r

node_states = json.load(open("2_RHOM_dict.txt"))
for node in node_states:
    node_states[node]['mask']=False

ips={}

for pr in proc:
    ips[pr]={}
    print(pr)
    ls = []
    lr = []
    li = []

''' #koristiti ovo parce ako se mreza modifikuje pre simulacije
  for node in node_states:
            node_states[node]['mask'] = False
        maskpr(node_states, pr)
 '''
    for sim in range(100):
        print('////'+str(sim))
        St = {}
        It = {}
        Rt = {}

            #koristiti ovo parce ako se mreza modifikuje u svakoj situaciji
        for node in node_states:
            node_states[node]['mask'] = False
        maskpr(node_states, pr)

        infected = rd.choice(list(node_states.keys()))
        node_states[infected]['t'] = 1
        for t in range(1,timestep):
            St[t] = []
            It[t] = []
            Rt[t] = []

            for i in node_states:

                if node_states[i]['susedi'][t] != []:#prvo proverimo jel ima susede uopste

                    if node_states[i]['t'] == 1: # if infected
                        node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0] # verovatnoca za oporavak

                    for j in node_states[i]['susedi'][t-1]: # za njegove susede
                        if node_states[i]['mask']==True and node_states[j]['mask']==True:
                            if node_states[j]['t+1']==0:
                                node_states[j]['t+1']= nrd.choice([0, 1], 1, p=newprob(beta,98.5))[0]
                        elif node_states[i]['mask']==False and node_states[j]['mask']==True:
                            if node_states[j]['t+1'] ==0:
                                node_states[j]['t+1']= nrd.choice([0, 1], 1, p=newprob(beta, 70))[0]
```

```
                        elif node_states[i]['mask']==True and node_states[j]['mask']==False:
                            if node_states[j]['t+1'] ==0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=newprob(beta,95))[0]

                        else:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=pb)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1']= 2


                elif node_states[i]['susedi'][t] == []:

                    if node_states[i]['t'] == 1:
                     node_states[i]['t+1'] = nrd.choice([1,2], 1, p=pm)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1']= 2

            for i in node_states:
                node_states[i]['t']=node_states[i]['t+1']
                if node_states[i]['t+1'] == 0:
                    St[t].append(0)
                elif node_states[i]['t+1'] == 1:
                    It[t].append(1)
                elif node_states[i]['t+1']== 2:
                    Rt[t].append(2)

        S = {}
        I = {}
        R = {}
        for t in St.keys():
            S[t] = len(St[t])
            I[t] = len(It[t])
            R[t] = len(Rt[t])

        ls.append(list(S.values()))
        li.append(list(I.values()))
        lr.append(list(R.values()))

        for k in node_states:
            node_states[k]['t'] = 0
            node_states[k]['t+1'] = 0

    ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
    yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
    yi = np.mean(li, axis=0) / len(list(node_states.keys()))

    ips[pr]['S']=ys
    ips[pr]['I']=yi
    ips[pr]['R']=yr

markers=['o','v','^','s','p','P','X','D']
x=list(range(1,timestep))
fig, (ax1,ax2,ax3) = plt.subplots(3)
fig.suptitle('SIR model '+r'$\beta$= '+str(beta)+r' $\mu$= '+str(mu) )

for pr in range(len(proc)):
    ax1.plot(x,ips[proc[pr]]['S'],'#177c99ff',marker=markers[pr],markeredgecolor="#0e5468ff",markevery=3,label=str(
        ↪ proc[pr])+'%')
    ax2.plot(x,ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr],markeredgecolor="#bb2226ff",markevery=3, label=
        ↪ str(proc[pr]) + '%')
    ax3.plot(x, ips[proc[pr]]['R'], '#44bd7bff', marker=markers[pr],markeredgecolor="#009d6bff",markevery=3, label=
        ↪ str(proc[pr]) + '%')

ax1.set( ylabel='Populacija')
ax1.set_title('Susceptible populacija')
ax1.legend()
ax2.set_title('Infected populacija')
ax2.legend()
ax2.set( ylabel='Populacija')
ax3.set_title('Recovered populacija')
ax3.legend()
ax3.set(xlabel='Vreme', ylabel='Populacija')

plt.show()
```

Nošenje maski od prvog dana SEIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd

def newprob(prob,mp): #prob ce biti pb a mp verovatnoca za odgovarajucu situaciju maske
    x=prob-prob*mp/100
    newprob=[1-x,x]
    return newprob

def maskpr(nstates,proc):
    mpop=int(len(nstates.keys())*proc/100)

    masked=np.ndarray.tolist(nrd.choice(list(nstates.keys()),mpop,replace=False))
    for m in masked:
        nstates[m]['mask']=True
    return nstates

proc=list(range(0,80,10))
timestep = 100
beta=0.05
mu=0.07
ep=0.3
pb=[1-beta,beta] #verovatnoca prelaska s-i
pm=[1-mu,mu] #verovatnoca prelaska i-r
pe=[1-ep,ep]
node_states = json.load(open("2_RHOM_dict.txt"))
for node in node_states:
    node_states[node]['mask']=False

ips={}

for pr in proc:
    ips[pr]={}
    print(pr)
    ls = []
    lr = []
    li = []
    le = []

    '''
    for node in node_states:
        node_states[node]['mask'] = False

    maskpr(node_states, pr)
    '''

    for sim in range(100):

        St = {}
        It = {}
        Rt = {}
        Et = {}

        for node in node_states:
            node_states[node]['mask'] = False

        maskpr(node_states, pr)

        infected = rd.choice(list(node_states.keys()))
        node_states[infected]['t'] = 1
        for t in range(1,timestep):
            St[t] = []
            It[t] = []
            Rt[t] = []
            Et[t] = []

            for i in node_states:

                if node_states[i]['susedi'][t] != []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]
```

```
                    for j in node_states[i]['susedi'][t - 1]:
                        if node_states[i]['mask'] == True and node_states[j]['mask'] == True:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=newprob(beta, 98.5))[0]
                        elif node_states[i]['mask'] == False and node_states[j]['mask'] == True:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=newprob(beta, 70))[0]
                        elif node_states[i]['mask'] == True and node_states[j]['mask'] == False:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=newprob(beta, 95))[0]

                        else:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=pb)[0]

                elif node_states[i]['t'] == 2:
                    node_states[i]['t+1'] = nrd.choice([2, 1], 1, p=pe)

                elif node_states[i]['t'] == 3:
                    node_states[i]['t+1'] = 3

            elif node_states[i]['susedi'][t] == []:

                if node_states[i]['t'] == 1:
                    node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                elif node_states[i]['t'] == 3:
                    node_states[i]['t+1'] = 3

        for i in node_states:
            node_states[i]['t']=node_states[i]['t+1']
            if node_states[i]['t+1'] == 0:
                St[t].append(0)
            elif node_states[i]['t+1'] == 1:
                It[t].append(1)
            elif node_states[i]['t+1']== 3:
                Rt[t].append(3)
            elif node_states[i]['t+1']== 2:
                Et[t].append(2)
    S = {}
    I = {}
    R = {}
    E={}
    for t in St.keys():
        S[t] = len(St[t])
        I[t] = len(It[t])
        R[t] = len(Rt[t])
        E[t] = len(Et[t])

    ls.append(list(S.values()))
    li.append(list(I.values()))
    lr.append(list(R.values()))
    le.append(list(E.values()))

    for k in node_states:
        node_states[k]['t'] = 0
        node_states[k]['t+1'] = 0

ys = np.mean(ls, axis=0) / len(list(node_states.keys()))
yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
yi = np.mean(li, axis=0) / len(list(node_states.keys()))
ye = np.mean(le, axis=0) / len(list(node_states.keys()))

ips[pr]['S'] = ys
ips[pr]['I'] = yi
ips[pr]['E'] = ye
ips[pr]['R'] = yr

markers = ['o', 'v', '^', 's', 'p', 'P', 'X', 'D']
x = list(range(1,timestep))
fig, axs = plt.subplots(2, 2)
fig.suptitle('SEIR␣model' + r'$\beta$=␣' + str(beta) + r'␣$\mu$=␣' + str(mu) + r'␣$\epsilon$=␣' + str(ep))
for pr in range(8):
    axs[0, 0].plot(x, ips[proc[pr]]['S'], '#177c99ff', marker=markers[pr], markeredgecolor="#0e5468ff", markevery=3,
        ↪ label=str(proc[pr]) + '%')
    axs[0, 1].plot(x, ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr], markeredgecolor="#bb2226ff", markevery=3,
        ↪ label=str(proc[pr]) + '%')
    axs[1, 0].plot(x, ips[proc[pr]]['E'], '#8643b0ff', marker=markers[pr], markeredgecolor="#8341acff", markevery=3,
```

```
        ↪ label=str(proc[pr]) + '%')
    axs[1, 1].plot(x, ips[proc[pr]]['R'], '#44bd7bff', marker=markers[pr], markeredgecolor="#009d6bff", markevery=3,
        ↪ label=str(proc[pr]) + '%')

axs[0, 0].set_title('Susceptible populacija')
axs[0, 0].legend()
axs[0, 1].set_title('Infected populacija')
axs[0, 1].legend()
axs[1, 0].set_title('Exposed populacija')
axs[1, 0].legend()
axs[1, 1].set_title('Recovered populacija')
axs[1, 1].legend()

for ax in axs.flat:
    ax.set(xlabel='Vreme', ylabel='Populacija')

plt.show()
```

Nošenje maski od t_m-tog dana SIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd


def newprob(prob, mp): # prob ce biti pb a mp verovatnoca za odgovarajucu situaciju maske
    x = prob - prob * mp / 100
    newprob = [1 - x, x]
    return newprob

def maskpr(nstates, proc):
    mpop = int(len(nstates.keys()) * proc / 100)

    masked = np.ndarray.tolist(nrd.choice(list(nstates.keys()), mpop, replace=False))
    for m in masked:
        nstates[m]['mask'] = True
    return nstates

proc = list(range(0, 80, 10))
timestep = 100
tm=5
beta = 0.05
mu = 0.07

pb = [1 - beta, beta] # verovatnoca prelaska s-i
pm = [1 - mu, mu] # verovatnoca prelaska i-r

node_states = json.load(open("2_RHOM_dict.txt"))
for node in node_states:
    node_states[node]['mask'] = False

ips = {}

for pr in proc:
    ips[pr] = {}
    print(pr)
    ls = []
    lr = []
    li = []

        #koristiti ovo parce ako se mreza modifikuje pre simulacija
    for node in node_states:
        node_states[node]['mask'] = False

    maskpr(node_states, pr)

    for sim in range(100):
        print('////' + str(sim))
        St = {} # bice formata S={Vremenski trenutak : [susceptible stanja]}
        It = {}
        Rt = {}

        '''#koristiti ovo parce ako se mreza modifikuje u svakoj simulaciji
```

```
for node in node_states:
    node_states[node]['mask'] = False

        maskpr(node_states, pr)
'''

infected = rd.choice(list(node_states.keys()))
node_states[infected]['t'] = 1
for t in range(1,timestep):
    St[t] = []
    It[t] = []
    Rt[t] = []
    # print('////'+str(t))
    if t < tm:
        for i in node_states:
            if node_states[i]['susedi'][t] != []:

                if node_states[i]['t'] == 1:
                    node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0]

                    for j in node_states[i]['susedi'][t - 1]:
                        if node_states[j]['t+1'] == 0:
                            node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=pb)[0]

                elif node_states[i]['t'] == 2:
                    node_states[i]['t+1'] = 2

            elif node_states[i]['susedi'][t] == []:

                if node_states[i]['t'] == 1:
                    node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0]

                elif node_states[i]['t'] == 2:
                    node_states[i]['t+1'] = 2
    else:
        for i in node_states:

            if node_states[i]['susedi'][t] != []: # prvo proverimo jel ima susede uopste

                if node_states[i]['t'] == 1: # if infected
                    node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0] # verovatnoca za oporavak

                    for j in node_states[i]['susedi'][t - 1]: # za njegove susede
                        if node_states[i]['mask'] == True and node_states[j]['mask'] == True:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=newprob(beta, 98.5))[0]
                        elif node_states[i]['mask'] == False and node_states[j]['mask'] == True:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=newprob(beta, 70))[0]
                        elif node_states[i]['mask'] == True and node_states[j]['mask'] == False:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=newprob(beta, 95))[0]

                        else:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 1], 1, p=pb)[0]

                elif node_states[i]['t'] == 2:
                    node_states[i]['t+1'] = 2


            elif node_states[i]['susedi'][t] == []:

                if node_states[i]['t'] == 1:
                    node_states[i]['t+1'] = nrd.choice([1, 2], 1, p=pm)[0]

                elif node_states[i]['t'] == 2:
                    node_states[i]['t+1'] = 2

    for i in node_states:
        node_states[i]['t'] = node_states[i]['t+1']
        if node_states[i]['t+1'] == 0:
            St[t].append(0)
        elif node_states[i]['t+1'] == 1:
            It[t].append(1)
        elif node_states[i]['t+1'] == 2:
            Rt[t].append(2)

S = {}
```

```
        I = {}
        R = {}
        for t in St.keys():
            S[t] = len(St[t])
            I[t] = len(It[t])
            R[t] = len(Rt[t])

        ls.append(list(S.values()))
        li.append(list(I.values()))
        lr.append(list(R.values()))

        for k in node_states:
            node_states[k]['t'] = 0
            node_states[k]['t+1'] = 0

    ys = np.mean(ls, axis=0) / len(list(node_states.keys())) # racuna srednje vrednosti svake kolone
    yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
    yi = np.mean(li, axis=0) / len(list(node_states.keys()))

    ips[pr]['S'] = ys
    ips[pr]['I'] = yi
    ips[pr]['R'] = yr

markers=['o','v','^','s','p','P','X','D']
x=list(range(1,timestep))

for pr in range(8):
    plt.plot(x,ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr],markeredgecolor="#bb2226ff",markevery=3, label=str
        ↪ (proc[pr]) + '%')

plt.title('SIR model '+r'$\beta$= '+str(beta)+r' $\mu$= '+str(mu)+'\n'+r'$t_m$= '+str(tm)+' Pre' )
plt.legend()
plt.xlabel('Vreme')
plt.ylabel('Populacija')

plt.show()
```

Nošenje maski od t_m-tog dana SEIR

```
import json
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import random as rd
import numpy.random as nrd

def newprob(prob,mp): #prob ce biti pb a mp verovatnoca za odgovarajucu situaciju maske
    x=prob-prob*mp/100
    newprob=[1-x,x]
    return newprob

def maskpr(nstates,proc):
    mpop=int(len(nstates.keys())*proc/100)

    masked=np.ndarray.tolist(nrd.choice(list(nstates.keys()),mpop,replace=False))
    for m in masked:
        nstates[m]['mask']=True
    return nstates

proc=list(range(0,80,10))

timestep = 100
tm=20
beta=0.05
mu=0.07
ep=0.3
pb=[1-beta,beta] #verovatnoca prelaska s-i
pm=[1-mu,mu] #verovatnoca prelaska i-r
pe=[1-ep,ep]
node_states = json.load(open("2_RHOM_dict.txt"))
for node in node_states:
    node_states[node]['mask']=False

ips={}

for pr in proc:
```

```
ips[pr]={}
print(pr)
ls = []
lr = []
li = []
le = []

for node in node_states:
    node_states[node]['mask'] = False

maskpr(node_states, pr)

for sim in range(100):

    St = {}
    It = {}
    Rt = {}
    Et = {}
    '''
    for node in node_states:
        node_states[node]['mask'] = False

    maskpr(node_states, pr)
    '''
    infected = rd.choice(list(node_states.keys()))
    node_states[infected]['t'] = 1
    for t in range(1,timestep):
        St[t] = []
        It[t] = []
        Rt[t] = []
        Et[t] = []
        #print('////'+str(t))
        if t<tm:
            for i in node_states:

                if node_states[i]['susedi'][t] != []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                        for j in node_states[i]['susedi'][t - 1]:
                            if node_states[j]['t+1'] == 0:
                                node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=pb)[0]

                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = nrd.choice([2, 1], 1, p=pe)

                    elif node_states[i]['t'] == 3:
                        node_states[i]['t+1'] = 3

                elif node_states[i]['susedi'][t] == []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                    elif node_states[i]['t'] == 3:
                        node_states[i]['t+1'] = 3
        else:
            for i in node_states:

                if node_states[i]['susedi'][t] != []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                        for j in node_states[i]['susedi'][t - 1]:
                            if node_states[i]['mask'] == True and node_states[j]['mask'] == True:
                                if node_states[j]['t+1'] == 0:
                                    node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=newprob(beta, 98.5))[0]
                            elif node_states[i]['mask'] == False and node_states[j]['mask'] == True:
                                if node_states[j]['t+1'] == 0:
                                    node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=newprob(beta, 70))[0]
                            elif node_states[i]['mask'] == True and node_states[j]['mask'] == False:
                                if node_states[j]['t+1'] == 0:
                                    node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=newprob(beta, 95))[0]

                            else:
                                if node_states[j]['t+1'] == 0:
                                    node_states[j]['t+1'] = nrd.choice([0, 2], 1, p=pb)[0]
```

```
                    elif node_states[i]['t'] == 2:
                        node_states[i]['t+1'] = nrd.choice([2, 1], 1, p=pe)

                    elif node_states[i]['t'] == 3:
                        node_states[i]['t+1'] = 3

                elif node_states[i]['susedi'][t] == []:

                    if node_states[i]['t'] == 1:
                        node_states[i]['t+1'] = nrd.choice([1, 3], 1, p=pm)[0]

                    elif node_states[i]['t'] == 3:
                        node_states[i]['t+1'] = 3

            for i in node_states:
                node_states[i]['t']=node_states[i]['t+1']
                if node_states[i]['t+1'] == 0:
                    St[t].append(0)
                elif node_states[i]['t+1'] == 1:
                    It[t].append(1)
                elif node_states[i]['t+1']== 3:
                    Rt[t].append(3)
                elif node_states[i]['t+1']== 2:
                    Et[t].append(2)
        S = {}
        I = {}
        R = {}
        E = {}
        for t in St.keys():
            S[t] = len(St[t])
            I[t] = len(It[t])
            R[t] = len(Rt[t])
            E[t] = len(Et[t])

        ls.append(list(S.values()))
        li.append(list(I.values()))
        lr.append(list(R.values()))
        le.append(list(E.values()))

        for k in node_states:
            node_states[k]['t'] = 0
            node_states[k]['t+1'] = 0

    ys = np.mean(ls, axis=0) / len(list(node_states.keys())) # racuna srednje vrednosti svake kolone
    yr = np.mean(lr, axis=0) / len(list(node_states.keys()))
    yi = np.mean(li, axis=0) / len(list(node_states.keys()))
    ye = np.mean(le, axis=0) / len(list(node_states.keys()))

    ips[pr]['S'] = ys
    ips[pr]['I'] = yi
    ips[pr]['E'] = ye
    ips[pr]['R'] = yr

markers = ['o', 'v', '^', 's', 'p', 'P', 'X', 'D']
x = list(range(1, timestep))

for pr in range(8):
    plt.plot(x, ips[proc[pr]]['I'], '#ed4145ff', marker=markers[pr], markeredgecolor="#bb2226ff", markevery=3, label=
        ↪ str(proc[pr]) + '%')

plt.title('SEIR␣model␣' +r'$\beta$=␣'+str(beta)+r'␣$\mu$=␣'+str(mu) + r'␣$\epsilon$=␣' + str(ep)+'\n'+r'
    ↪ $t_m$=␣'+str(tm)+'␣Pre')
plt.legend()
plt.xlabel('Vreme')
plt.ylabel('Populacija')

plt.show()
```

MODEL

```
import numpy as np
import matplotlib.pyplot as plt

ks = 0.01
def funkcija(x):#funkcija f
    funk=float(1+(1-np.exp(-ks*x)))
    return funk
# parameteri
k = {}
k['Y'] = 12
k['M'] = 14
k['O'] = 7

eta = {}
eta['Y'] = 1.0 / 2.34
eta['M'] = 1.0 / 2.34
eta['O'] = 1.0 / 2.34

alpha = {}
alpha['Y'] = 1.0 / 5.06
alpha['M'] = 1.0 / 2.86
alpha['O'] = 1.0 / 2.86

mu = {}
mu['Y'] = 1.0
mu['M'] = 1.0 / 3.2
mu['O'] = 1.0 / 3.2

gamma = {}
gamma['Y'] = 0.002
gamma['M'] = 0.05
gamma['O'] = 0.36

omega = {}
omega['Y'] = 0.42
omega['M'] = 0.42
omega['O'] = 0.42

p = {} # moblity factor
p['Y'] = 0
p['M'] = 1.0
p['O'] = 0

psi = {} # psi
psi['Y'] = 1.0 / 7.0
psi['M'] = 1.0 / 7.0
psi['O'] = 1.0 / 7.0

hi = {}
hi['Y'] = 1.0 / 10.0
hi['M'] = 1.0 / 10.0
hi['O'] = 1.0 / 10.0

betaA = 0.06
betaI = 0.06

nm = {}
ro = {}
roO = {}
states = ['S', 'E', 'A', 'I', 'H', 'D', 'R']
G = ['Y', 'M', 'O']
R = {}
Pi = {}
P = {}
Ng = {}
n = {}
s = {}
z = {}
C = {}
nigeff = {}
nieff = {}
N = 163
for m in states:
    ro[m] = {}
    roO[m] = {}
```

```
    for g in G:
        ro[m][g] = {}
        roO[m][g] = {}
        for i in range(0, N+1):
            if m != 'S':
                ro[m][g][i] = 0.0
                roO[m][g][i] = 0.0
            else:
                ro[m][g][i] = 1.0
                roO[m][g][i] = 1.0

for g in G:
    Pi[g] = {}
    n[g] = {}
    P[g] = {}
    for i in range(0, N+1):
        Pi[g][i] = 0.0
        n[g][i] = 0.0
        P[g][i] = 0.0

nmh = {}
for m in ['A','I']:
    nmh[m]={}
    for h in G:
        nmh[m][h]={}
        for j in range(0,N+1):
            nmh[m][h][j]={}
            for i in range(0,N+1):
                nmh[m][h][j][i]=0.0

# Populacija i povrsine
f = open("Srbija_opstine_ind.csv", "r")
for l in f.readlines(): #n[g][i]
    #nq=len(l)
    d=l.split(";")
    n['Y'][int(d[0])]=int(d[2])
    n['M'][int(d[0])]=int(d[3])
    n['O'][int(d[0])]=int(d[4])
    s[int(d[0])]=int(d[-2])


R={} #matrica mobilnosti
for i in range(0,N+1):
    R[i]={}
    for j in range(0,N+1):
        R[i][j]=0.0

f2=open('Srbija_opstine_mobility.csv','r')
for l in f2.readlines():

    d=l.split(";")
    v1=int(d[0])
    v2=int(d[1])
    R[v1][v2]=float(d[2])
f2.close()

nuk={}
for i in range(0,N+1):
    uk=0
    for j in range(0,N+1):
        uk=uk+R[i][j]
    nuk[i]=uk

for i in range(0,N+1):
    R[i][i]=float(n['M'][i]-nuk[i])

#f.close()

#normiranje Rij
for i in range(0,N+1):
    for j in range(0,N+1):
        R[i][j]=float(R[i][j]/n['M'][i])


# matrica kontakata
f = open("C.csv", "r")
u = 0
for l in f.readlines():
    nq = len(l)
```

```
    d = l[0:nq - 1].split(";")
    if u == 0:
        C['Y'] = {}
        C['Y']['Y'] = float(d[0])
        C['Y']['M'] = float(d[1])
        C['Y']['O'] = float(d[2])
        u += 1
    else:
        if u == 1:
            C['M'] = {}
            C['M']['Y'] = float(d[0])
            C['M']['M'] = float(d[1])
            C['M']['O'] = float(d[2])
            u += 1
        else:
            C['O'] = {}
            C['O']['Y'] = float(d[0])
            C['O']['M'] = float(d[1])
            C['O']['O'] = float(d[2])

# jednacina 5.13
for g in G:
    nigeff[g] = {}
    for i in range(0, N+1):
        nigeff[g][i] = 0.0
        for j in range(0, N+1):
            if i != j:
                nigeff[g][i] = nigeff[g][i] + p[g] * R[j][i] * n[g][j]
            else:
                nigeff[g][i] = nigeff[g][i] + (1 - p[g]) * n[g][j] + p[g] * R[j][i] * n[g][j]


# jednacina 5.12
for i in range(0, N+1):
    nieff[i] = 0
    for g in G:
        nieff[i] = nieff[i] + nigeff[g][i]

#Ng za jednacinu 5.11
for g in G:
    Ng[g] = 0
    for i in range(0, N+1):
        Ng[g] = Ng[g] + n[g][i]


# jednacina 5.10
for g in G:
    z[g] = 0
    sumz = 0
    for i in range(0, N+1):
        x=float(nieff[i]/s[i])
        sumz = sumz + funkcija(x)* nigeff[g][i]
    z[g] = Ng[g] / sumz


fak = {}
for g in G:
    fak[g] = {}
    for i in range(0, N+1):
        x = float(nieff[i] / s[i])
        fak[g][i] = z[g] * k[g] * funkcija(x)

f = open("zarazeni_ini.csv", "r")
for l in f.readlines():
    nq = len(l)
    d = l[0:nq - 1].split(";")
    ro['S']['M'][int(d[0])] = 1.0 - float(d[1])
    roO['S']['M'][int(d[0])] = 1.0 - float(d[1])
    ro['A']['M'][int(d[0])] = float(d[1])
    roO['A']['M'][int(d[0])] = float(d[1])
f.close()

timestep=150

ro_sum={} #pamtice sumu svakog stanja(za svaku grupu i svaku opstinu) u svakom trenutku
for st in states:
    ro_sum[st]=[]

for t in range(timestep):
```

```
    print(t)
    NS = 0
    NA = 0
    NI = 0
    NE = 0
    NH = 0
    ND = 0
    NR = 0
    #nmhji,jednacina 14
    for m in ['A','I']:#infectious states
        for h in G:
            for j in range(0,N+1):
                for i in range(0,N+1):
                    #nmh[m][h][j][i]=0
                    if j==i:
                        nmh[m][h][j][i] =n[h][j]*roO[m][h][j]*((1-p[h])+p[h]*R[j][i])
                    else:
                        nmh[m][h][j][i] = n[h][j] * roO[m][h][j] * (p[h] * R[j][i])

    expsum=0
    for g in G:
        for i in range(0,N+1):
            sum=0
            for h in G:
                for j in range(0,N+1):
                    sum=sum+fak[g][i]*C[g][h]*(nmh['A'][h][j][i]/nigeff[h][i])*np.log(1-betaA)+fak[g][i]*C[g][h]*(nmh
                        ↪ ['I'][h][j][i]/nigeff[h][i])*np.log(1-betaI)
                #expsum=expsum+sum
            P[g][i]=1-np.exp(sum)

    for g in G:
        for i in range(0,N+1):
            Pi[g][i]=(1-p[g])*P[g][i]
            for j in range(0,N+1):
                if i!=j:
                    Pi[g][i]=Pi[g][i]+p[g]*R[i][j]*P[g][j]

    for g in G:
        for i in range(0,N+1):
            ro['S'][g][i]=roO['S'][g][i]*(1.0-Pi[g][i])
            ro['E'][g][i]=roO['S'][g][i]*Pi[g][i]+(1.0-eta[g])*roO['E'][g][i]
            ro['A'][g][i]=roO['E'][g][i]*eta[g]+(1.0-alpha[g])*roO['A'][g][i]
            ro['I'][g][i]=roO['A'][g][i]*alpha[g]+(1.0-mu[g])*roO['I'][g][i]
            ro['H'][g][i]=roO['I'][g][i]*mu[g]*gamma[g]+roO['H'][g][i]*omega[g]*(1.0-psi[g])+(1.0-omega[g])*(1.0-hi[g
                ↪ ])*roO['H'][g][i]
            ro['D'][g][i]=roO['H'][g][i]*omega[g]*psi[g]+roO['D'][g][i]
            ro['R'][g][i]=roO['I'][g][i]*mu[g]*(1.0-gamma[g])+(1.0-omega[g])*hi[g]*roO['H'][g][i]+roO['R'][g][i]

            for m in states:

                roO[m][g][i]=ro[m][g][i]
                NI = NI + int(round(roO['I'][g][i] * n[g][i]))
                NE = NE + int(round(roO['E'][g][i] * n[g][i]))
                NH = NH + int(round(roO['H'][g][i] * n[g][i]))
                ND = ND + int(round(roO['D'][g][i] * n[g][i]))
                NA = NA + int(round(roO['A'][g][i] * n[g][i]))


    print ('----------'+str(NI))
    ro_sum['I'].append(NI)
    ro_sum['E'].append(NE)
    ro_sum['H'].append(NH)
    ro_sum['D'].append(ND)
    ro_sum['A'].append(NA)


model={}
model['I']={'y':ro_sum['I']}
model['E']={'y':ro_sum['E']}
model['H']={'y':ro_sum['H']}
model['D']={'y':ro_sum['D']}
model['A']={'y':ro_sum['A']}

x=list(range(timestep))
fig,ax=plt.subplots()
fig.suptitle('Evolucija␣bolesti␣COVID-19␣u␣Srbiji')
#ax.plot(x,ro_sum['S'],'#177c99ff',label='S')
ax.plot(x,ro_sum['E'],'#b17cd3ff',label='E')
```

```
ax.plot(x,ro_sum['A'],'#f1d16aff',label='A')
ax.plot(x,ro_sum['I'],'#ed4145ff',label='I')
ax.plot(x,ro_sum['H'],'#9d9d9dff',label='H')
ax.plot(x,ro_sum['D'],'#2a2a2aff',label='D')
#ax.plot(x,ro_sum['R'],'#44bd7bff',label='R')

#ax.plot(x,ro_sum['R'],'#44bd7bff',label='R')
ax.set_xlabel('Dan')
ax.set_ylabel('Populacija')

leg=ax.legend()
plt.show()
```

R_0 MODEL

```
import numpy as np


ks = 0.01
def funkcija(x):#funkcija f
    funk=float(1+(1-np.exp(-ks*x)))
    return funk
# parameteri
k = {}
k['Y'] = 12
k['M'] = 14
k['O'] = 7

eta = {}
eta['Y'] = 1.0 / 2.34
eta['M'] = 1.0 / 2.34
eta['O'] = 1.0 / 2.34

alpha = {}
alpha['Y'] = 1.0 / 5.06
alpha['M'] = 1.0 / 2.86
alpha['O'] = 1.0 / 2.86

mu = {}
mu['Y'] = 1.0
mu['M'] = 1.0 / 3.2
mu['O'] = 1.0 / 3.2

gamma = {}
gamma['Y'] = 0.002
gamma['M'] = 0.05
gamma['O'] = 0.36

omega = {}
omega['Y'] = 0.42
omega['M'] = 0.42
omega['O'] = 0.42

p = {} # moblity factor
p['Y'] = 0
p['M'] = 1.0
p['O'] = 0

psi = {} # psi
psi['Y'] = 1.0 / 7.0
psi['M'] = 1.0 / 7.0
psi['O'] = 1.0 / 7.0

hi = {}
hi['Y'] = 1.0 / 10.0
hi['M'] = 1.0 / 10.0
hi['O'] = 1.0 / 10.0

betaA = 0.06
betaI = 0.06

nm = {}
ro = {}
roO = {}
```

```
states = ['S', 'E', 'A', 'I', 'H', 'D', 'R']
G = ['Y', 'M', 'O']
R = {}
Pi = {}
P = {}
Ng = {}
n = {}
s = {}
z = {}
C = {}
nigeff = {}
nieff = {}
N = 163
for m in states:
    ro[m] = {}
    roO[m] = {}
    for g in G:
        ro[m][g] = {}
        roO[m][g] = {}
        for i in range(0, N+1):
            if m != 'S':
                ro[m][g][i] = 0.0
                roO[m][g][i] = 0.0
            else:
                ro[m][g][i] = 1.0
                roO[m][g][i] = 1.0

for g in G:
    Pi[g] = {}
    n[g] = {}
    P[g] = {}
    for i in range(0, N+1):
        Pi[g][i] = 0.0
        n[g][i] = 0.0
        P[g][i] = 0.0

nmh = {}
for m in ['A','I']:
    nmh[m]={}
    for h in G:
        nmh[m][h]={}
        for j in range(0,N+1):
            nmh[m][h][j]={}
            for i in range(0,N+1):
                nmh[m][h][j][i]=0.0



# Populacija i povrsine
f = open("Srbija_opstine_ind.csv", "r")
for l in f.readlines(): #n[g][i]
    #nq=len(l)
    d=l.split(";")
    n['Y'][int(d[0])]=int(d[2])
    n['M'][int(d[0])]=int(d[3])
    n['O'][int(d[0])]=int(d[4])
    s[int(d[0])]=int(d[-2])


R={} #matrica mobilnosti
#matrica mobilnosti
for g in G:
    R[g]={}
    for i in range(0, N+1):
        #print(i)
        R[g][i] = {}
        for j in range(0, N+1):
            R[g][i][j] = 0.0

f2=open('Srbija_opstine_mobility.csv','r')
for l in f2.readlines():

    d=l.split(";")
    v1=int(d[0])
    v2=int(d[1])
    R['M'][v1][v2]=float(d[2])
f2.close()
```

```
nuk={}
for i in range(0,N+1):
    uk=0
    for j in range(0,N+1):
        uk=uk+R['M'][i][j]
    nuk[i]=uk

for i in range(0,N+1):
    R['M'][i][i]=float(n['M'][i]-nuk[i])


#normiranje Rij
for i in range(0,N+1):
    for j in range(0,N+1):
        R['M'][i][j]=float(R['M'][i][j]/n['M'][i])


#matrica komunikacije
f = open("C.csv", "r")
u = 0
for l in f.readlines():
    nq = len(l)
    d = l[0:nq - 1].split(";")
    if u == 0:
        C['Y'] = {}
        C['Y']['Y'] = float(d[0])
        C['Y']['M'] = float(d[1])
        C['Y']['O'] = float(d[2])
        u += 1
    else:
        if u == 1:
            C['M'] = {}
            C['M']['Y'] = float(d[0])
            C['M']['M'] = float(d[1])
            C['M']['O'] = float(d[2])
            u += 1
        else:
            C['O'] = {}
            C['O']['Y'] = float(d[0])
            C['O']['M'] = float(d[1])
            C['O']['O'] = float(d[2])

for g in G:
    nigeff[g] = {}
    for i in range(0, N+1):
        nigeff[g][i] = 0.0
        for j in range(0, N+1):
            if i != j:
                nigeff[g][i] = nigeff[g][i] + p[g] * R[g][j][i] * n[g][j]
            else:
                nigeff[g][i] = nigeff[g][i] + (1 - p[g]) * n[g][j] + p[g] * R[g][j][i] * n[g][j]
        # print(g,i,nigeff[g][i],n[g][i])

for i in range(0, N+1):
    nieff[i] = 0
    for g in G:
        nieff[i] = nieff[i] + nigeff[g][i]
#Ng za jednacinu 10
for g in G:
    Ng[g] = 0
    for i in range(0, N+1):
        Ng[g] = Ng[g] + n[g][i]


for g in G:
    z[g] = 0
    sumz = 0
    for i in range(0, N+1):
        x=float(nieff[i]/s[i])
        sumz = sumz + funkcija(x)* nigeff[g][i]
    z[g] = Ng[g] / sumz

fak = {}
for g in G:
    fak[g] = {}
    for i in range(0, N+1):
        x = float(nieff[i] / s[i])
        fak[g][i] = z[g] * k[g] * funkcija(x)
```

```
M1={}
M2={}
M3={}
M4={}

for g in G:
    print(g)
    M1[g]={}
    M2[g]={}
    M3[g]={}
    M4[g]={}
    for h in G:

        M1[g][h]= {}
        M2[g][h] = {}
        M3[g][h] = {}
        M4[g][h] = {}
        for i in range(0, N+1):
            M1[g][h][i] = {}
            M2[g][h][i] = {}
            M3[g][h][i] = {}
            M4[g][h][i] = {}

            for j in range(0, N+1):
                if i==j:

                    M1[g][h][i][j] = (1-p[g])*z[g]*k[g]*funkcija(nieff[i]/s[i])*C[g][h]*(1-p[g])*n[h][j]/nigeff[h][i]
                    M2[g][h][i][j] = (1-p[g])*z[g]*k[g]*funkcija(nieff[i]/s[i])*C[g][h]*R[h][j][i]*p[h]*n[h][j]/nigeff[h][i]
                    M3[g][h][i][j] = p[g]*R[g][i][j]*z[g]*k[g]*funkcija(nieff[i]/s[i])*C[g][h]*(1-p[g])*n[h][j]/nigeff[h][i]
                    M4[g][h][i][j] = 0.0
                    for kx in range (0, N+1):
                        M4[g][h][i][j]= M4[g][h][i][j] + p[g]*R[g][i][kx]*z[g]*k[g]*funkcija(nieff[kx]/s[kx])*C[g][h]*R[h][j][
                            ↪ kx]*p[h]*n[h][j]/nigeff[h][kx]

                else:
                    M1[g][h][i][j] = 0
                    M2[g][h][i][j] = (1 - p[g]) * z[g] * k[g] * funkcija(nieff[i] / s[i]) * C[g][h] * R[h][j][i] * p[h] * n[h][j] /
                        ↪ nigeff[h][i]
                    M3[g][h][i][j] = p[g] * R[g][i][j] * z[g] * k[g] * funkcija(nieff[i] / s[i]) * C[g][h] * (1 - p[g]) * n[h][j] /
                        ↪ nigeff[h][i]
                    M4[g][h][i][j] = 0.0
                    for kx in range(0, N + 1):
                        M4[g][h][i][j] = M4[g][h][i][j] + p[g] * R[g][i][kx] * z[g] * k[g] * funkcija(nieff[kx] / s[kx]) * C[g][
                            ↪ h] * R[h][j][kx] * p[h] * n[h][j] / nigeff[h][kx]

M={}
for g in G:
    print('M',g)
    M[g]={}
    for h in G:
        M[g][h]={}
        for i in range(0,N+1):
            M[g][h][i]={}
            for j in range(0,N+1):
                M[g][h][i][j]=M1[g][h][i][j]+M2[g][h][i][j]+M3[g][h][i][j]+M4[g][h][i][j]

bA=np.log(1.0/(1.0-betaA))
bI=np.log(1.0/(1.0-betaI))

Z={}
for g in G:
    print('Z--',g)
    Z[g]={}
    for h in G:
        Z[g][h]={}
        for i in range(0,N+1):
            Z[g][h][i]={}
            for j in range(0,N+1):
                Z[g][h][i][j]=(bA/alpha[h]+bI/mu[h])*M[g][h][i][j]

from ast import literal_eval

Zm={}
ZZ={}
for g in G:
    Zm[g]={}
    for h in G:
        Zm[g][h]={}
```

```
        udict=Z[g][h]
        Zm[g][h]=np.array([[v[j] for j in list(range(0,N+1)) ] for keys, v in udict.items()])

ZM=np.array([v[j] for j in G for keys, v in Zm.items()])
#udruzuje manje blok matrice u 3 vece
ZM1=np.concatenate((ZM[0], ZM[1],ZM[2]), axis=1)
ZM2=np.concatenate((ZM[3],ZM[4],ZM[5]), axis=1)
ZM3=np.concatenate((ZM[6],ZM[7],ZM[8]), axis=1)
  #od blok matrica pravi jednu veliku matricu
ZMall=np.concatenate((ZM1,ZM2,ZM3),axis=0)

L=np.linalg.eigvals(ZMall)
print('R0=',max(L))
```

Literatura i reference

[1] William Ogilvy Kermack and A. G. McKendrick *A contribution to the mathematical theory of epidemics* (1927)

[2] Albert-László Barabási, *Network Science* , Cambridge University Press (2016)

[3] Bruno Gonçalves , *Epidemic Modeling 101: Or why your CoVID-19 exponential fits are wrong* Medium Mag (2020)

[4] Bruno Gonçalves, *Epidemic Modeling 102: All CoVID-19 models are wrong, but some are useful* Medium Mag (2020)

[5] Brauer F. *Compartmental Models in Epidemiology* In: Brauer F., van den Driessche P., Wu J. (eds) Mathematical Epidemiology. Lecture Notes in Mathematics, vol 1945. Springer, Berlin, Heidelberg. (2008)

[6] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, Alessandro Vespignani *Epidemic processes in complex networks* Rev. Mod. Phys. 87, 925 (2015)

[7] Greenhalgh, S., Day, T. *Time-varying and state-dependent recovery rates in epidemiological models.* Infectious Disease Modelling, (2017)

[8] U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES, *Principles of Epidemiologyin Public Health Practice* An Introduction to Applied Epidemiology and Biostatistics (2006)

[9] National Academies of Sciences, Engineering, and Medicine. *Using 21st Century Science to Improve Risk-Related Evaluations.* Washington, DC: The National Academies Press (2017)

[10] Stehlé, J., Voirin, N., Barrat, A. et al. *Simulation of an SEIR infectious disease model on the dynamic contact network of conference attendees.* BMC Med 9, 87 (2011)

[11] The SocioPatterns project (*http://www.sociopatterns.org/*)

[12] O. Diekmann, J. A. P. Heesterbeek and M. G. Roberts *The construction of next-generation matrices for compartmental epidemic models* J. R. Soc. Interface 7:873–885 (2010)

[13] Leon Danon,Ashley P. Ford,Thomas House,Chris P. Jewell,Matt J. Keeling, Gareth O. Roberts,Joshua V. Ross and Matthew C. Vernon *Networks and the Epidemiology of Infectious Disease* Interdisciplinary Perspectives on Infectious Diseases,2011:284909 (2011)

[14] Petter Holme, *Fast and principled simulations of the SIR model on temporal networks* Tokyo Tech World Research Hub Initiative (WRHI), Institute of Innovative Research,Tokyo Institute of Technology (2020)

[15] Barabási, A.-L., and R. Albert *Emergence of Scaling in Random Networks* Science 509-512 (1999)

[16] Baronchelli, A., R. Ferrer-i-Cancho, R. Pastor-Satorras,N. Chater, and M. H. Christiansen *Networks in cognitive science* Trends in Cognitive Sciences (7):348-60 (2013)

[17] Newman, M. *Networks: An Introduction* Oxford University Press, New York, NY (2010)

[18] Caldarelli, G. *Scale-Free Networks: Complex Webs in Nature and Technology* Oxford University Press, Oxford (2007)

[19] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes *Critical phenomena in complex networks*Rev. Mod. Phys. 80, 1275 (2008)

[20] Henkel, M., H. Hinrichsen, and S.Lübeck*Non-equilibrium phase transition: Absorbing Phase Transitions* Springer Verlag, Netherlands (2008)

[21] Danon, L., A. P. Ford, T. House, C. P. Jewell, M. J. Keeling, G. O. Roberts, J. V. Ross, and M. C. Vernon *Networks and the epidemiology of infectious disease* Interdisciplinary Perspectives on Infectious Diseases ID 284909 (2011)

[22] Matt J Keeling and Ken T.D Eames *Networks and epidemic models* The Royal Society 2: 295–307(2005)

[23] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani *Dynamical and Correlation Properties of the Internet* Phys. Rev. Lett 87, 258701 (2001)

[24] Erzsébet Ravasz and Albert-László Barabási *Hierarchical organization in complex networks* Phys. Rev. E 67, 026112 (2003)

[25] Wasserman, S., and K. Faust , *Social Network Analysis: Methods and Applications* Cambridge University Press (1994)

[26] Santo Fortunato *Community detection in graphs* 486(3-5), 75–174 (2010)

[27] Paul L. Delamater, Erica J. Street, Timothy F. Leslie, Y. Tony Yang, and Kathryn H. Jacobsen *Complexity of the Basic Reproduction Number (R0)* Emerg Infect Dis. 25(1):1-4 (2019)

[28] Renyi Zhang,Yixin Li, Annie L. Zhang,Yuan Wang, and Mario J. Molina *Identifying airborne transmission as the dominantroute for the spread of COVID-19* PNAS 117(26):14857-14863. (2020)

[29] Alex Arenas, Wesley Cota, Jesus Gomez-Gardenes, Sergio Gómez, Clara Granell, Joan T. Matamalas, David Soriano-Panos, Benjamin Steinegger *A mathematical model for the spatiotemporal epidemic spreading of COVID19* medRxiv 2020.03.21.20040022 (2020)

[30] Ing AJ, Cocks C, Green JP *COVID-19: in the footsteps of Ernest Shackleton* Thorax 75:693-694 (2020)

[31] Sergio Gomez, Alex Arenas, Javier Borge-Holthoefer, Sandro Meloni, and Yamir Moreno. *Discrete-time Markov chain approach to contact-based disease spreading in complex networks.* EPL (Europhysics Letters) 89(3) (2010)

[32] Vasyl Palchykov, Marija Mitrović, Hang-Hyun Jo, Jari Saramäki & Raj Kumar Pan *Inferring human mobility using communication patterns* Sci Rep 4, 6174 (2014)

[33] Barthelemy, M. *Spatial networks.* Phys. Rep.499, 1–101 (2010)

[34] Krings, G., Calabrese, F., Ratti, C. & Blondel, V. D. U*rban gravity: a model for inter-city telecommunication flows.* J. Stat. Mech. L07003 (2009).

[35] Anderson, J. E. *The gravity model.* Annu. Rev. Econ., 3(1), 133-160 (2011)

[36] Sanche, S.; Lin, Y. T.; Xu, C.; Romero-Severson, E.; Hengartner, E.; Ke, R. *High Contagiousness and Rapid Spread of Severe Acute Respiratory Syndrome Coronavirus 2.* Emerging Infectious Diseases. 26(7):1470-1477 (2020)

[37] https://www.idmod.org/docs/emod/hiv/index.html / *Compartmental models and EMOD » SI and SIS models*

[38] Luciano da F. Costa; Francisco A. Rodrigues; Alexandre S. Cristino *Complex networks: the key to systems biology* Genet. Mol. Biol. (2008)

[39] Prem K, Cook AR, Jit M *Projecting social contact matrices in 152 countries using contact surveys and demographic data.* PLoS Comput Biol 13(9): e1005697. (2017)