Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet

Dorian Stipić

Analiza scenarija COVID-19 pandemije simulacijama na grafu

Zagreb, 2020.

Ovaj rad izrađen je na Matematičkom Odsjeku u sklopu istraživačkog projekta koji se bavi suzbijanjem pandemije COVID-19 pod vodstvom mentora prof. dr. sc. Borisa Podobnika i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2019./2020.

Sadržaj

1 Uvod **1**

2 Standardni epidemiološki modeli **3**

2.1 Osnovni SIR model 3

2.2 Varijante SIR modela 5

2.3 Epidemiološki modeli pomoću mreža 6

3 Detaljan opis modela **9**

3.1 Struktura mreže 10

3.2 Stanja i prijelazi 10

3.3 Dinamika 12

3.4 Parametri modela 14

4 Simulacije i rezultati **17**

4.1 Utvrđivanje neslobodnih parametara 17

4.2 Ekstrakcija značajki 18

4.3 Prikaz značajki osnovnog modela 19

4.4 Ekstenzije 24

4.5 Vremenski nizovi 32

5 Zaključak **37**

A Implementacija najbitnijih dijelova projekta **38**

Zahvale **49**

Bibliografija **52**

Sažetak **53**

Summary **54**

Životopis **55**

Poglavlje 1

Uvod

Pandemija COVID-19 bolesti izazvana novim koronavirusom SARS-CoV-2 koja je pogodila svijet krajem 2019. godine je brzo ušla u kolektivnu memoriju kao jedan od najznačajnih događaja 21. stoljeća.

Mnogi stručnjaci su dugi niz godina upozoravali javnost oko povećanog rizika destruktivnih pandemija[1] u današnjem, sve više globaliziranom svijetu. Nažalost takva upozorenja su se ostvarila i trenutno se nalazimo u možda potencijalno najkatastrofalnijoj pandemiji u zadnjih 100 godina [25].

Cilj ovog rada je upravo analizirati i simulirati efekte i posljedice koje COVID-19 pandemija može imati na svijet i ljude, te probati odgovoriti na pitanja efikasnosti različitih mjera u sprječavanju daljnjeg širenja pandemije, odnosno virusa po populaciji. Fokus rada je isključivo epidemiološkog tipa odnosno traženje onih mjera i strategija civilnog ponašanja koje će omogućiti ispravno funkcioniranje zdravstvenog sustava s ciljem da u konačnici broj smrtnih slučajeva (direktno ili indirektno prouzrokovanih COVID-19) bude što manji.

Svjesni smo da je u svakoj analizi ovog tipa *ekonomija* možda najbitnija stavka no obzirom na kompleksnost problema takva analiza je ostavljena za potencijalne kasnije projekte. Bez obzira na to sigurni smo da se ova analiza, kao i zaključci, mogu lako uklopiti u širi ekonomski okvir na prirodan način i bez većih poteškoća.

Valja podsjetiti kako su ljude kroz povijest pratile manje ili više destruktivne pandemije: od Atenske i Rimske kuge (430. pr. n.e. i 165. n.e.) sve do Španjolske gripe (1918.–1920.) i HIV/AIDS pandemije (1959. do danas), stoga napominjemo kako su uz

[1] Pandemija: širenje neke bolesti na velika prostranstva, tj. na više država, cijeli kontinent ili cijeli svijet. Destruktivne zovemo one pandemije koje su usmrtile (u relativno kratkom periodu) značajan dio populacije (barem 0.1%) pogođenog teritorija ili, ako taj podatak nije dostupan, one pandemije o kojima postoje jasni povijesni zapisi (dakle koje su ostavile trag u kolektivnoj ljudskoj memoriji).

malu modifikaciju modelskih parametara (vremena inkubacije, smrtnost, itd.) sve analize i zaključci navedeni u ovom radu za COVID-19 jednako primjenjivi i za prošle (kao i nažalost buduće) pandemije.

Poglavlje 2

Standardni epidemiološki modeli

Predloženih i korištenih epidemioloških modela u literaturi ima jako puno [12], u ovom poglavlju ćemo se ograničiti samo na povijesno najznačajnije i najutjecajnije modele, kao što su SIR model i njegove varijante.

Epidemiološki modeli se dijele na modele stohastičke i determinističke prirode [21]. *Stohastički* modeli se baziraju na slučajnim varijablama i simuliraju širenje pandemija pomoću slučajnih događaja. *Deterministički* modeli nemaju slučajnu komponentu, već se osnovno širenje modelira pomoću diferencijalnih jednadžbi. Obično se populacija jedinki podijeli na nekoliko disjunktnih skupova, gdje ti skupovi predstavljaju različite stadije bolesti (ovaj pristup se može naravno primijeniti i u stohastičkim modelima).

2.1 Osnovni SIR model

Osnovni matematički model za epidemijske bolesti postavili su 1927. Kermack i McKendrick [1]. Njihov model primjenjiv je za epidemijske bolesti relativno kratkog vijeka (u usporedbi sa životnim vijekom) koje se iznenada pojave te zaraze značajan dio populacije. U osnovnom modelu populacija se dijeli u tri disjunktna skupa:

- skup **S**, neinficirane jedinke koje su podložne zarazi.
- skup **I**, jedinke zaražene promatranim mikroorganizmom.
- skup **R**, jedinke koje su se oporavile od infekcije te stekle trajni imunitet.

Širenje bolesti se uobičajeno prikazuje preko sljedećeg kompartmentalnog dijagrama, jedini mogući prijelazi su dani strelicama, te je njihov intenzitet kontroliran preko konstanti β i γ.

Slika 2.1: Kompartmentalni dijagram osnovnog SIR modela.

Osnovni model je izveden uz tri glavne pretpostavke:

1. Populacija je zatvorena tj. nema rođenja, smrti i migracija.
2. Bolest se prenosi kontaktom podložnih zarazi i inficiranih jedinki.
3. Prostorna homogenost populacije (svi kontakti među jedinkama su jednako vjerojatni).

Model predstavljamo s tri obične diferencijalne jednadžbe

$$\frac{dS}{dt} = -\frac{\beta IS}{N} \tag{2.1}$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I \tag{2.2}$$

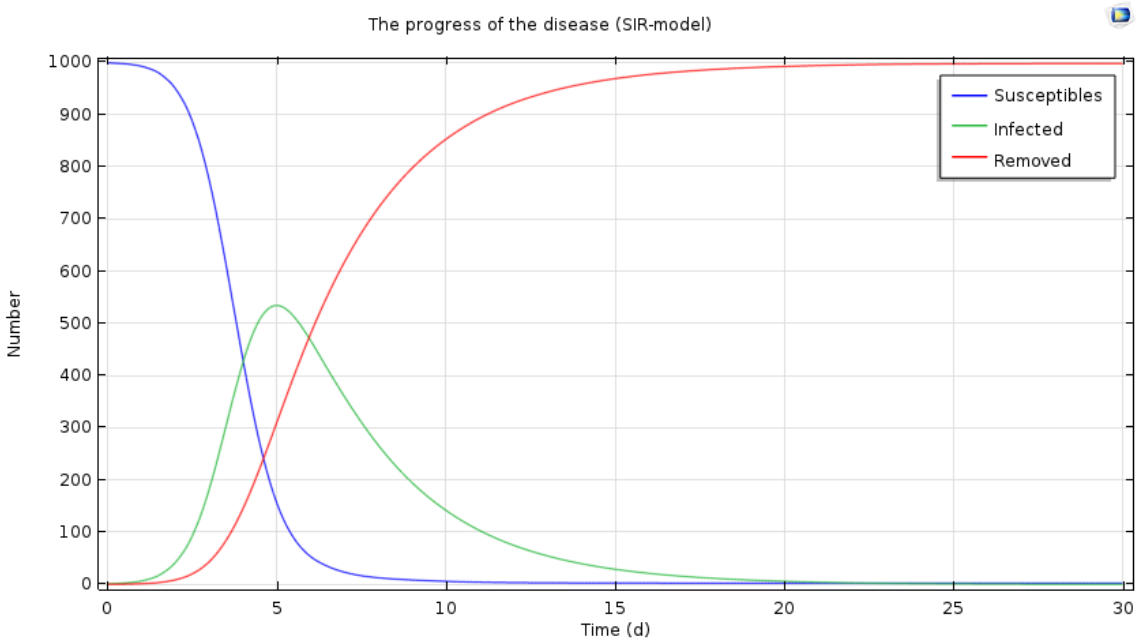
$$\frac{dR}{dt} = \gamma I \tag{2.3}$$

gdje su S, I i R dani kao funkcije koje ovise o vremenu t. Iz $\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt} = 0$ slijedi da je ukupna populacija $S + I + R = N$ konstantna. Izraz IS je ukupan broj kontakata podložnih i inficiranih jedinki i direktno je proporcionalno broju novozaraženih. β je parametar koji kontrolira intenzitet tog prijelaza. Slično broj oporavljenih direktno je proporcionalan broju inficiranih I uz parametar γ.

Standardno se simulira situacija u kojoj je u početku zaraženo samo nekoliko jedinki:

$$S(0) \approx N, \quad I(0) \approx 0, \quad R(0) = 0. \tag{2.4}$$

Rješavanjem gornjih diferencijalnih jednadžbi uz dane početne uvjete dobiju se grafovi funkcija $S(t), I(t)$ i $R(t)$ kao na slici 2.2 (grafovi naravno ovise o odabranim parametrima modela β i γ).



Slika 2.2: Primjer širenja bolesti kroz vrijeme, opisane SIR modelom.

Postoji jako puno suptilnih teorijskih tvrdnji vezanih za SIR model koje nećemo navesti jer ne spadaju u glavni fokus ovog rada.

Spomenimo međutim jako bitnu konstantu koju vežemo za svaki patogenski mikroorganizam a to je njegov **bazični reprodukcijski broj** R_0 koji se definira kao *očekivani* broj jedinki koje će jedna zaražena osoba zaraziti tijekom svog infektivnog perioda, pod uvjetom da su sve druge jedinke podložne (početak epidemije). Ako je $R_0 > 1$ tada je bolest u stanju širiti se po populaciji.

Uz R_0 promatra se i efektivni reprodukcijski broj R_e[1]. Svi modeli na različite načine i preko različitih metoda računaju i aproksimiraju te brojeve[2] koji su fundamentalni za bilo kakvu predikciju odnosno razumijevanje brzine širenja epidemije.

2.2 Varijante SIR modela

Postoji mnogo varijanti SIR modela u kojima su uvedene nešto *drugačije pretpostavke* od triju navedenih za osnovni SIR model te uvođenje novih skupova (kompartmenta).

[1]Reprodukcijski broj promatran kao vremenski niz, odnosno *očekivani* broj jedinki koje će jedna zaražena osoba zaraziti u danom trenutku t. $R_e(t)$ je nerastuća funkcija i za $t = 0$ po definiciji vrijedi $R_e(0) = R_0$.

[2]U SIR modelu je npr. $R_0 = \frac{\beta}{\gamma}$

Jedna od najčešćih ekstenzija je dodavanje takvih kompartmenta za umrle (ako promatrana bolest nije bezopasna) kao i novorođene jedinke, tzv. modeli sa životnim dinamikama.

Bolesti za koje nakon ozdravljenja ne slijedi imunitet (npr. većina herpes virusa) modeliraju se preko SIS modela, obične diferencijalne jednadžbe koje opisuju taj model su očite (prve dvije jednadžbe SIR modela bez γI dijela u drugoj).

Slično za bolesti u kojima se imunitet stekne, ali ne doživotno, govorimo o SIRS modelu.

Za bolesti poput AIDS-a u kojima je period inkubacije bolesti značajno dugačak koristi se SEIR model u kojem između kompartmenta S i I postoji novi kompartment E (eng. exposed), diferencijalne jednadžbe u pitanju su jako slične kao i za SIR model.

Korištene varijante su također SEIS (inkubacija plus manjak imuniteta) i SEIRS (inkubacija plus imunitet koji nije doživotni).

Mnogo je još ekstenzija SIR modela koje smo izostavili jer nisu relevantne za kasniju diskusiju.

2.3 Epidemiološki modeli pomoću mreža

Jedan od najprirodnijih načina modeliranja infektivnih bolesti je korištenje mreže[3] gdje se svaka jedinka može prirodno poistovjetiti s jednim vrhom u mreži a svaki kontakt među jedinkama (preko kojeg može doći do prijenosa infekcije) jednim bridom. Ovakav pristup je sve češći u literaturi zbog sve većeg razvoja računalne snage koja omogućava vremenski prihvatljive simulacije za mreže čiji je red veličine usporediv s populacijom jedne države[4].

Prvi epidemiološki modeli koristili su slučajne mreže u kojima se na početku generiraju vrhovi te se u svakoj iteraciji stvaraju novi slučajni bridovi na način da je prosječan stupanj vrhova konstantan[5]. Međutim, u stvarnosti većina jedinki u populaciji ima određeni relativno mali broj fiksnih kontakata preko kojih se zaraza može prenijeti, zbog kojih su se u zadnje vrijeme počele razvijati mreže s takvim svojstvom, vidi [7].

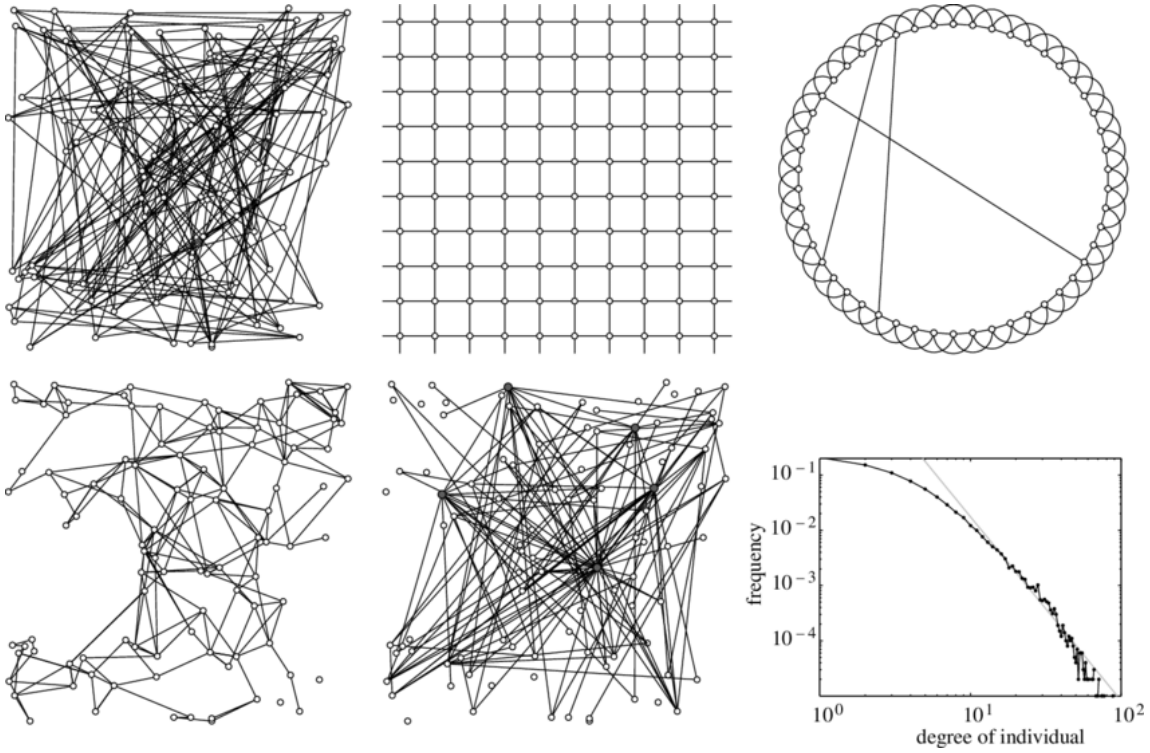
Na slici 2.3 vidimo primjer mreža preko kojih se u raznim istraživačkim radovima pokušalo simulirati stvarne ljudske relacije.

Prva slika, gore lijevo, prikazuje primjer slučajne mreže čija su se (loša) svojstva već navela.

[3] U ovom kontekstu mreža zovemo neusmjereni graf $G(V, E, \psi)$ gdje je V skup vrhova, E skup bridova a ψ funkcije incidencije.

[4] 10^6-10^8 vrhova.

[5] Stupanj vrha v u grafu G je broj bridova koji su incidentni s v.



Slika 2.3: Pet različitih mreža sa 100 vrhova čiji je prosječni stupanj 4. Gornji redak, s lijeva na desno: slučajna, *rešetkasta* i *small-world*. Donji redak: *prostorna* i *scale-free*. Donja desna slika predstavlja distribuciju stupnjeva vrhova *scale-free* mreže (generirane preko 1000 simulacija), uočimo kako distribucija slijedi zakon potencija. Izvor: [7].

Druga slika, gore u sredini, prikazuje primjer *rešetkaste* mreže, takve mreže su se povijesno koristile u tzv. *forest-fire* modelima koji simuliraju i opisuju širenje požara po šumi [2], međutim kasnije analize su pokazale da tipično valovito širenje infekcije po takvim mrežama jako dobro aproksimira stvarnu situaciju u prostorno širokim područjima [5]. Osnovno svojstvo takvih mreža je to što većinu vrhova spajaju jako dugi (najkraći) putovi[6], ipak pokazalo se da stvarne ljudske socijalne mreže i poznanstva nemaju to svojstvo nego da prate tzv. *small-world* pravilo odnosno da većinu ljudi veže lanac od najviše 6 poznanstva [23].

Treća slika, gore desno, prikazuje primjer *small-world* mreže, dobivena iz obične *rešetke* u kojoj su nasumično povezani "daleki" vrhovi, pokazalo se da dodavanje malog broja bridova može značajno promijeniti cijelu dinamiku širenja infekcije i u suštini učiniti mrežu podložnijom na pandemije.

Četvrta slika, dolje lijevo, prikazuje primjer *prostorne* mreže, u takvoj mreži su vrhovi

[6]Put je niz vrhova i njima incidentnih bridova u kojima su svi vrhovi osim eventualno prvog i zadnjeg različiti.

pozicionirani na nekoj površini (ili volumenu) a dva vrha su povezana bridom preko neke vjerojatnosne distribucije koja ovisi o njihovim udaljenostima, takve mreže su generalizacija prethodnih, dakle slučajne, *rešetkaste* i *small-world* mreže spadaju u ovu kategoriju.

Peta slika, dolje u sredini, prikazuje primjer *scale-free* mreže čija je glavna karakteristika ta da distribucija stupnjeva X njihovih vrhova slijedi zakon potencija, formalno

$$P(X > x) \sim L(x)x^{-(\alpha-1)} \text{ za } \alpha > 1 \tag{2.5}$$

gdje je $L(x)$ slabo varirajuća funkcija $\lim_{x\to\infty} L(rx)/L(x) = 1$.

Intuitivno, distribucija stupnjeva vrhova u mreži nije homogena, nego postoji mali broj *super-povezanih* vrhova dok većina ima relativno mali broj susjeda. Takve mreže uočene su u autorskim suradnjama [4] i ljudskim seksualnim kontaktima [6]. Šesta slika prikazuje zakon potencija za $\alpha = 4.3$ dobivene preko numeričkih simulacija *scale-free* mreže.

Poglavlje 3

Detaljan opis modela

U prethodnom poglavlju je opisano nekoliko različitih pristupa u modeliranju širenja infektivnih bolesti po populaciji, također smo utvrdili da su povijesni modeli koristili obične diferencijalne jednadžbe za opis dinamike, čiji pristup ima prednost u tome što je jednostavan, a istovremeno dovoljno dobar u uvjetima prostorne homogenosti populacije. Ipak, puno prirodniji pristup je modeliranje infektivnih bolesti preko mreže jer se samo tako mogu najbolje simulirati sve suptilne dinamike kao i lokalno širenje (epidemije počinju iz jednog ili više žarišta). Na kraju prethodnog poglavlja spomenuli smo nekoliko poznatih mreža koje dobro aproksimiraju ljudsko društvo i međusobne kontakte jedinki. Poznato je međutim da jako mala razlika u strukturi dviju mreža može dovesti do puno drugačijeg širenja infekcije po tim mrežama, što znači da u slučaju prediktivnih i analitičkih potreba treba ”jako precizno” generirati mrežu koja je skoro savršena preslika stvarnosti; takav pothvat je nažalost vremenski zahtjevan i gotovo nemoguć [7, str.297].

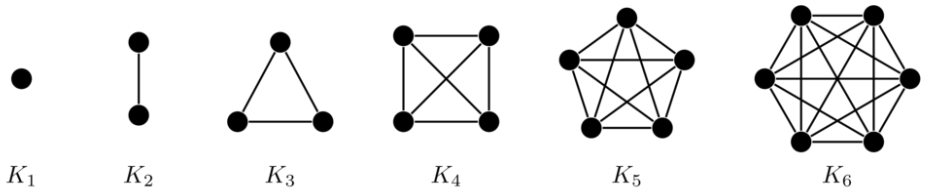
Model za koji smo se odlučili zaobilazi gornji problem: bazira se na jednostavnoj i općenitoj mreži čija struktura sadržava kućanstva[1] koja su najrelevantniji dio opserviran u empiriji ([3] i specifično za COVID-19 [17]) po kojem se infekcija širi. Širenje infekcije po ostatku grafa je riješeno na elegantan način koji posjeduje poželjno svojstvo *small-worlda* a svi prijelazi stanja (zaraza, ozdravljenje, itd.) su tretirani kao stohastički.

Razvijeni model, kao što ćemo vidjeti, omogućava dovoljnu fleksibilnost za uvođenje logičko/semantičkih smislenih restrikcija na ”kretanje” jedinki po mreži koje mjerodavno simuliraju stvarne socijalne i ekonomske mjere implementirane za suzbijanje širenja infekcije. Takva fleksibilnost u modelu je ključna jer su uvedene mjere na ovakvoj skali povijesno jedinstvene za COVID-19 pandemiju.

[1] i eventualni uski broj ljudi susjeda i/ili prijatetlja s kojima je jedinka svakodnevno u kontaktu.

3.1 Struktura mreže

Implementirana mreža je **jednostavna** i sastoji se od ukupno N vrhova (jedinki)[2] koji su posloženi u potpune[3] podgrafove veličine K. Slika 3.1 geometrijski prikazuje nekoliko potpunih grafova s malim brojem vrhova. Potpune podgrafove u mreži zovemo **grozdovi**, gdje jedan grozd predstavlja jedno kućanstvo (ili skup ljudi koji su *svakodnevno* u kontaktu). Grozdovi **nisu** međusobno povezani bridovima, nego je njihova "komunikacija" riješena na drugačiji način (vidi potpoglavlje 3.3).



Slika 3.1: Potpuni grafovi.

3.2 Stanja i prijelazi

Sama struktura mreže je prilično jednostavna[4], međutim, moguća stanja (kompartmenti) u kojima se vrhovi mreže mogu nalaziti kao i sama dinamika prilično su kompliciranі.

Stanja u kojima se jedinke mogu naći su: *susceptible*, *infected*, *mild*, *recovered*, *icu covid19*, *icu nocovid19*, *dead covid19*, *dead nocovid19*. Detaljnije:

- ***susceptible*** : jedinka koja je podložna zarazi COVID-19[5].
- ***infected*** : jedinka koja se zarazila s COVID-19, ali još ne pokazuje simptome.
- ***mild*** : jedinka koja se zarazila s COVID-19 i pokazuje simptome[6]
- ***recovered*** : jedinka koja je ozdravila od COVID-19 i stekla imunitet[7].

[2]U grafu vrh predstavlja jednu jedinku populacije. Kroz rad koristit će se obje riječi ovisno o kontekstu.

[3]Graf u kojem je svaki par različitih vrhova povezan, potpun graf od K vrhova ima $\binom{K}{2}$ bridova.

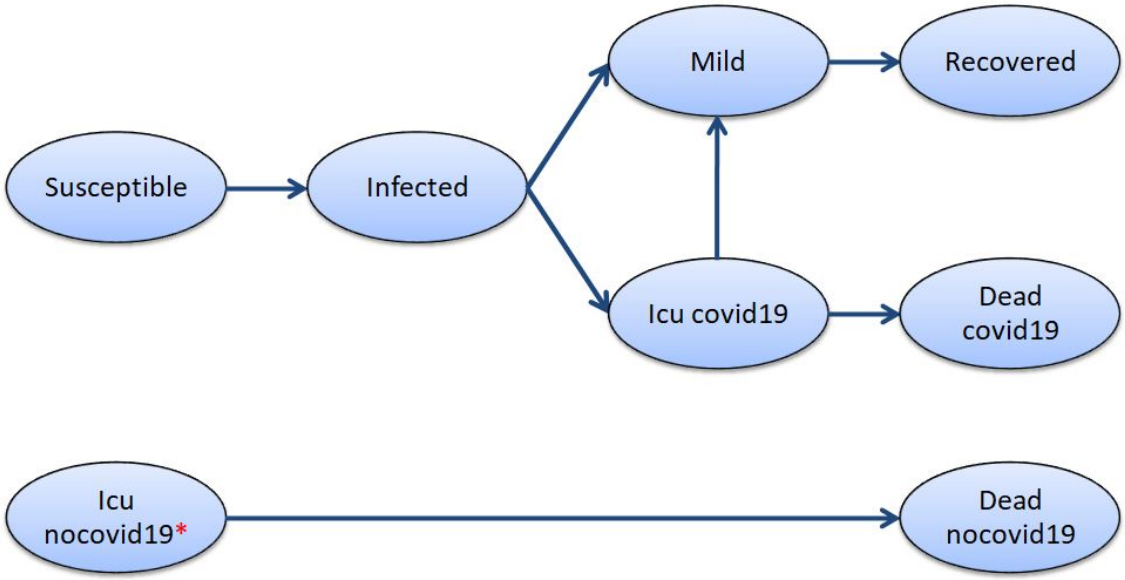
[4]Mreža iako jednostavna istovremeno omogućava kompleksnu dinamiku i mjerodavnu simulaciju stvarnog svijeta.

[5]COVID-19 je bolest prouzrokovana SARS-CoV-2 virusom. Dalje u tekstu, zbog jednostavnosti, govorimo o COVID-19 zarazi iako nije skroz precizno.

[6]U modelu, zbog jednostavnosti, svi zaraženi nakon određenog vremena pokazuju simptome, postoje sumnje o asimptomatskim slučajevima međutim još uvijek su kontradiktorni podaci o postotku takvih slučajeva kao i o njihovoj zaraznosti.

[7]Imunitet može biti doživotni ili kratkotrajni, više u poglavlju 4.

- ***icu covid19*** : jedinka koja se zarazila s težim oblikom bolesti COVID-19 i zauzela 1 respirator.
- ***dead covid19*** : jedinka koja je umrla zbog COVID-19 bolesti.
- ***icu nocovid19*** : jedinka koja je oboljela od nespecificirane bolesti i zauzela 1 respirator.
- ***dead nocovid19*** : jedinka koja je umrla zbog nespecificirane bolesti.



Slika 3.2: Shema stanja i prijelaza.

Slika 3.2 prikazuje stanja i njihove međusobne prijelaze. Gornji dio grafa odnosi se na dinamiku relevantnu za COVID-19. Vidimo da samo jedinke na respiratoru (*icu covid19*) mogu imati smrtonosan ishod, što možemo pretpostaviti bez smanjenja općenitosti[8]. Prijelaz iz stanja *icu covid19* u stanje *mild* je uveden zbog logike i opservacije stvarnog tijeka bolesti (bolesnici koji dobiju teži oblik bolesti, ako prežive, pređu u lakši oblik bolesti prije potpunog ozdravljenja). Također, *infected* stanje predstavlja trenutak kad je jedinka već zaražena (i zarazna), ali još ne pokazuje simptome što se ne smije miješati s tzv. asimptomatskim bolesnicima koji u modelu pripadaju *mild* stanju zbog jednostavnosti.

[8]Iako u stvarnosti neki bolesnici umru *prije* nego što dođu na respirator, u modelu vjerojatnost umiranja na respiratoru obuhvaća i vjerojatnost umiranja bez respiratora.

Donji dio grafa odnosi se na ostale nespecificirane bolesti (koje zahtijevaju respirator za uspješno liječenje). Zvjezdica s kojom je označeno stanje *icu nocovid19* je kratica za nekoliko prijelaza: sva stanja osim *icu covid19*, *dead covid19* i *dead nocovid19* mogu prijeći u to stanje dok jedinke u tom stanju mogu prijeći u stanja *dead nocovid19*, *susceptible* ili *recovered* ovisno o situaciji (više o tome kasnije, potpoglavlje 3.3).

U modelu nisu uzeti u obzir procesi rođenja ili migracija jer je broj promatranih iteracija dovoljno malen što ih čini zanemarivim (vidi poglavlje 4).

Svi prijelazi između stanja kao i vremena provedena u svakom stanju su empirijski utvrđeni (podaci iz medicinskih i znanstvenih članaka).

3.3 Dinamika

U ovom potpoglavlju opisat ćemo dinamiku modela implementiranu u računalnom programu, te navesti relevantne varijable i parametre.

Dinamika je **diskretna i sekvencijalna**, odnosno u svakoj iteraciji (ekvivalentnoj jednom danu) ažuriraju se stanja svih vrhova u mreži na sekvencijalan način (od prvog do zadnjeg po redoslijedu koji je fiksiran na početku).

Prije početka simulacije slijedi **inicijalizacija** mreže i stanja: korisnik definira prijelazne vjerojatnosti svih stanja, vremena zadržavanja u svakom, ukupni broj vrhova, broj respiratora, veličinu grozdova[9], broj iteracija, itd. Također moguće je definirati proizvoljni broj kategorija (tipova) jedinki i njihove omjere u mreži (npr. po dobnim skupinama, čija razlika može biti drugačija vjerojatnost preživljavanja bolesti), vidi potpoglavlja 3.4 i 4.2 za više detalja.

Nakon inicijalizacije slijedi simulacija; u nultoj iteraciji sve su jedinke u stanju *susceptible*, ali postoji vjerojatnost (parametar koji postavlja korisnik) s kojom će svaka jedinka, u svakoj iteraciji, *interno podbaciti*[10] i prijeći u stanje *infected*. Ovaj proces simulira uvedene slučajeve odnosno bilježi početak pandemije. S pojavom prvih *infected* jedinki pandemija se počinje širiti među populacijom na način da u svakoj iteraciji, svaka bolesna jedinka, ima vjerojatnost r zaraziti svakog susjeda unutar grozda. Jedinke koje su zaražene na ovaj način kažemo da su *eksterno podbacile*.

Nakon što su se ažurirala stanja unutar grozdova slijedi ključan proces koji kontrolira **zarazu između različitih grozdova**: u svakoj iteraciji svaka jedinka ima vjerojatnost p_1 da *ođe na putovanje* (na posao, u školu, u dućan itd.) i na taj način riskira da ju zarazi

[9] Zbog jednostavnosti veličina grozdova u mreži je homogena.

[10] Interni podbačaj eng. internal failure znači da se jedinka zarazi interno neovisno o okolišu.

neka druga bolesna jedinka na putovanju. Vjerojatnost zaraze za *susceptible* jedinku na putovanju je dana formulom

$$\hat{r} = k \cdot \frac{r \cdot b}{n}, \tag{3.1}$$

gdje je b ukupan broj **bolesnih** jedinki na putovanju, n broj **svih** jedinki na putovanju, a k običan parametar.

Dodatni slučaj implementiran u modelu je ponašanje jedinki s barem jednim *mild* ili icu-covid19 susjedom[11]: takve jedinke, s obzirom da imaju susjeda koji je bolestan i pokazuje simptome vrše oblik samoizolacije i odlaze na putovanje (puno rjeđe) s vjerojatnosti $p_1 \cdot p_2$ gdje je p_2 vjerojatnost (ne)poštivanja preporučenog ponašanja[12]. Za jedinke koje su u stanjima *mild* ili *icu covid19* pretpostavlja se da **ne idu** (vjerojatnost nula) na putovanje (prvi jer su bolesni i dovoljno savjesni da ne šire takvu bolest a drugi jer su doslovno u bolnici na respiratoru). U potpoglavlju 4.4 analizirat će se situacije gdje ova prepostavka ne vrijedi. Dodatna pretpostavka u modelu za ranije spomenute jedinke je da gube veze (bridove) koji ih vežu za svoje susjede unutar grozda dakle ne šire zarazu ukućanima (prvi se uspješno samoizoliraju a drugi su u bolnici na respiratoru)[13].

Prijelazi u stanje *icu nocovid19*, vidi sliku 3.2, implementirani su kao *interni podbačaj* dakle u svakoj iteraciji proizvoljna jedinka ima neku vjerojatnost da se teško razboli i zauzme jedan slobodni respirator[14] (slučajne nesreće, druge infektivne bolesti, upale pluća itd. su tretirane kao interni podbačaj zbog jednostavnosti). *susceptible* jedinka koja prijeđe u ovo stanje i preživi vraća se u *susceptible* stanje dok se *infected*, *mild* i *recovered* jedinke sve vraćaju u *recovered* stanje nakon ozdravljenja (u ovo spada slučaj gdje npr. osoba dođe u bolnicu zbog infartka i onda se otkrije da je COVID-19 pozitivna)

Potrebno je napomenuti još jedan implementacijski detalj vezan za respiratore: jedinke u stanjima *icu covid19* i *icu nocovid19* idu "u bolnicu" i zauzmu respirator, međutim postojanje bolnice je *implicitno*; u modelu postoji samo ukupni broj respiratora R i svaka jedinka u tim stanjima zauzima jedan. Trenutak u kojem broj slobodnih respiratora padne na nulu i umre prva jedinka kojoj treba slobodni respirator, a takvih nema zovemo **kolaps zdravstvenog sustava** (ključno u simulacijama, poglavlje 4).

[11] Podsjećamo da se svi susjedi jedne jedinke nalaze samo unutar njenog grozda.

[12] p_2 djeluje na p_1 kao nezavisan događaj te što je p_2 bliže 1 to je ponašanje jedinke s bolesnikom i bez bolesnika u kučanstvu sličnije.

[13] Pretpostavka za *mild* slučajeve se možda ne čini potpuno opravdanom, međutim, provedene simulacije su pokazale da čak s tom pretpostavkom se više od 90% jedinki u grozdu razboli nakon prvog *infected* slučaja što odgovara stvarnosti.

[14] Zbog relevantnosti u modelu se promatraju samo one bolesti koje zahtijevaju korištenje respiratora jer je to u direktnoj koliziji s potrebama liječenja teških COVID-19 bolesnika, novija verzija modela mogla bi uključiti *sve* bolesti.

Za kraj spomenimo vjerojatnost prijelaza u stanja *dead covid19* i *dead nocovid19* zbog **inovativnosti**. Vjerojatnosti prijelaza iz stanja *icu covid19* u *dead covid19* $\hat{p}_c$ i iz stanja *icu nocovid19* u *dead nocovid19* $\hat{p}_{nc}$ su *promjenjive* i dane formulama

$$\hat{p}_c = 1 - (1 - p_c) \cdot \exp\left(-\mu \cdot \frac{m + ic}{n'}\right) \tag{3.2}$$

$$\hat{p}_{nc} = 1 - (1 - p_{nc}) \cdot \exp\left(-\mu \cdot \frac{m + ic}{n'}\right) \tag{3.3}$$

gdje je m ukupan broj jedinki u stanju *mild*, *ic* ukupan broj jedinki u stanjima *icu covid19* i *icu nocovid19*, n' ukupan broj živih jedinki, μ je sloban parametar koji kontrolira opterećenje zdravstvenog sustava, a p_c i p_{nc} su prijelazne vjerojatnosti u nultoj iteraciji (prije početka pandemije) koje definira korisnik.

Vjerojatnosti umiranja $\hat{p}_c$ i $\hat{p}_{nc}$ su vremenski nizovi (promjenjivi u svakoj iteraciji) koji ovise o **opterećenju** zdravstvenog sustava (kvaliteta njege dana pojedinačnoj jedinki nužno opada s povećanjem broja bolesnika). Razlomak iz formula (3.2) i (3.3) je zapravo udio bolesnika (na kućnoj ili bolničkoj njezi) u (živoj) populaciji. Ukoliko su svi respiratori zauzeti sljedeća jedinka, kojoj je potreban respirator, umire istog trena s vjerojatnosti 1.

3.4 Parametri modela

Na slici 3.3 vidimo sučelje preko kojeg korisnik (ili neki drugi program) fiksira parametre modela. Slika prikazuje prave vrijednosti parametara korištene u simulacijama, detalji u potpoglavlju 4.2. Sekcija *graph_generation* odnosi se na dimenzije i strukturu grafa, fiksira se broj grozdova, broj jedinki u svakom grozdu K, kategorije jedinki i njihove omjere ([72 28] znači da postoje dvije kategorije jedinki, 72% jedinki spada u prvu a 28% u drugu kategoriju). Sekcija *simulation* brine se o simulaciji: *stopping_conditions* su uvjeti zaustavljanja programa (maksimalni broj iteracija ili ako je zdravstveni sustav u kolapsu); zatim slijede parametri: *num_icus* je broj respiratora R, *mu* je parametar μ, *prob_transmission* je parametar eksterne zaraze r, *k_trip* je parametar k vezan za putovanja te *isolate_cluster_on_known_case*, ako je *true*, aktivira opisani proces samoizolacije jedinki sa *mild* ili icu-covid19 susjedom.

Podsekcija *initial_params* fiksira sve relevantne parametre za stanja i prijelaze (vjerojatnosti prijelaza i vremena zadržavanja) za sve generirane kategorije (u ovom slučaju dvije). Istaknimo samo da je *prob_goes_on_trip* vjerojatnost odlazaka na putovanje p_1 i *prob_m_neighbour_trip candidate* vjerojatnost (ne)poštivanja mjera samoizolacije p_2.

```
{
  "graph_generation": [
    {
      "num_clusters": 200000,
      "num_people_per_cluster": 5,
      "category_ratios": [72, 28]
    }
  ],
  "simulation": {
    "stopping_conditions": {
      "num_days": 1200,
      "on_icu_overflow": false
    },
    "num_icus": 200,
    "mu": 5,
    "prob_transmission": 0.1,
    "k_trip": 2.5,
    "isolate_cluster_on_known_case": true,
    "initial_params": [
      {
        "prob_goes_on_trip": 0.3,
        "prob_m_trip_candidate": 0,
        "prob_m_neighbour_trip_candidate": 0.9,
        "prob_s_to_i": 0,
        "days_i_to_sick": 5,
        "prob_i_to_ic": 0.00471,
        "days_m_to_r": 11,
        "days_ic_to_r_or_m": 10,
        "prob_ic_to_d": 0.286,
        "prob_to_nic": 0.00000618,
        "prob_nic_to_d": 0.36,
        "days_nic": 5
      },
      {
        "prob_goes_on_trip": 0.3,
        "prob_m_trip_candidate": 0,
        "prob_m_neighbour_trip_candidate": 0.9,
        "prob_s_to_i": 0,
        "days_i_to_sick": 5,
        "prob_i_to_ic": 0.11285,
        "days_m_to_r": 11,
        "days_ic_to_r_or_m": 10,
        "prob_ic_to_d": 0.286,
        "prob_to_nic": 0.00000618,
        "prob_nic_to_d": 0.36,
        "days_nic": 5
      }
    ],
    "events": [
      {
        "label": "pocela korona",
        "day": 15,
        "update_params": {
```

```
            "prob_s_to_i": [0.000001, 0.000001]
          }
        },
        {
          "label": "kraj uvez. korona",
          "day": 45,
          "update_params": {
            "prob_s_to_i": [0, 0]
          }
        }
      ]
    }
}
```

Slika 3.3: Parametri modela.

Podsekcija *events* je metoda koja omogućava da se vrijednost bilo kojeg parametra promijeni u nekoj danoj iteraciji (npr. *day* : 15 znači da se uvedena promjena odnosi na sve iteracije nakon 15.).

Poglavlje 4

Simulacije i rezultati

U ovom poglavlju opisujemo provedene simulacije i grafički ih demonstriramo. U prvom potpoglavlju bavimo se utvrđivanjem parametara citirajući relevantne izvore, opisujemo provedene simulacije i relevantne promatrane značajke, a zadnja potpoglavlja prezentiraju dobivene rezultate.

Korišteni su programski jezici: C++ (implementacija modela) i Python (prikaz svih rezultata), vidi dodatak A.

4.1 Utvrđivanje neslobodnih parametara

Na slici 3.3 vidimo da model zahtijeva veliku količinu parametara. Neki od tih parametara su slobodni i zahtijevaju prostorno pretraživanje, dok se ostali mogu dobiti iz relevantnih medicinskih izvora kao i iz znanstvenih članaka za COVID-19 bolest[1].

Slobodni parametri su: vjerojatnost odlaska na putovanje p_1, vjerojatnost (ne) poštivanja samoizolacije p_2, veličina grozdova K i parametar vezan za smrtnost μ.

Parametar eksterne infekcije r i parametar vezan za putovanja k se ne smatraju slobodnima: r je fiksiran na način da se, u slučaju nepoduzimanja ikakvih mjera, preko 90% jedinki u grozdu, čiji je barem jedan član zaražen, razboli; vrijednost je fiksirana na 0.1 što je kasnije potvrdila i studija [15] [2]. Parametar k je fiksiran pomoću simulacija: provedeno je 100.000 simulacija za sve razumne veličine grozdova (2-10) i numerički je izračunat reprodukcijski broj R_0, odabrana je vrijednost $k = 2.5$ čiji se R_0 najbolje slaže s raznim

[1] Napomenimo da je u zadnjih nekoliko mjeseci objavljen niz članaka vezanih za COVID-19 s ponekad kontradiktornim informacijama i zaključcima, stoga sam proces traženja *pravih* parametara može biti dug posao koji gotovo uvijek ostavlja prostora dodatnom poboljšanju. Također neki parametri (”svojstva” virusa) jednostavno su još nepoznati jer se radi o novoj bolesti.

[2] Studija zaključuje da je vjerojatnost zaraze 3%-13% za bliske kontakte.

izvorima iz [20] [3].

Ostali parametri vezani za COVID-19 bolest kao i parametri vezani za nespecificirane respiratorne bolesti dobiveni su iz pouzdanih medicinskih izvora: [24], [22], [13], [9], [8], [14]. Smrtnost COVID-19 bolesti u modelu za generalnu populaciju je fiksirana na 1% a vjerojatnost obolijevanja od težeg oblika bolesti koji zahtijeva respirator na 3.5%[4]. Broj dana inkubacije (prije pojave simptoma) je fiksiran na 5, broj dana intenzivnog liječenja (respirator) na 10, a broj dana do ozdravljenja od blagog oblika bolesti na 11 (dakle, za bolesnike na respiratoru od prvih simptoma do potpunog ozdravljenja prođe ukupno $10+11=21$ dan). Na sličan način su fiksirani parametri za nedefinirane bolesti: 5 dana intenzivnog liječenja (na respiratoru) i vjerojatnost umiranja od 36% uz najbolju moguću njegu. Šansa obolijevanja je fiksirana na način da su bolnički kapaciteti (u prosjeku) na 20%[5], odnosno da u slučaju pandemije 80% respiratora može biti alocirano za COVID-19 bolesnike.

4.2 Ekstrakcija značajki

Provedeno je nekoliko simulacija za sve smislene kombinacije slobodnih parametara: vjerojatnosti p_1 i p_2 unutar segmenta $[0,1]$, veličina grozda K za sve vrijednosti u skupu $S=\{2,3,...,10\}$, a parametar μ je fiksiran na 5 osim ako nije drugačije navedeno.

Broj jedinki u svim simulacijama je $N=10^6$ (u okvirima komputacijske moći ali i dovoljno velik za relevantne zaključke). Omjeri kategorija jedinki i broj respiratora R prilagođeni su za Hrvatsku; jedinke su podijeljene u dvije skupine: mlađi i stariji ($\geq$) od 60 godina. U Hrvatskoj 72% pripada prvoj kategoriji, a 28% drugoj [27], stariji od 60 godina smatraju se rizičnom skupinom čija je vjerojatnost obolijevanja od težeg oblika bolesti (potreba za respiratorom) uz posljedično veću smrtnost; Hrvatska ima ukupno 800 respiratora [28] i nešto više od 4 milijuna ljudi pa je u modelu $R=200$. Ukupan broj iteracija (dana) u svim simulacijama je fiksiran na 1200. U svim simulacijama interni podbačaji (uvezeni slučajevi) za COVID-19 bolest počinju u 15. i završavaju u 45. danu (država npr. zatvori granice odnosno uvede posebne mjere karantene za putnike).

Za svaku uređenu četvorku slobodnih parametara promatrano je nekoliko relevantnih značajki:

[3] R_0 koji se najčešće nalazi u 95% intervalu pouzdanosti svih spomenutih članaka u [20].

[4] Iz ovih postotaka smrtnost pacijenata na respiratoru, uz *najbolju moguću njegu*, ispada nešto manja od 30% što po najnovijim istraživanjima izgleda optimistično.

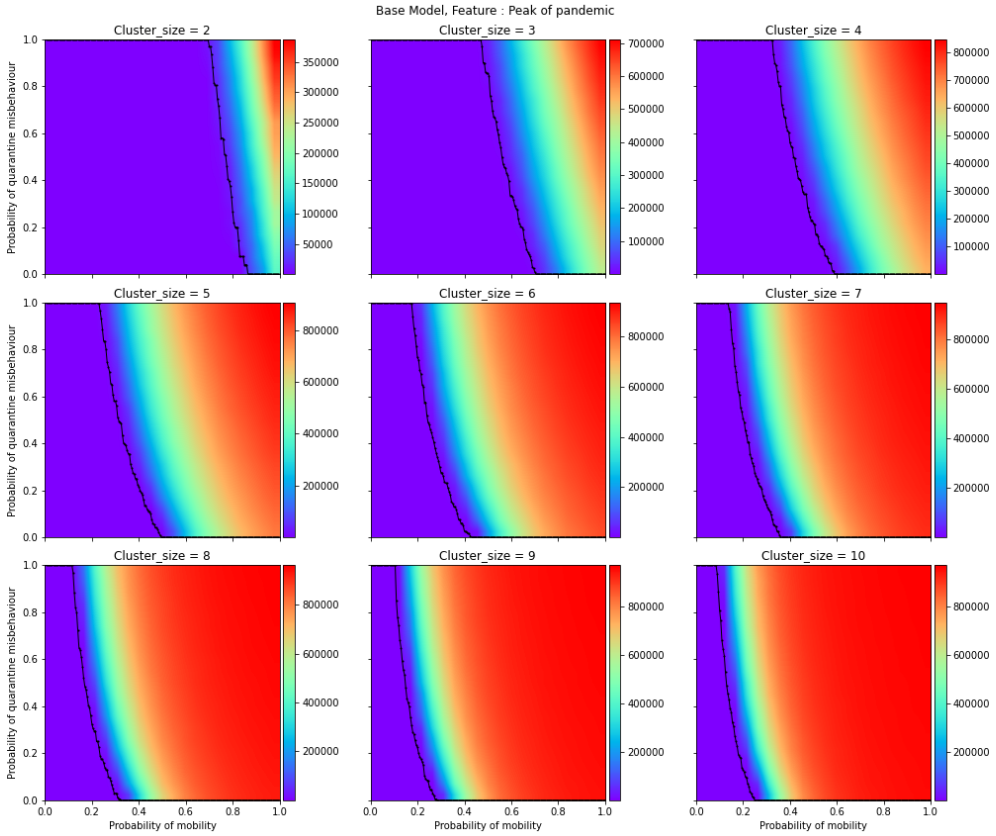
[5] U praksi postotak je puno veći tijekom sezonskih bolesti poput gripe.

- **Vrhunac pandemije**: maksimum zbroja jedinki u stanjima *infected*, *mild* i *icu covid19* (po svim iteracijama se promatra zbroj i odabere maksimum).
- **Ukupna smrtnost (COVID-19)**: broj jedinki u stanju *dead covid19* u zadnjoj iteraciji.
- **Ukupna smrtnost (nespecificirana)**: broj jedinki u stanju *dead nocovid19* u zadnjoj iteraciji.
- **Duljina pandemije**: broj dana između prve *infected* i zadnje *mild* jedinke.
- **Duljina kolapsa**: broj dana u kojima je zdravstveni sustav u kolapsu. Ako u bilo kojem danu i proizvoljnoj jedinki treba respirator a svi su popunjeni kažemo da je sustav u kolapsu (u tom danu).
- **Prvi dan kolapsa**: broj dana simulacije u kojem se prvi put dogodio kolaps zdravstvenog sustava.
- **Relativna promjena (nespecificiranih) smrtnosti**: relativna promjena broja umrlih od nespecificiranih respiratornih bolesti s obzirom na očekivani broj.
- **Postotak imunih**: postotak živih jedinki koje su prebolile COVID-19.

4.3 Prikaz značajki osnovnog modela

U ovom potpoglavlju prikazujemo sve relevantne značajke simulacija u ovisnosti o slobodnim parametrima. Na x-osi prikazan je parametar p_1 (u grafovima *probability of mobility*), na y-osi parametar p_2 (u grafovima *probability of quarantine misbehaviour*), s desne strane grafova nalazi se legenda. Crna linija označava **rub kolapsa zdravstvenog sustava**: za vrijednosti parametara s "lijeve" strane ruba kolaps se nije nikad dogodio, kažemo da je **pandemija pod kontrolom**, dok se s "desne" strane kolaps dogodio u barem jednom danu. Za očekivati je da veća vrijednost p_1 povlači vjerojatniji kolaps sustava (veća mobilnost jedinki izvan svog grozda), slično vrijedi za p_2. Slijedi grafički prikaz značajki *osnovnog* modela čiji su parametri i struktura mreže prethodno opisani.

Slika 4.1 prikazuje značajku *vrhunac pandemije* za veličine grozdova od 2 do 10. Može se jasno primijetiti da vrhunac ovisi o veličini grozdova (veći grozdovi povlače ekstremnije vrhunce), vidimo da scenariji u kojima je pandemija pod kontrolom imaju vrhunac od otprilike 50.000 jedinki. Suprotno, u najekstremnijim scenarijima vidimo da vrhunac

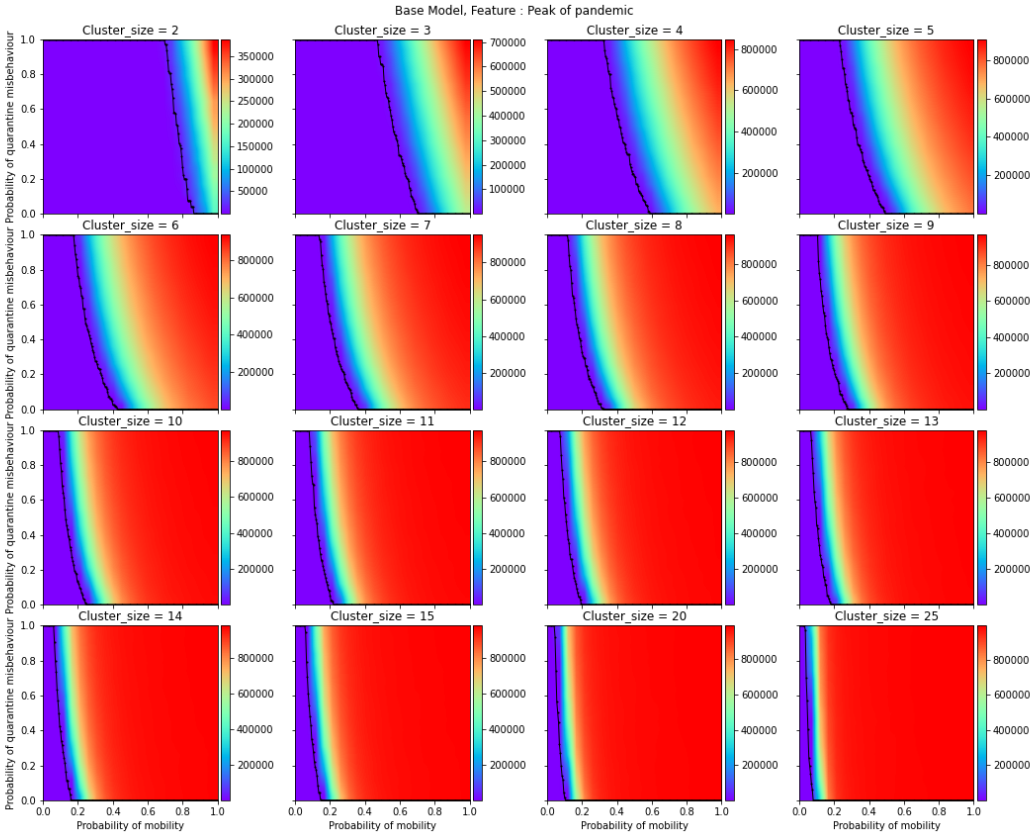


Slika 4.1: Značajka: vrhunac pandemije. Veličine grozdova 2–10.

može dosegnuti i do 800.000 jedinki što govori da u takvim simulacijama u najgorem danu imamo oko 80% zaraženih jedinki istovremeno. (ovaj scenariji je svakako moguć, npr. u gustom gradu od milijun ljudi gdje nikakve mjere nisu poduzete). Pod pretpostavkom da u normalnim uvjetima svaka osoba izlazi van barem jednom ($p_1 = 1$) možemo vidjeti da i za najpovoljnije veličine grozdova (2–3) potrebno je **prosječno smanjenje kretanja ljudi za minimalno 15**% ($p_1 = 0.85$), ali u **stvarnosti za barem 30**%, da bi se izbjegao kolaps[6].

Slika 4.2 prikazuje značajku *vrhunac pandemije* za veličine grozdova od 2 do 25. U simulacijama s jako velikim grozdovima možemo uočiti iznenadne tranzicije (eng. *abrupt transitions*) sustava za jako male razlike u parametrima odnosno sustav pokazuje dvofazno ponašanje (kontrola pandemije i totalna katastrofa bez međufaza), što je za očekivati [10]. Vidimo također da je parametar p_2 sve manje značajan za velike grozdove, zaključujemo da život u **jako velikim zajednicama nije povoljna epidemiološka situacija**, a **samoizolacija** u takvim zajednicama nakon otkrića jednog slučaja, vjerojatno **neće imati željene**

[6]Crna se linija ruba za veličinu grozdova 2 nalazi u intervalu 0.7-0.85 a za veličinu grozdova 3 u intervalu 0.45-0.7 (po parametru p_1). Za veličinu prosječnog Hrvatskog kućanstva vidi opis slike 4.4, str. 22.



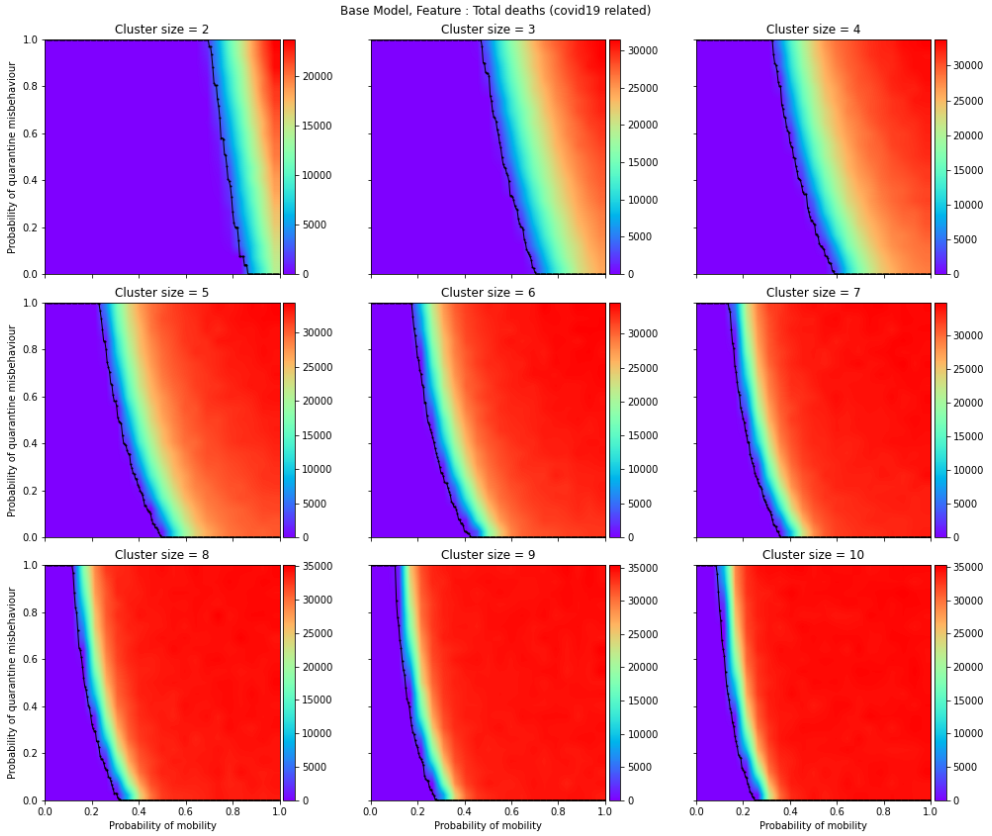
Slika 4.2: Značajka: vrhunac pandemije. Veličine grozdova 2–25.

rezultate (bolest se već jako raširila).

Slika 4.3 prikazuje značajku *ukupna smrtnost (COVID-19)* za veličine grozdova od 2 do 10. Situacija se jako dobro poklapa sa slikom 4.1 što je očekivano. Vidimo da u scenarijima u kojima je pandemija pod kontrolom broj umrlih od COVID-19 bolesti nikad ne prelazi 5.000 (0.5% populacije) dok se popne na preko 30.000 u najekstremnijim scenarijima (većina onih koji su mogli biti spašeni uz bolničku njegu su umrli jer je sustav bio u kolapsu).

Slika 4.4 prikazuje značajku *duljina pandemije* za veličine grozdova od 2 do 10. Primijetimo da se područja **ekvilibrija**, jako duge pandemije ali s relativno niskim vrhuncem (vidi sliku 4.1) nalaze **jako blizu ruba** kolapsa (crvene ”mrlje” su jako blizu crne linije). Uočimo da je takvo ponašanje prilično iznenađujuće i da je zdravstveni sustav, barem što se tiče broja respiratora, **blizu optimalne točke**.

Pandemije, ako su nekontrolirane, rastu eksponencijalno; u slučaju COVID-19 bolesti u kojoj barem 3% bolesnika zahtijeva intenzivnu njegu, nemoguće je očekivati od zdravstvenog sustava da primi $80\% \cdot 3\% = 2.4\%$ cijele populacije. Zdravstveni sustav s takvim



Slika 4.3: Značajka: ukupna smrtnost (COVID-19). Veličine grozdova 2–10.

kapacitetom bio bi izuzetno skup i uglavnom neiskorišten osim u uvjetima destruktivnih pandemija. Ono što se može očekivati je upravo otpornost na situacije ekvilibrija, odnosno endemske situacije[7].

Vidimo da se u Hrvatskoj za veličinu grozdova 2 i 3 nalazimo u optimalnoj situaciji (crvene ”mrlje” su lijevo od ruba) dok smo za ostale grozdove u suboptimalnoj. Kako je veličina prosječnog kućanstva[8] u Hrvatskoj oko 2.8 [26] zaključujemo da bi Hrvatska, uz eventualnu dobru alokaciju dostupnih respiratora, trebala zadovoljiti svoje potrebe kroz ovu pandemiju[9].

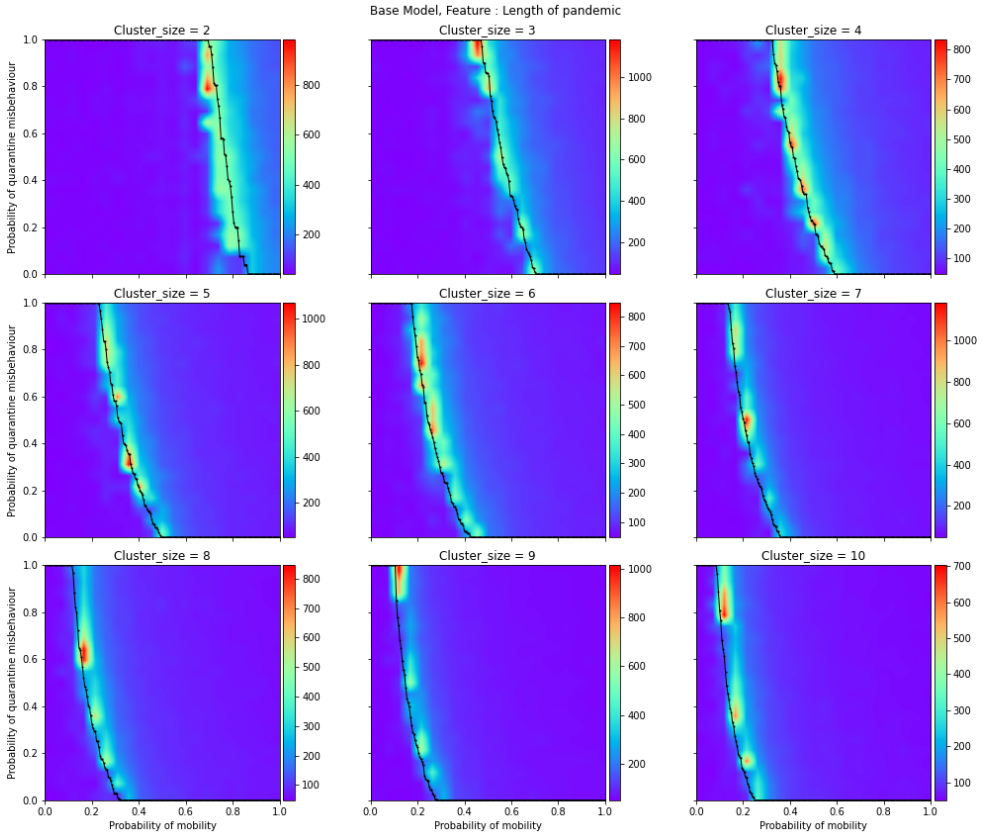
Slika 4.5 prikazuje značajku *duljina kolapsa* za veličine grozdova od 2 do 10. Stepenasti oblik je konstrukt diskretnog rešetkastog pretraživanja (eng. grid search)[10]. Vidimo

[7]U onim uvjetima gdje je bolest dovoljno raširena da se održi ali kroz relativno mali i gotovo konstantni broj slučajeva, bez strmih vrhunaca.

[8]Podsjetimo se da je veličina grozda u pravilu nešto veća od veličine kućanstva jer obuhvaća i druge eventualne kontatke pod uvjetom da su *svakodnevni*.

[9]Uz mjere socijalnog distanciranja koje bi držale pandemiju pod kontrolom.

[10]Crni rub se ponekad ne poklapa savršeno sa ljubičastom bojom, to je isto konstrukt diskretne rešetke, **crni rub** je stvoren s finijom rešetkom dakle **pouzdaniji** je.



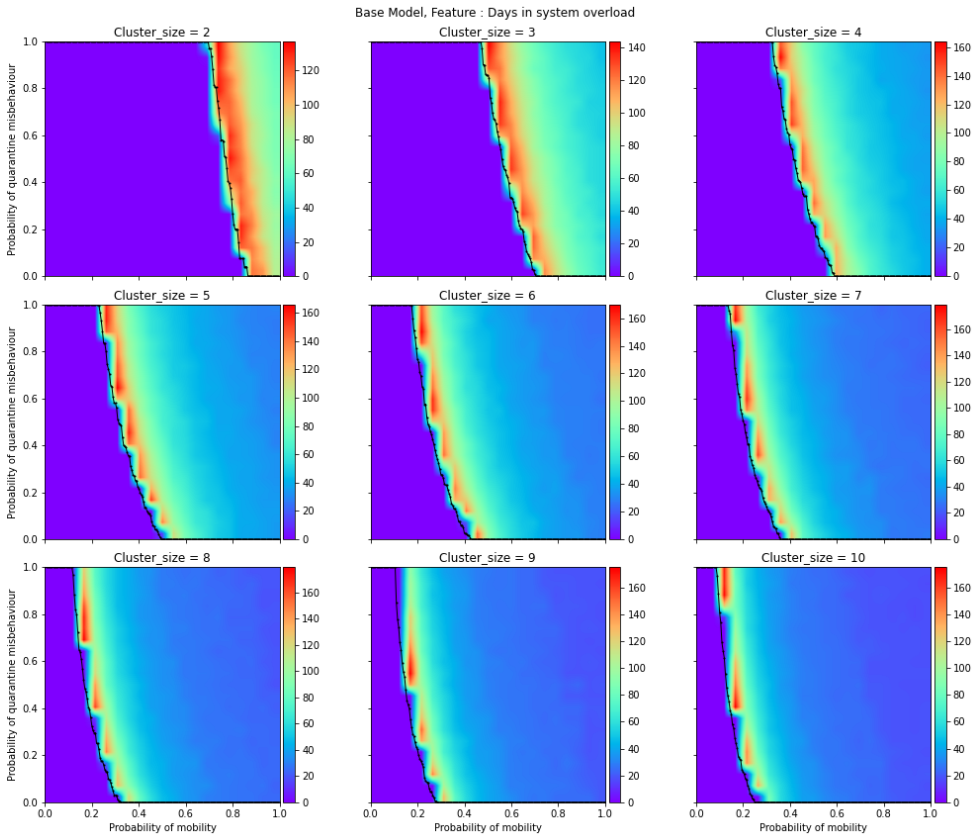
Slika 4.4: Značajka: duljina pandemije. Veličine grozdova 2–10.

da su scenariji u kojima je zdravstveni sustav **najdulje u kolapsu jako blizu ruba**, to su situacije u kojima su poduzete mjere socijalnog distanciranja dovoljno jake da smanje vrhunac pandemije (slika 4.1) ali nedovoljno da izbjegnu kolaps zdravstvenog sustava. Uočimo da je u najekstremnijim situacijama po broju umrlih i vrhuncu pandemije (slike 4.1 i 4.3) duljina kolapsa relativno kratka: odnosno scenariji u kojima pandemija ima katastrofalne posljedice na sreću kratko traju.

Slika 4.6 prikazuje značajku *relativna promjena (nespecificiranih) smrtnosti* za veličinu grozdova 4 i vrijednosti parametra μ: 5,500 i 50.000. Uočimo da je u scenarijima kolapsa zdravstvenog sustava **broj umrlih nevezanih direktno za COVID-19 povećan**[11] i za 60%-160% od očekivanog (legendu tumačimo na način da broj 0.6 znači inkrement od 60% a -0.2 znači smanjenje od 20%). Vidimo da veći parametar μ povlači veće inkremente (crvena boja u grafu s $\mu = 5$ predstavlja inkrement od 60% a u grafu s $\mu = 50.000$ od 160%). Značajka je prikazana samo u onim situacijama gdje je pandemija trajala dulje od 100 dana (slika 4.4), što je dovoljno dug period da rezultati ne ovise o slučajnim

[11] Formalno, odnosi se na osobe koje su koristile respirator.

događajima (zakon velikih brojeva).



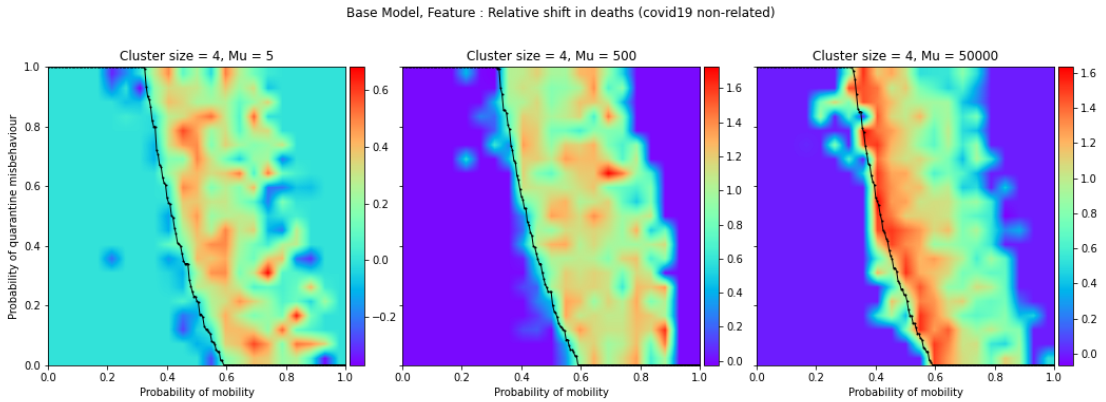
Slika 4.5: Značajka: duljina kolapsa. Veličine grozdova 2–10.

4.4 Ekstenzije

Bitna pretpostavka (možda dosad prešućena) osnovnog modela je **homogena distribucija populacije**: svaka jedinka je inicijalizirana na slučajan način s vjerojatnostima danim iz kategorijskih omjera; posljedica je da većina grozdova ima miješanu populaciju. Takva pretpostavka ne mora vrijediti, štoviše uvođenje nehomogene distribucije odnosno *specijaliziranih* grozdova može dodati novo bogatsvo simulacijama.

Specijalizirani domovi

Specijalizirani domovi su ekstenzija u kojoj se određen postotak grozdova pretvori u tzv. grozdove specijaliziranih domova. Takvi specijalni grozdovi imaju jednaku veličinu kao i ostali grozdovi u simulaciji, razlika je u tome što njihove jedinke **poštuju mjere više od**



Slika 4.6: Značajka: relativna promjena (nespecificiranih) smrtnosti. Veličine grozdova 4, parametar μ: 5, 500 i 50.000.

prosjeka: vjerojatnost odlazaka na putovanje $p_1/10$, vjerojatnost (ne)poštivanja samoizolacije $p_2/10$ gdje su p_1 i p_2 parametri za ostatak populacije u danoj simulaciji (vjerojatnost odlaska na putovanje ako u stanju *mild* je 0, kao i za ostatak populacije, vidi potpoglavlje 3.3)

Riječima: jedinke iz ovih grozdova **idu na putovanje puno rjeđe** od ostatka. Zbog logike i jednostavnosti, sve jedinke iz specijaliziranih domova su iz druge kategorije dakle starije od 60 godina.

Superširitelji

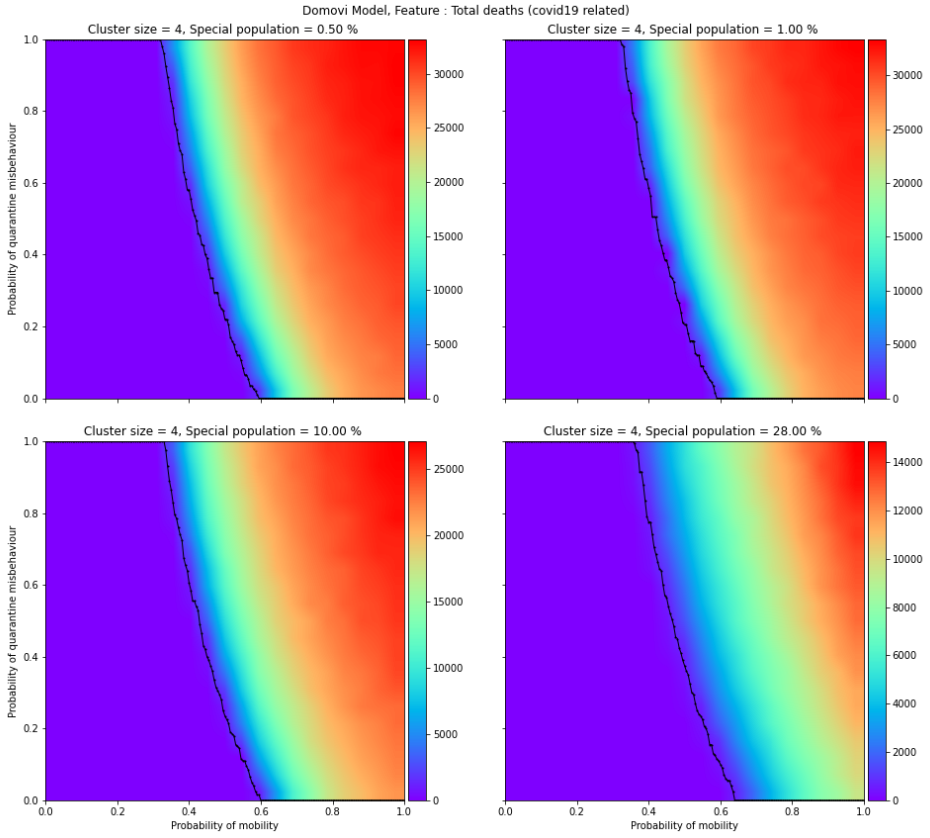
Superširitelji (eng. Superspreaders) su ekstenzija u kojoj se određen postotak grozdova pretvori u tzv. grozdove superširitelja. Takvi specijalni grozdovi imaju jednaku veličinu kao i ostali grozdovi u simulaciji, razlika je u tome što njihove jedinke **ne poštuju mjere**: vjerojatnost odlazaka na putovanje $p_1 = 1$, vjerojatnost (ne)poštivanja samoizolacije $p_2 = 1$ i čak ako su u stanju *mild* (simptomatično bolesni) odlaze na putovanje s vjerojatnosti 1. Riječima: jedinke iz ovih grozdova **idu na putovanje svaki dan** čak i ako su bolesne. Zbog logike i jednostavnosti, superširitelji su svi iz prve kategorije dakle mlađi od 60 godina.

Za obje ektenzije potreban je dodatni parametar sp (eng. special population) čija vrijednost označava broj specijalnih jedinki (koje popunjavaju specijalne grozdove)[12]. Parametar sp je u grafovima dan kako postotak (%).

[12]Npr. ako u ekstenziji *Superširitelji* stavimo $sp = 10.000$, mreža će biti inicijalizirana s 990.000 "običnih" po grozdovima homogeno distribuiranih i 10.000 specijalnih jedinki.

Prikaz značajki ekstenzije *specijalizirani domovi*

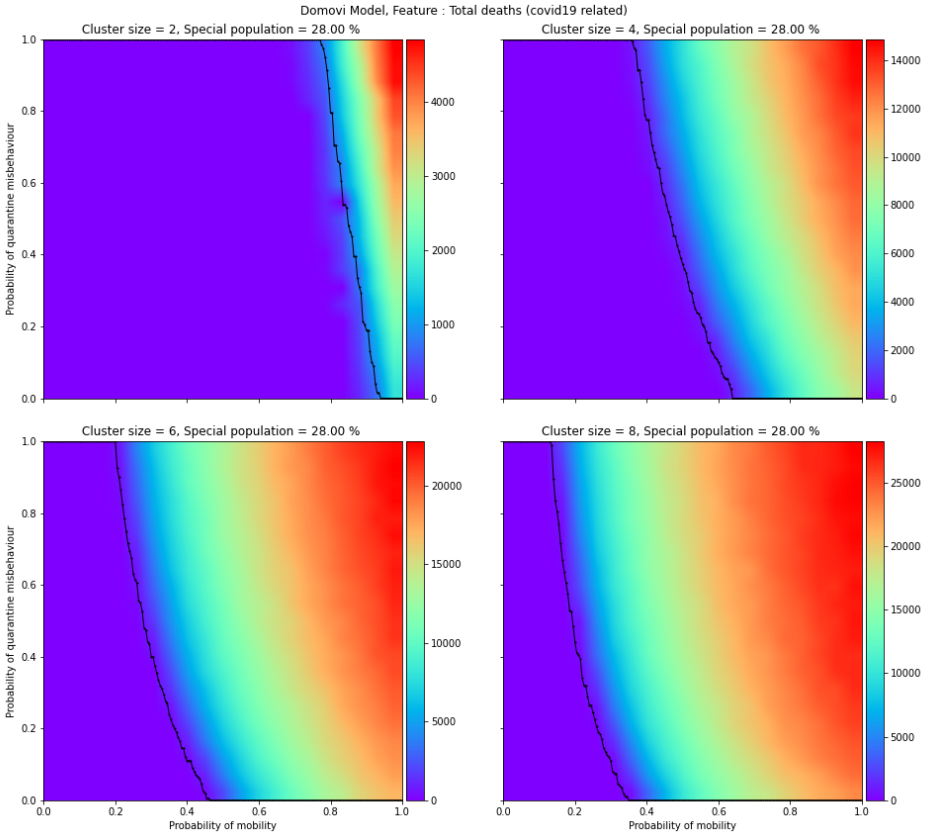
Slika 4.7 prikazuje značajku *ukupna smrtnost (COVID-19)* za veličine grozdova 4 i parametar *sp* dan u postotcima od 0.5% do 28%. Uočimo da veći broj jedinki u specijaliziranim domovima povlači manju ukupnu smrtnost od COVID-19 bolesti: možemo primijetiti povećanu glatkoću u prostoru parametara pogotovo za $sp = 28\%$ (međufaze su jako široke i najektremniji scenariji s 30.000 pada na 14.000 umrlih). Dakle posebno stroga **izolacija osoba starijih od 60 godina**, kao rizična skupina, je **jako efikasna mjera** u borbi protiv COVID-19 bolesti. Treba napomenuti kako u Hrvatskoj ima malo manje od 20.000 ljudi u domovima za starije i nemoćne osobe [11] što je oko 0.5% populacije. Nažalost kao što vidimo taj broj je premali: većina ljudi iz rizične skupine mora biti u strožoj izolaciji da bi rezultati bili zadovoljavajući.



Slika 4.7: Značajka: ukupna smrtnost (COVID-19). Veličine grozdova 4, *sp*: 0.5%-28%.

Slika 4.8 prikazuje značajku *ukupna smrtnost (COVID-19)* za veličine grozdova 2–8 i parametar $sp = 28\%$. Odnosno promatramo danu značajku u ovisnosti o veličini grozdova u najpovoljnijim uvjetima: stroga izolacija cijele populacije starije od 60 godina (28% Hrvatske populacije). Rezultati su očekivani: puno manje umrlih (usporedi sa slikom 4.3).

Posebno odskače graf koji se odnosi na veličine grozdova 2, ako ga usporedimo s grafom sa slike 4.3 za istu veličinu grozdova vidimo ne samo da je rub kolapsa značajno pomaknut udesno nego i da u najkatastrofalnijem scenariju ima ukupno 4.000 umrlih od COVID-19 (u usporedbi s preko 20.000 u osnovnom modelu), što je više nego 5 puta manje. Prirodan zaključak je taj da bi **izolacija starijih osoba bila učinkovitija u malim zajednicama** od 1–3 osobe[13].



Slika 4.8: Značajka: ukupna smrtnost (COVID-19). Veličine grozdova 2–8, $sp = 28\%$.

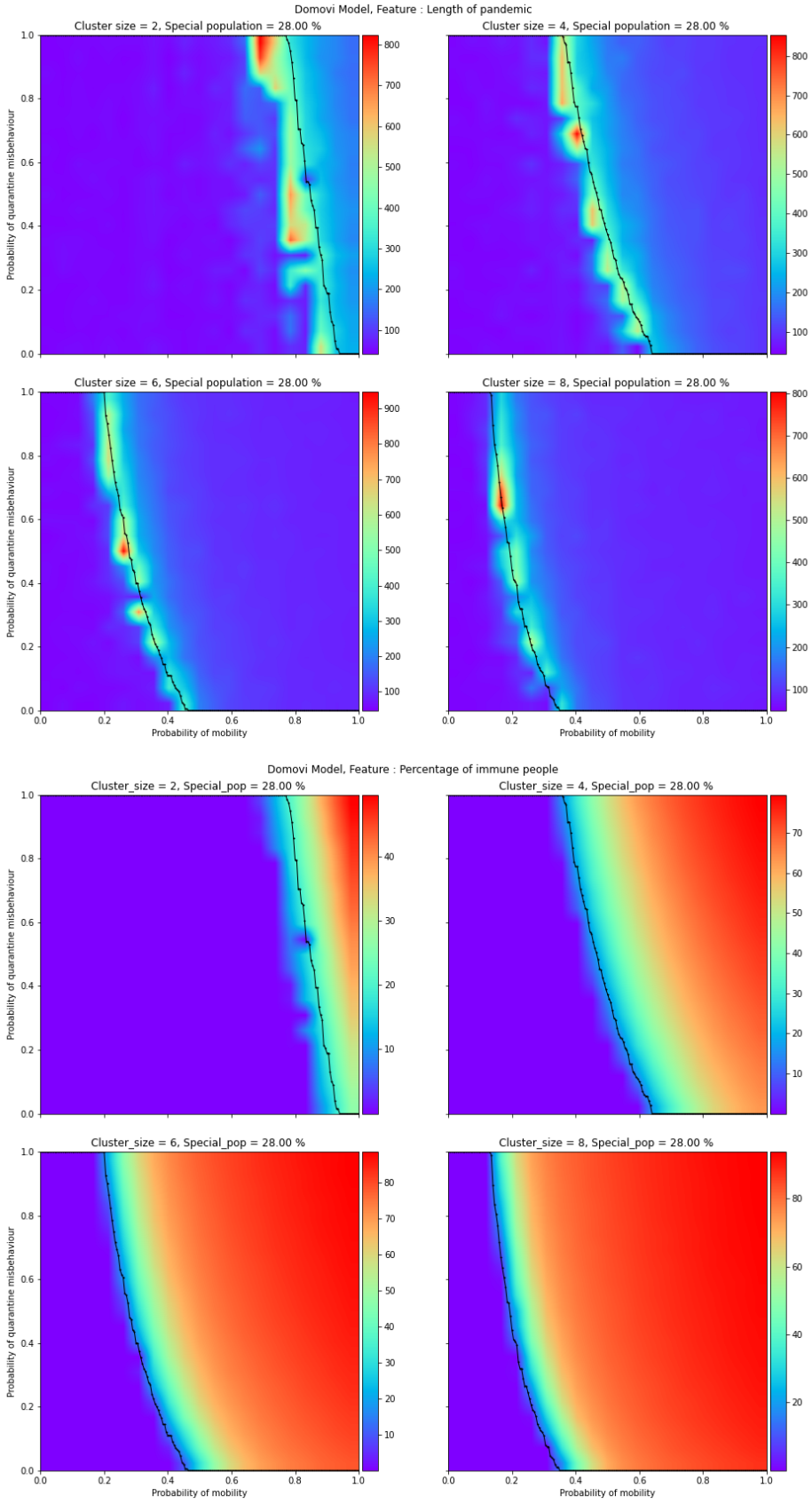
Slika 4.9 (gornji dio) prikazuje značajku *duljina pandemije* za veličine grozdova 2–8 i parametar $sp = 28\%$. Opservacije i zaključci su jako slični onima vezanih za sliku 4.4 razlika je u tome što u ovoj situaciji zdravstveni sustav izgleda **optimalno pozicioniran** za sve veličine grozdova jednake 6 ili manje (crvene mrlje su lijevo od ruba kolapsa). Ta opservacija dodatno potvrđuje status izolacije starijih osoba i rizičnih skupina kao efikasnu metodu u borbi protiv pandemije.

Slika 4.9 (donji dio) prikazuje značajku *postotak imunih* za veličine grozdova 2–8 i parametar $sp = 28\%$. Vidimo da postotak imunih, pod uvjetom da **sustav nije nikad u**

[13]Npr. izolirane sobe/stanovi sa minimalnim međusobnim kontaktima.

kolapsu ne prelazi 30%. Promatrajući gornji dio iste slike vidimo da je prosječna duljina pandemije oko ruba ≈ 400 dana, što povlači da su potrebne barem 2 godine da 60% populacije stekne imunitet, što se smatra realističnim pragom za bolesti poput COVID-19 za **imunitet krda**[14] [18].

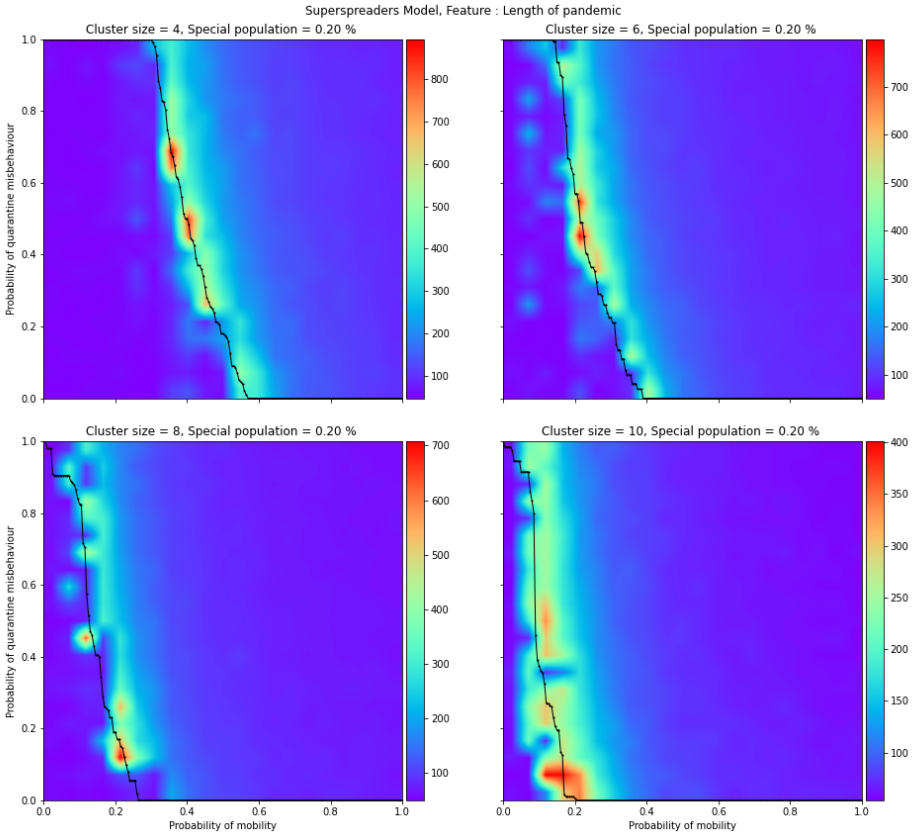
[14] Količina imunih u populaciji postane dovoljno velika da spriječi daljnje širenje infekcije.



Slika 4.9: Značajke: duljina pandemije (gore) i postotak imunih (dolje), grozdovi 2–8, $sp = 28\%$.

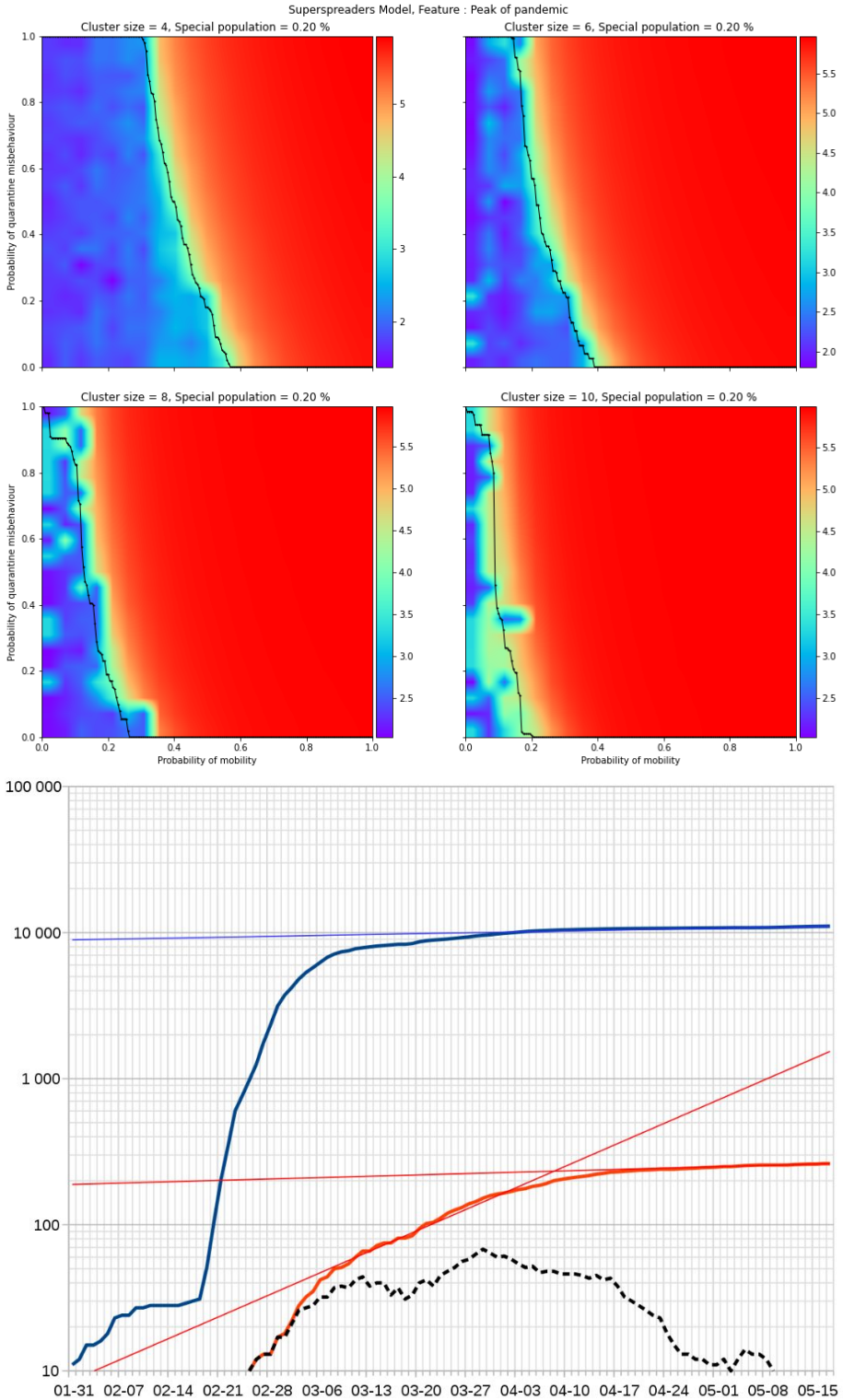
Prikaz značajki ekstenzije *superširitelji*

Slika 4.10 prikazuje značajku *duljina pandemije* za veličine grozdova 4-10 i parametar $sp = 0.2\%$. Dakle već u situaciji gdje samo 0.2% populacije ne poštuje apsolutno nikakve mjere možemo primijetiti veliku **nestabilnost** (područja lijevo od ruba imaju nekoliko "mrlja" u kojima su pandemije neobično duge) kao i kolapse sustava na neočekivanim gotovo slučajnim lokacijama (crni rub ima neobične "udubine"). U situacijama gdje se to dogodilo pogođen je jedan (ili više) grozd superširitelja koji je očito dovoljan da promijeni tok pandemije na razini cijele nacije/grada. Vidi slučaj u Južnoj Koreji gdje je jedna simptomatska osoba, tijekom vjerskih okupljanja, zarazila preko 35 drugih osoba i značajno pogoršala epidemiološku situaciju države [19], donji dio slike 4.11 (preuzeto s [16]).



Slika 4.10: Značajka: duljina pandemije. Veličine grozdova 4-10, $sp = 0.2\%$.

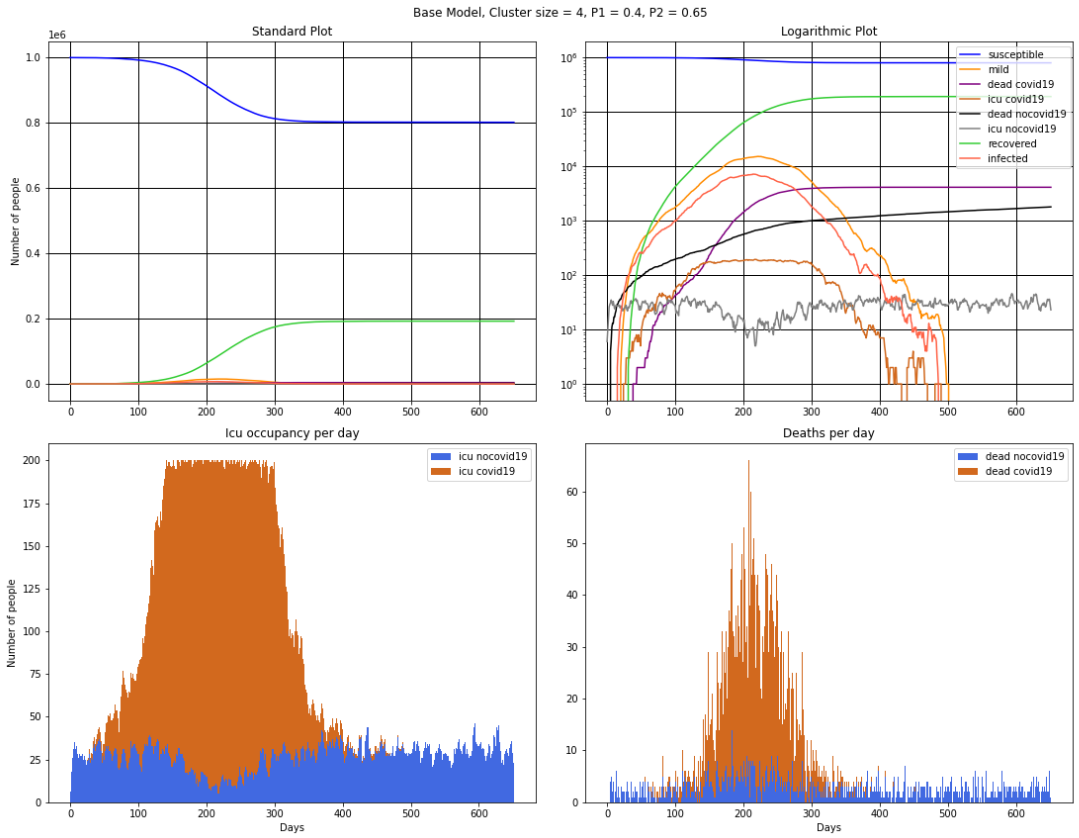
Slika 4.11 prikazuje značajku *vrhunac pandemije* (u logaritamskoj skali) za veličine grozdova 4-10 i parametar $sp = 0.2\%$. Zaključujemo da grozdovi superširitelja ne utječu samo na duljinu pandemije nego i na njen vrhunac (jako puno skokova čak i na mjestima koja nisu vidljiva na slici 4.10)



Slika 4.11: Gore, značajka: vrhunac pandemije *(log-skala)*. Veličine grozdova 4-10, $sp = 0.2\%$. Dolje, tijek COVID-19 pandemije u Južnoj Koreji. Broj ukupno otkrivenih slučajeva (plavo) i broj umrlih (crveno).

4.5 Vremenski nizovi

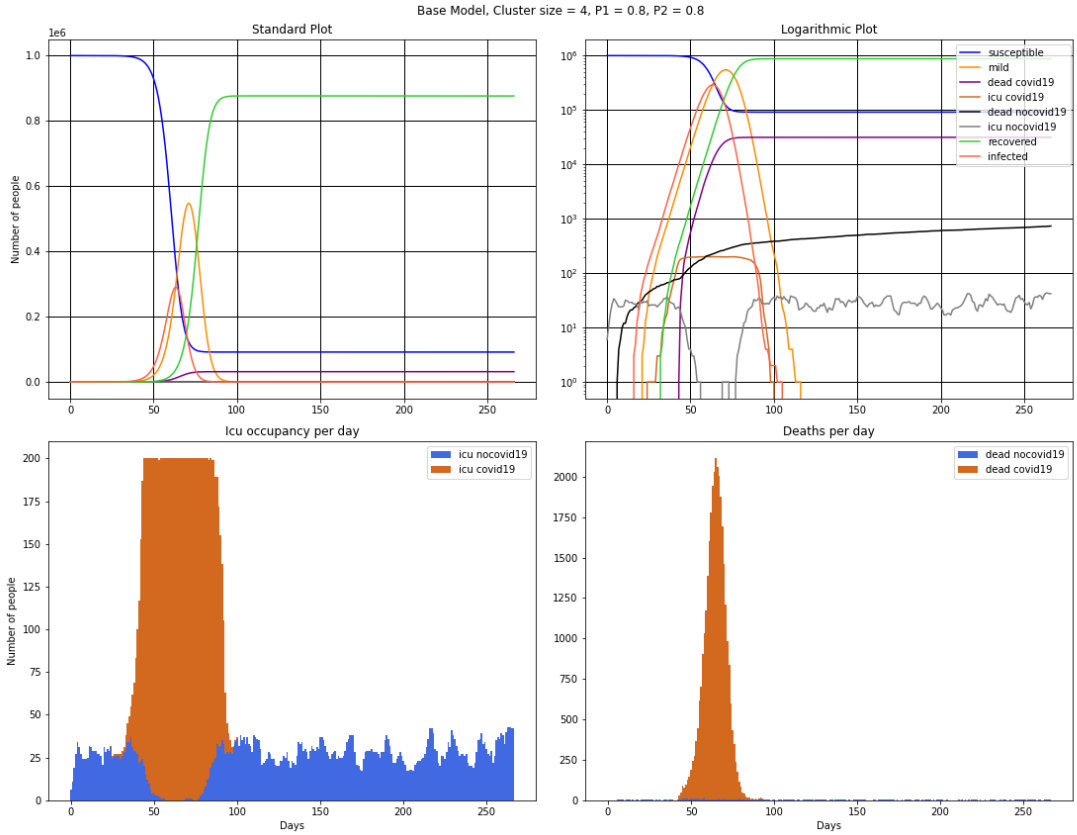
U ovom potpoglavlju grafički prikazujemo rezultate pojedinačnih simulacija za dane slobodne parametre (svaki graf je jedna točka u parametarskom prostoru). Svi grafovi su podijeljeni u četiri podgrafa i organizirani na isti način: u gornjem dijelu prikazani su brojevi jedinki u svakom stanju kroz iteracije u običnoj skali (lijevo) i logaritamskoj skali (desno); u donjem dijelu je fokus na popunjenost icu jedinica odnosno respiratora (lijevo) i broj umrlih (desno). U svim podgrafovima na x-osi nalaze se iteracije (dani), a na y-osi broj jedinki. Prikazani su samo dani u vremenu trajanja pandemije.



Slika 4.12: Osnovni model, veličina grozdova 4, $p_1 = 0.4$ i $p_2 = 0.65$.

Slika 4.12 prikazuje *osnovni* model za veličinu grozdova 4 i parametrima $p_1 = 0.4$ i $p_2 = 0.65$. Sa slike 4.3 vidimo da se točka nalazi iznad ruba u **području kolapsa**. Kolaps se može jasno opservirati u donjem lijevom podgrafu gdje broj jedinki na respiratoru doseže maksimalnu brojku od 200; takvo stanje se zadrži oko 150 dana što očito predstavlja broj

dana u kolapsu[15]. U donjem desnom podgrafu vidimo situaciju s brojem umrlih gdje se vrhunac dosegne oko 220. dana i iznosi ≈ 60. Vidimo da je i broj umrlih koji nisu direktno povezani s COVID-19 također povećan u tom razdoblju. Gornji podgrafovi prikazuju broj jedinki i u ostalim stanjima: iz logaritamske skale vidimo da je u ovom scenariju ukupni broj umrlih od COVID-19 bolesti ≈ 5.000 i da je 20% populacije steklo imunitet (što se vidi iz obične skale).



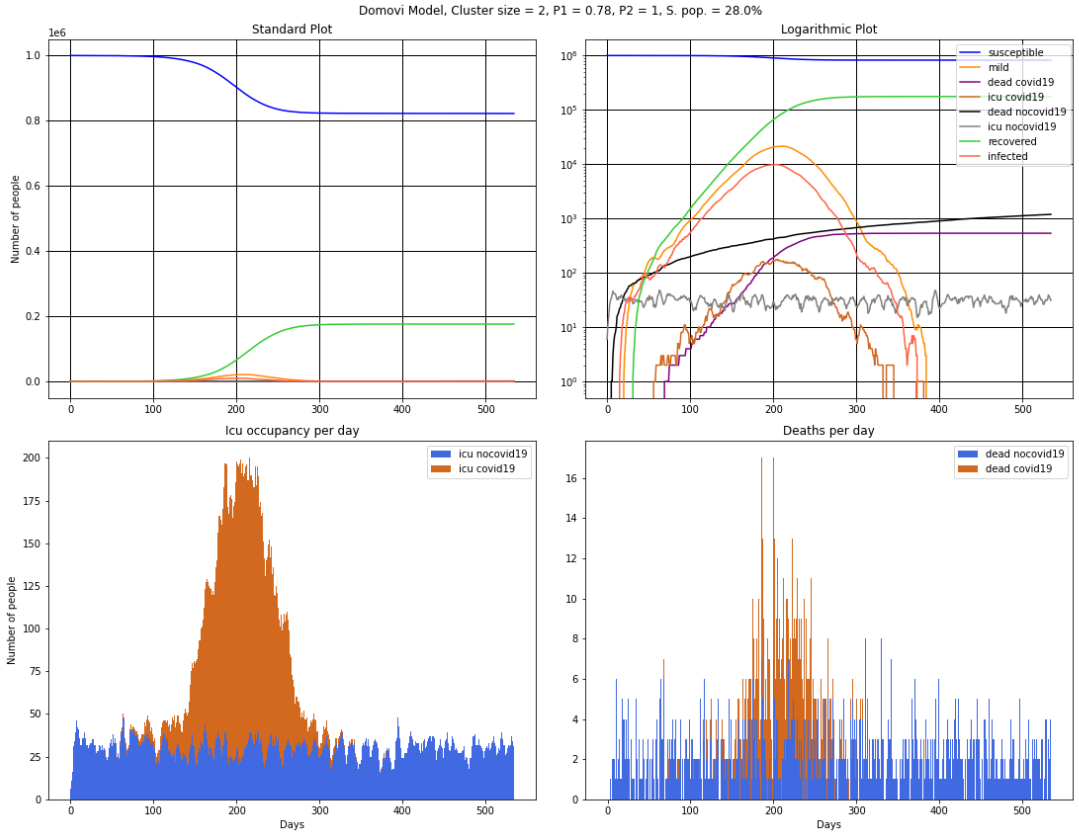
Slika 4.13: Osnovni model, veličina grozdova 4, $p_1 = 0.8$ i $p_2 = 0.8$.

Slika 4.13 prikazuje *osnovni* model za veličinu grozdova 4 i parametrima $p_1 = 0.8$ i $p_2 = 0.8$. Sa slike 4.3 vidimo da se točka nalazi desno od ruba u **području ekstremnog kolapsa**. Kolaps se također može jasno opservirati u donjem lijevom podgrafu, ali u trajanju od samo 50 dana. Uočimo kako tijekom vrhunca pandemije COVID-19 bolesnici preuzimaju gotovo sve respiratore[16]. U donjem desnom podgrafu vidimo situaciju s bro-

[15]Ponekad ukupan broj zauzetih respiratora je nešto manji od 200, to nije greška već proizvod slučajnog procesa umiranja i skidanja s respiratora (ako svaki dan obavještavamo o broju u isti sat može se dogoditi da se jedna osoba skine prije nego što druga popuni prazno mjesto).

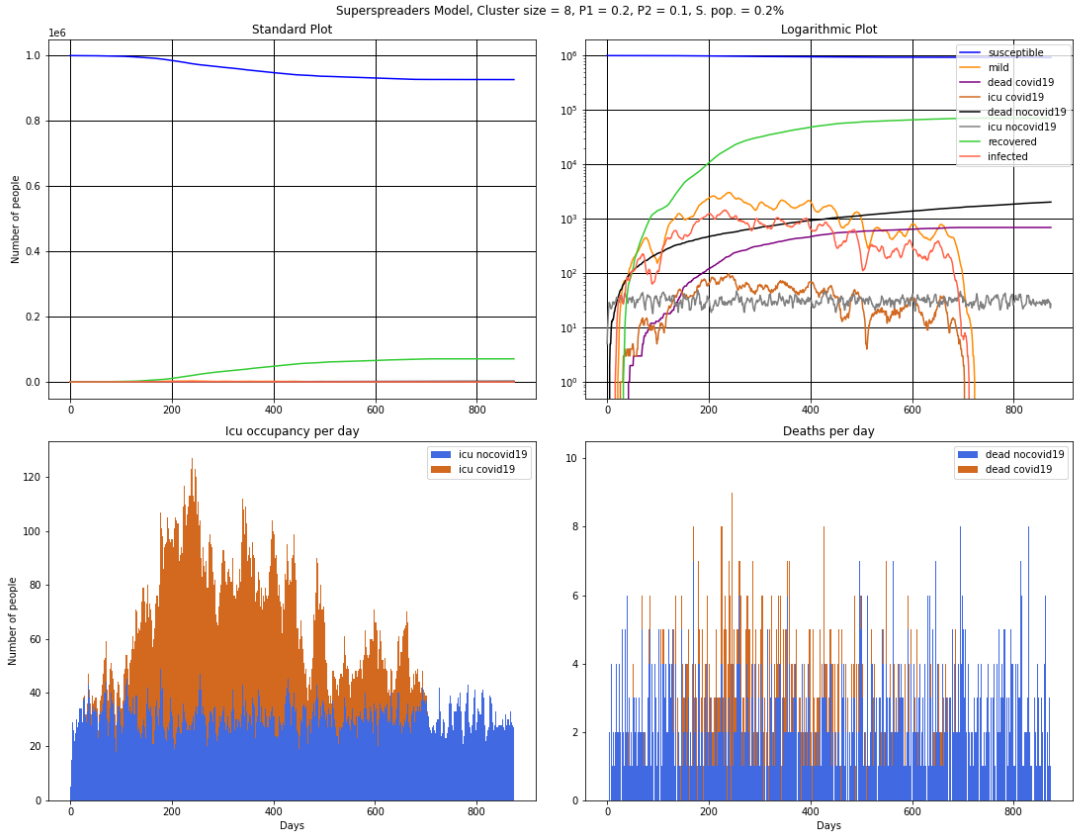
[16]Za vrijeme vrhunca pandemije ima puno više COVID-19 bolesnika nego ostalih dakle veća je vjerojatnost da neko od njih dobije slobodni respirator.

jem umrlih koja pokazuje stvarno stanje ekstremne katastrofe: vrhunac se dosegne oko 60. dana i prelazi 2.000, takva dnevna količina umrlih je ogromna uzevši u obzir da simuliramo s tek $N = 10^6$ jedinki. Gornji podgrafovi prikazuju broj jedinki i u ostalim stanjima: u običnom grafu vidimo "tipične" krivulje koje podsjećaju na SIR model (slika 2.2).



Slika 4.14: Specijalizirani domovi model, veličina grozdova 2, $p_1 = 0.78$, $p_2 = 1$ i $sp = 28\%$.

Slika 4.14 prikazuje model *specijaliziranih domova* za veličinu grozdova 2 i parametrima $p_1 = 0.78$, $p_2 = 1$ i $sp = 28\%$. Sa slike 4.8 vidimo da se točka nalazi jako blizu ruba, ali **izvan područja kolapsa**. Promatrajući donje podgrafove vidimo da postoje tri trenutka u kojima su gotovo svi respiratori popunjeni, ali istovremeno broj umrlih je razumno nizak. Promatrajući gornje podgrafove vidimo da je tijekom pandemije, koja je trajala oko godinu dana, ukupni broj umrlih od COVID-19 bolesti ispod 1.000 ali da je čak 20% populacije steklo imunitet. Grafovi potvrđuju sve izvedene zaključke slika 4.7, 4.8 i 4.9, odnosno da je izolacija starijih osoba jako efikasna mjera.

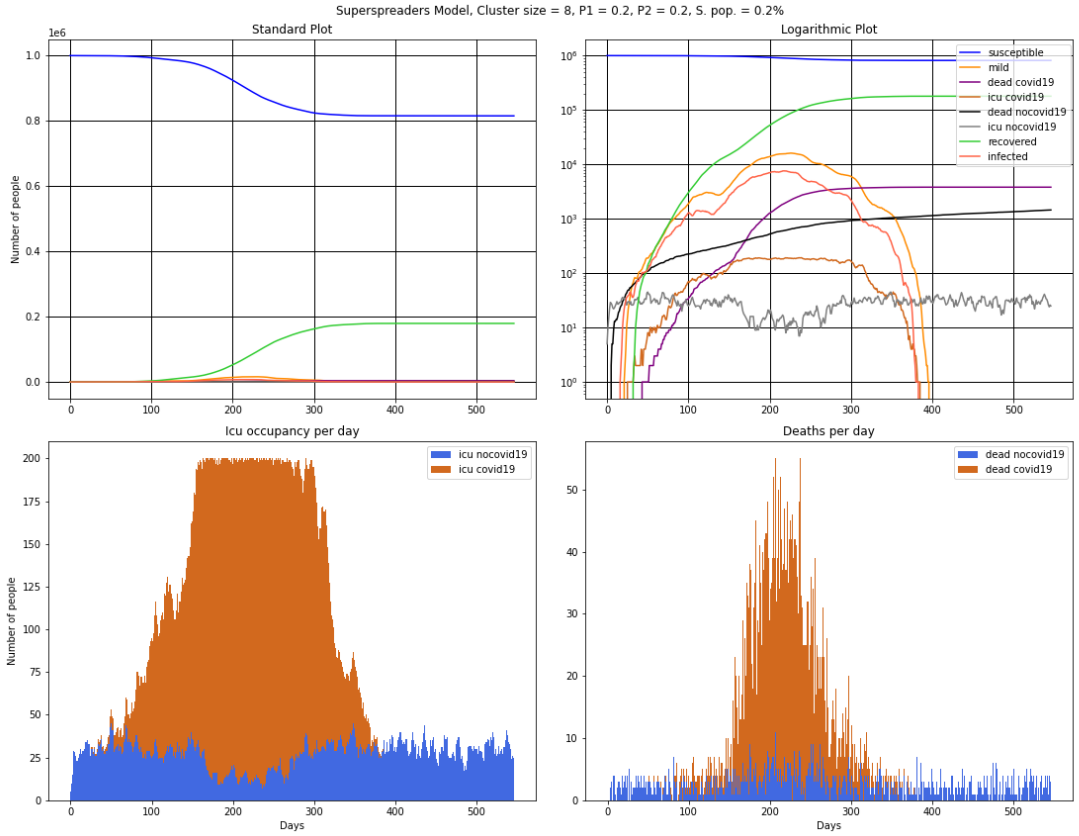


Slika 4.15: Superširitelji model, veličina grozdova 8, $p_1 = 0.2$, $p_2 = 0.1$ i $sp = 0.2\%$.

Slika 4.15 prikazuje model *superširitelja* za veličinu grozdova 8 i parametrima $p_1 = 0.2$, $p_2 = 0.1$ i $sp = 0.2\%$. Sa slike 4.10 vidimo da se točka nalazi jako blizu ruba **izvan područja kolapsa** u zoni "crvene mrlje" odnosno u **ekvilibriju**. Zaista, interesantna situacija može se vidjeti u gornjem desnom podgrafu gdje vidimo jasan ekvilibrij kojeg očigledno održavaju (zbog relativno širokih valova) privremena širenja zaraze na grozdove superširitelja (koja su se u ovom scenariju pokazala kao nedovoljno jaka da dovedu sustav u kolaps).

Slika 4.16 prikazuje model *superširitelja* za veličinu grozdova 8 i parametrima $p_1 = 0.2$, $p_2 = 0.2$ i $sp = 0.2\%$. Sa slike 4.10 vidimo da se točka nalazi jako blizu ruba u **području kolapsa**. Zanimljiv događaj može se primijetiti u gornjem desnom podgrafu oko 120. dana pandemije gdje se pogodi jedan (ili više) grozd *superširitelja* koji epidemiju od ≈ 3.000 aktivnih slučajeva[17] prebaci na preko 25.000; odnosno, iz **područja kontrolirane epidemije**, **prebaci u stanje kolapsa**. "Koljeno" u broju *infected* slučajeva podsjeća na donji graf slike 4.11 odnosno na nagli porast slučajeva izazvan jednim superširiteljem u

[17] Zbroj crvenih (*infected*), narančastih (*mild*) i smeđih (*icu covid19*) jedinki.



Slika 4.16: Superširitelji model, veličina grozdova 8, $p_1 = 0.2$, $p_2 = 0.2$ i $sp = 0.2\%$.

Južnoj Koreji[18].

[18]Graf Južne Koreje prikazuje kumulativan broj (poznatih slučajeva) ipak iznenadni rastući trend slučajeva je karakterističan i usporediv sa simulacijom.

Poglavlje 5

Zaključak

U ovom radu provedena je detaljna analiza mogućih scenarija i ishoda COVID-19 pandemije s podacima i parametrima koji su prilagođeni stanju u Republici Hrvatskoj.

Promatranjem simulacija zaključilo se da su bolnički kapaciteti, i specifično broj respiratora, u teoriji dovoljni za adekvatnu njegu svima kojima će biti potrebni, pod uvjetom da su implementirane potrebne rigorozne mjere (barem 30% manje kretanja ljudi od uobičajenog). Najefikasniji uočeni princip suživota s COVID-19 bolesti je posebna razina izolacije za rizične skupine i osobe starije od 60 godina. Nažalost, čak i pod takvim uvjetima suživot s virusom neće sigurno biti kraći od 2 godine, što je minimalno vrijeme do stjecanja imuniteta krda (uvodenje cjepiva može skratiti ovaj period).

Uočeno je također, da uz lošu politiku i neadekvatnu primjenu mjera ljudi neće samo umirati zbog COVID-19 nego i zbog drugih respiratornih bolesti. Broj umrlih od takvih bolesti će biti i do 160% veći od uobičajenog.

Konačno, kao što je pokazala situacija u Južnoj Koreji, dovoljan je mali broj ($< 1\%$) zaraženih ljudi (ili ljudi koji su bili u kontaktu sa zaraženom osobom) koji ne poštuju mjere samoizolacije da se epidemiološka slika cijele države brzo promijeni i zdravstveni sustav dovede u kolaps.

Zanimljivih proširenja ovog rada ima nekoliko: implementacija situacije da rigoroznije mjere povlače manju dostupnost bolničke njege i time veću smrtnost za sve ostale bolesti; implementacija cijele ekonomske pozadine koja povlači niz drugih optimizacijskih problema i/ili uvođenje imuniteta koji nije doživotan.

Dodatak A

Implementacija najbitnijih dijelova projekta

Implementacija modela, programski jezik: C++.

```
#include <algorithm>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <vector>
#include <cmath>
#include <random>
#include "json.hpp"

using json = nlohmann::json;
using RandomGenerator = std::minstd_rand;

enum class PersonState {
  SUSCEPTIBLE, // s
  INFECTIOUS, // i
  CONFIRMED, // c
  ICU, // ic
  DEAD, // d
  IMMUNE, // im
  NOCORONA_ICU, //nic
  NOCORONA_DEAD, // nd
};
const int NUM_STATES = static_cast<std::underlying_type<PersonState>::type> (
    PersonState::NOCORONA_DEAD) + 1;

std::string state_to_name(PersonState state) {
  switch (state) {
    case PersonState::SUSCEPTIBLE:
      return "susceptible";
    case PersonState::INFECTIOUS:
      return "infectious";
    case PersonState::CONFIRMED:
      return "confirmed";
    case PersonState::ICU:
      return "icu";
    case PersonState::DEAD:
      return "dead";
    case PersonState::IMMUNE:
      return "immune";
    case PersonState::NOCORONA_ICU:
      return "nocorona_icu";
    case PersonState::NOCORONA_DEAD:
      return "nocorona_dead";
```

```
44    }
45    // Can't happen.
46    exit(1);
47  }
48
49  struct Person {
50    Person(int id, int category) : id(id), category(category) {}
51
52    int id;
53    int category;
54    PersonState state = PersonState::SUSCEPTIBLE;
55    int days_until_next_state = 0;
56    // Needed because person can change state from IMMUNE to NOCORANA_ICU. If
57    // patient survives ICU then we need to know does it return to IMMUNE or
58    // SUSCEPTIBLE state.
59    bool is_immune = false;
60  };
61
62  struct CategoryParams {
63    CategoryParams(json params) :
64        prob_goes_on_trip(params["prob_goes_on_trip"]),
65        prob_c_trip_candidate(params["prob_c_trip_candidate"]),
66        prob_c_neighbour_trip_candidate(
67            params["prob_c_neighbour_trip_candidate"]),
68        prob_s_to_i(params["prob_s_to_i"]),
69        days_i_to_c(params["days_i_to_c"]),
70        prob_i_to_ic(params["prob_i_to_ic"]),
71        days_c_to_im(params["days_c_to_im"]),
72        days_ic_to_im_or_c(params["days_ic_to_im_or_c"]),
73        prob_ic_to_d(params["prob_ic_to_d"]),
74        prob_to_nic(params["prob_to_nic"]),
75        prob_nic_to_d(params["prob_nic_to_d"]),
76        days_nic(params["days_nic"]) {}
77
78    double prob_goes_on_trip;
79    double prob_c_trip_candidate;
80    double prob_c_neighbour_trip_candidate;
81    double prob_s_to_i;
82    int days_i_to_c;
83    double prob_i_to_ic;
84    int days_c_to_im;
85    int days_ic_to_im_or_c;
86    double prob_ic_to_d;
87    double prob_to_nic;
88    double prob_nic_to_d;
89    int days_nic;
90  };
91
92  class Graph;
93  json simulate(Graph &, json, RandomGenerator &);
94
95  class Graph {
96    public:
97      Graph(json graph_params, RandomGenerator &generator) {
98        for (const auto &subgraph_params: graph_params) {
99          // Generate num_persons. Put each person in one of the categories.
100         int num_clusters = subgraph_params["num_clusters"];
101         int num_people_per_cluster = subgraph_params["num_people_per_cluster"];
102         int last_bound = 0;
103         std::vector<int> category_bounds;
104         for (auto &x : subgraph_params["category_ratios"]) {
105           last_bound += x.get<int>();
106           category_bounds.push_back(last_bound);
107         }
108
109         std::uniform_int_distribution<> distribution(
110             0, category_bounds.back() - 1);
111         for (int i = 0; i < num_clusters; ++i) {
112           std::vector<Person> cluster;
113           for (int j = 0; j < num_people_per_cluster; ++j) {
114             int x = distribution(generator);
115             int category = std::upper_bound(
```

```
116                 category_bounds.begin(), category_bounds.end(), x)
117                   - category_bounds.begin();
118               int id = i * num_people_per_cluster + j;
119               Person person(id, category);
120               cluster.push_back(person);
121             }
122             clusters.push_back(cluster);
123           }
124         }
125       }
126
127     private:
128       friend json simulate(Graph &, json, RandomGenerator &);
129       std::vector<std::vector<Person>> clusters;
130   };
131
132   class BoolWithProbability {
133     public:
134       BoolWithProbability(RandomGenerator &generator) : generator(generator) {}
135
136       bool operator()(double p) {
137         return distribution(generator) < p;
138       }
139
140     private:
141       RandomGenerator &generator;
142       std::uniform_real_distribution<> distribution{0, 1};
143   };
144
145   double dying_probability(double p, double mu, double system_load) {
146     return 1 - (1 - p) * exp(-mu * system_load);
147   }
148
149   // Returns whether icu overflow happened.
150   inline bool before_trip_cluster_update(
151       std::vector<Person> &cluster,
152       int &num_icus_left,
153       const std::vector<CategoryParams> &params_for_categories,
154       double prob_transmission,
155       double mu,
156       double system_load,
157       BoolWithProbability &bool_with_probability) {
158     // Count number of infected people that can transmit corona virus in this
159     // cluster. We do this only once before calculating transitions of people
160     // so that all people are in analog position.
161     int cnt_infectious_persons = 0;
162     for (const auto &x : cluster) {
163       if (x.state == PersonState::INFECTIOUS) {
164         ++cnt_infectious_persons;
165       }
166     }
167     double p_in_cluster_transmission =
168         1 - pow(1 - prob_transmission, cnt_infectious_persons);
169
170     bool icu_overflow = false;
171     for (int i = 0; i < cluster.size(); ++i) {
172       auto &x = cluster[i];
173       const auto &params = params_for_categories[x.category];
174
175       if (x.state == PersonState::SUSCEPTIBLE) {
176         if (bool_with_probability(params.prob_s_to_i) ||
177             bool_with_probability(p_in_cluster_transmission)) {
178           x.state = PersonState::INFECTIOUS;
179           x.days_until_next_state = params.days_i_to_c;
180         }
181
182       } else if (x.state == PersonState::INFECTIOUS) {
183         if (!--x.days_until_next_state) {
184           // Person became symptomatic so he is either put in isolation or in
185           // icu.
186           if (bool_with_probability(params.prob_i_to_ic)) {
187             if (!num_icus_left) {
```

```
188             // Person need icu but there are none left so he dies.
189             x.state = PersonState::DEAD;
190             icu_overflow = true;
191           } else {
192             x.state = PersonState::ICU;
193             x.days_until_next_state = params.days_ic_to_im_or_c;
194             --num_icus_left;
195           }
196         } else {
197           x.state = PersonState::CONFIRMED;
198           x.days_until_next_state = params.days_c_to_im;
199         }
200       }
201
202     } else if (x.state == PersonState::CONFIRMED) {
203       if (!--x.days_until_next_state) {
204         x.state = PersonState::IMMUNE;
205         x.is_immune = true;
206       }
207
208     } else if (x.state == PersonState::ICU) {
209       if (!--x.days_until_next_state) {
210         double p_ic_to_d = dying_probability(
211             params.prob_ic_to_d, mu, system_load);
212         if (bool_with_probability(p_ic_to_d)) {
213           // Person died in ICU.
214           x.state = PersonState::DEAD;
215         } else {
216           // Person made it through ICU, so he is now mild case.
217           x.state = PersonState::CONFIRMED;
218           x.days_until_next_state = params.days_c_to_im;
219         }
220         ++num_icus_left;
221       }
222
223     } else if (x.state == PersonState::NOCORONA_ICU) {
224       if (!--x.days_until_next_state) {
225         double p_nic_to_d = dying_probability(
226             params.prob_nic_to_d, mu, system_load);
227         if (bool_with_probability(p_nic_to_d)) {
228           // Person died in ICU of illness that is not corona.
229           x.state = PersonState::NOCORONA_DEAD;
230         } else if (x.is_immune) {
231           x.state = PersonState::IMMUNE;
232         } else {
233           x.state = PersonState::SUSCEPTIBLE;
234         }
235         ++num_icus_left;
236       }
237     }
238
239     if (x.state == PersonState::IMMUNE ||
240         x.state == PersonState::SUSCEPTIBLE ||
241         x.state == PersonState::CONFIRMED ||
242         x.state == PersonState::INFECTIOUS) {
243       // Person can require ICU from other illnesses, not only corona.
244       if (bool_with_probability(params.prob_to_nic)) {
245         if (num_icus_left) {
246           if (x.state == PersonState::CONFIRMED ||
247               x.state == PersonState::INFECTIOUS) {
248             // If person who had corona went to an icu for unrelated reasons we
249             // presume that he will get over that corona infection during his
250             // time in icu. That does not neccessarily follow from given days
251             // for NOCORONA_ICU state and different stages of corona disease
252             // from config but simplifies implementation and has almost no
253             // impact on simulation since number of NOCORONA_ICU people is
254             // expected to be small.
255             x.is_immune = true;
256           }
257           x.state = PersonState::NOCORONA_ICU;
258           x.days_until_next_state = params.days_nic;
259           --num_icus_left;
```

```
260           } else {
261             x.state = PersonState::NOCORONA_DEAD;
262             icu_overflow = true;
263           }
264         }
265       }
266     }
267
268     return icu_overflow;
269   }
270
271   json simulate(Graph &g, json simulation_config, RandomGenerator &generator) {
272     BoolWithProbability bool_with_probability(generator);
273     // Extracting configuration parameters.
274     int num_days = simulation_config["stopping_conditions"]["num_days"];
275     bool on_icu_overflow =
276       simulation_config["stopping_conditions"]["on_icu_overflow"];
277     int num_icus_left = simulation_config["num_icus"];
278     double mu = simulation_config["mu"];
279     double prob_transmission = simulation_config["prob_transmission"];
280     double k_trip = simulation_config["k_trip"];
281     bool isolate_cluster_on_known_case =
282         simulation_config["isolate_cluster_on_known_case"];
283
284     std::vector<CategoryParams> params_for_categories;
285     auto all_params = simulation_config["initial_params"];
286     for (const auto &params : all_params) {
287       params_for_categories.emplace_back(params);
288     }
289
290     std::vector<json> events = simulation_config["events"];
291     sort(events.begin(), events.end(), [](const json &x, const json &y) {
292         return x["day"] < y["day"];
293     });
294     auto event = events.begin();
295
296     // Declaring stats variables.
297     std::unordered_map<std::string, std::vector<int>> num_per_state_history;
298     std::string stopping_condition = "num_days";
299     int num_days_icu_overflow = 0;
300     int first_day_icu_overflow = -1;
301     int last_day_icu_overflow = -1;
302
303     for (int day = 0; day < num_days; ++day) {
304       std::cerr << "Simulating day " << day << "/" << num_days << "\n";
305       bool this_day_icu_overflow = false;
306
307       while (event != events.end() && event->at("day") == day) {
308         auto update_params = event->at("update_params");
309         for (auto param: update_params.items()) {
310           if (!all_params[0].count(param.key())) {
311             std::cerr << "Invalid key `" << param.key() << "` in an event\n";
312             exit(1);
313           }
314           if (param.value().size() != all_params.size()) {
315             std::cerr << "Invalid number of categories for key `" << param.key()
316               << "` in an event\n";
317             exit(1);
318           }
319           for (int i = 0; i < all_params.size(); ++i) {
320             all_params[i][param.key()] = param.value()[i];
321           }
322         }
323         params_for_categories.clear();
324         for (const auto &params : all_params) {
325           params_for_categories.emplace_back(params);
326         }
327         ++event;
328       }
329
330       // Calculate system_load factor.
331       int cnt_alive_people = 0;
```

```
332     int cnt_burden = 0;
333     for (const auto &cluster : g.clusters) {
334       for (const auto &x : cluster) {
335         if (x.state != PersonState::DEAD &&
336             x.state != PersonState::NOCORONA_DEAD) {
337           ++cnt_alive_people;
338         }
339         if (x.state == PersonState::ICU ||
340             x.state == PersonState::CONFIRMED) {
341           ++cnt_burden;
342         }
343       }
344     }
345     double system_load = (double)cnt_burden / cnt_alive_people;
346
347     // Before "trip" updates.
348     for (auto &cluster : g.clusters) {
349       bool cluster_icu_overflow = before_trip_cluster_update(
350           cluster,
351           num_icus_left,
352           params_for_categories,
353           prob_transmission,
354           mu,
355           system_load,
356           bool_with_probability);
357       if (cluster_icu_overflow && !this_day_icu_overflow) {
358         this_day_icu_overflow = true;
359         ++num_days_icu_overflow;
360         last_day_icu_overflow = day;
361         if (first_day_icu_overflow == -1) {
362           first_day_icu_overflow = day;
363         }
364       }
365     }
366
367     // Pick people who go to the trip.
368     std::vector<Person*> persons_on_trip;
369     for (auto &cluster : g.clusters) {
370       // Check if there is someone with known corona disease in cluster.
371       bool has_known_corona = false;
372       for (auto &x : cluster) {
373         if (x.state == PersonState::ICU ||
374             x.state == PersonState::CONFIRMED) {
375           // It can happen that person gets to NOCORONA_ICU and already had
376           // corona. In that case this flag would stay false, however I don't
377           // think it is a problem since this should happen quite rarely.
378           has_known_corona = true;
379         }
380       }
381
382       for (auto &x : cluster) {
383         const auto &params = params_for_categories[x.category];
384         if (x.state == PersonState::SUSCEPTIBLE ||
385             x.state == PersonState::INFECTIOUS ||
386             x.state == PersonState::IMMUNE) {
387           if (!has_known_corona || !isolate_cluster_on_known_case ||
388               bool_with_probability(params.prob_c_neighbour_trip_candidate)) {
389             if (bool_with_probability(params.prob_goes_on_trip)) {
390               persons_on_trip.push_back(&x);
391             }
392           }
393         }
394         if (x.state == PersonState::CONFIRMED) {
395           // Person knows that it has corona but it can disobey order for
396           // staying home and becomes trip candidate.
397           if (bool_with_probability(
398                 params.prob_c_trip_candidate * params.prob_goes_on_trip)) {
399             persons_on_trip.push_back(&x);
400           }
401         }
402       }
403     }
```

```
404
405       // Count number of infectious and confirmed people on a trip.
406       int cnt_contagious = 0;
407       for (Person *x: persons_on_trip) {
408         if (x->state == PersonState::INFECTIOUS ||
409             x->state == PersonState::CONFIRMED) {
410           ++cnt_contagious;
411         }
412       }
413
414       // Spread infection during the trip.
415       double contagious_ratio = (double)cnt_contagious / persons_on_trip.size();
416       double p_transmission = std::min(
417           prob_transmission * k_trip * contagious_ratio, 1.0);
418       for (Person *x: persons_on_trip) {
419         if (x->state == PersonState::SUSCEPTIBLE &&
420             bool_with_probability(p_transmission)) {
421           const auto &params = params_for_categories[x->category];
422           x->state = PersonState::INFECTIOUS;
423           x->days_until_next_state = params.days_i_to_c;
424         }
425       }
426
427       // Collecting stats.
428       std::vector<int> num_per_state(NUM_STATES);
429       for (const auto &cluster : g.clusters) {
430         for (const auto &x : cluster) {
431           ++num_per_state[
432             static_cast<std::underlying_type<PersonState>::type>(x.state)];
433         }
434       }
435
436       // Iterating over all possible states instead of states in num_per_state
437       // map since it is not necessary that all possible states are there, but
438       // we still want to append zero to history vector.
439       for (int j = 0; j < NUM_STATES; ++j) {
440         auto state_name = state_to_name(static_cast<PersonState>(j));
441         num_per_state_history[state_name].push_back(num_per_state[j]);
442       }
443
444       if (this_day_icu_overflow && on_icu_overflow) {
445         stopping_condition = "icu_overflow";
446         break;
447       }
448     }
449
450     return {
451       {"stopping_condition", stopping_condition},
452       {"num_days_icu_overflow", num_days_icu_overflow},
453       {"first_day_icu_overflow", first_day_icu_overflow},
454       {"last_day_icu_overflow", last_day_icu_overflow},
455       {"stats", num_per_state_history},
456     };
457   }
458
459   int main(int argc, char *argv[]) {
460     if (argc != 3) {
461       std::cerr << "Expected arguments: config_file seed\n"
462                 << "  config_file = path to json configuration file\n"
463                 << "                (see example_config.json for format)\n"
464                 << "  seed = number passed to generator constructor\n\n";
465       exit(1);
466     }
467
468     std::ifstream config_stream(argv[1]);
469     json config;
470     config_stream >> config;
471     config_stream.close();
472
473     RandomGenerator generator(atoi(argv[2]));
474     Graph g(config["graph_generation"], generator);
475     auto data = simulate(g, config["simulation"], generator);
```

```
476     std::cout << data << "\n";
477     return 0;
478 }
```

Ekstrakcija značajki, programski jezik: Python.

```
from __future__ import print_function
import numpy as np
import json
from multiprocessing import Pool
import datetime
import sys
from argparse import ArgumentParser
from matplotlib import rc
import subprocess
import sys
import os

parser = ArgumentParser()
parser.add_argument('cluster_sizes', metavar='cluster_size', type=int, nargs='+')
parser.add_argument('-np', dest='num_processes', default=1, type=int,
                    help='Num of processors to use')
parser.add_argument('-mu', dest='mu', default=100, type=int,
                    help='mu parametar in model config')
parser.add_argument('-k', dest='k', default=10, type=float, help='k_trip parametar in model config')
parser.add_argument('-ext', dest='ext', default=0, type=int, help='extension type in simulation, 0:base model, 1:
    superspreaders, 2:domovi')
parser.add_argument('-extpop', dest='extpop', default=0, type=int, help='population in extra subgraph')
parsed = parser.parse_args()

# LOGICAL STRUCTURE OF THE GRID SEARCH: (keys:elements) is a dictionary, [] is a list.
        # cluster_dict : p1_dict : p2_dict : [seed_dict : [list of 8 elements], ratio_of_success_seeds]
        # cluster_dict is just logical, implementationally non necessary (we save the whole below structure using the
            cluster value)
            #(we parallelize therefore always only one value cluster.)
        # p1_dict is a dictionary which contains all values p1
        # p2_dict is a dictionary which contains all values p2
        # for each value of p2 we have a list of 2 elements a seed_dict and ratio_of_success_seeds
        # seed_dict is a dictionary which contains all values seed
        # for each value of seed we have a list of 5 elements (relevant info we want to keep track of) :
            #1 first day of infectious case,2 length of pandemic (first infectious, last mild case),
            #3 peak of pandemic (max (infectious+mild+ic) ) 4 peak of pandemic known cases,
            #5 total corona deaths, 6 total no_corona deaths
            #7 1st day system overload,8 n days in system overload.
        # ratio_of_success_seeds is simply the ratio of seeds for which the system is never in fail mode.

def model_cluster_trip(config_file_name, seed, devnull):
    if sys.version_info > (3, 0):
        return subprocess.run(["./model_cluster_trip_v2", config_file_name, str(seed)],
                stdout=subprocess.PIPE, stderr=devnull).stdout
    else:
        p = subprocess.Popen(["./model_cluster_trip_v2", config_file_name, str(seed)],
                stdout=subprocess.PIPE, stderr=devnull)
        return "".join(p.stdout.readlines())

def graph_generation(baseline_icu,baseline_nodes,baseline_days,scale,scaledays,ext,cluster_size,mu,k,extpop):
    num_icus=baseline_icu // scale
    num_days=baseline_days // scaledays
    config_filenames = [
            "config_for_grid_search.json",
            "config_for_grid_search_superspreaders.json",
            "config_for_grid_search_domovi.json"]
    with open(config_filenames[ext]) as f:
        config = json.load(f)
    if ext==0:
        num_nodes=baseline_nodes // scale
        num_clusters=num_nodes // cluster_size
        config["graph_generation"][0]["num_people_per_cluster"]=cluster_size
        config["graph_generation"][0]["num_clusters"]=num_clusters
    if ext==1 or ext==2: # superspreaders model and/or domovi model
        num_nodes0=(baseline_nodes-extpop) // scale
        num_clusters0=num_nodes0 // cluster_size
        num_nodes1=extpop // scale
        num_clusters1=num_nodes1 // cluster_size
        config["graph_generation"][0]["num_people_per_cluster"]=cluster_size
        config["graph_generation"][0]["num_clusters"]=num_clusters0
        config["graph_generation"][1]["num_people_per_cluster"]=cluster_size
```

```
70          config["graph_generation"][1]["num_clusters"]=num_clusters1
71          a=config["graph_generation"][0]["category_ratios"][0]
72          b=config["graph_generation"][0]["category_ratios"][1]
73          if ext==1:
74              config["graph_generation"][0]["category_ratios"][0]=int(max(baseline_nodes*a/(a+b)/scale-num_nodes1,0)) #
                    diminish <60 because some are superspreaders
75              config["graph_generation"][0]["category_ratios"][1]=int(max(baseline_nodes*b/(a+b)/scale,0))
76          if ext==2:
77              config["graph_generation"][0]["category_ratios"][0]=int(max(baseline_nodes*a/(a+b)/scale,0))
78              config["graph_generation"][0]["category_ratios"][1]=int(max(baseline_nodes*b/(a+b)/scale-num_nodes1,0)) #
                    diminish >=60 because some are in domovi
79      config["simulation"]["stopping_conditions"]["num_days"]=num_days
80      config["simulation"]["num_icus"]=num_icus
81      config["simulation"]["mu"] = mu
82      config["simulation"]["k_trip"] = k
83      config["simulation"]["events"][0]["update_params"]["prob_s_to_i"]=[el * scale for el in config["simulation"]["events
            "][0]["update_params"]["prob_s_to_i"]]
84      return config
85
86  def grid_search_parameters(config,p1,p2,ext,k,mu,cluster_size,seed,extpop):
87      config["simulation"]["initial_params"][0]["prob_goes_on_trip"] = p1
88      config["simulation"]["initial_params"][1]["prob_goes_on_trip"] = p1
89      config["simulation"]["initial_params"][0]["prob_c_neighbour_trip_candidate"] = p2
90      config["simulation"]["initial_params"][1]["prob_c_neighbour_trip_candidate"] = p2
91      baseline_file_name = "_tmp_config_rgs{}_{}_{}_{}"
92      extra_param = "_extpop{}"
93      if ext==0:
94          config_file_name = ("tmp/Base" + baseline_file_name + ".json").format(
95              k, mu, cluster_size, seed)
96      if ext==1:
97          config_file_name = ("tmp/Superspreaders" + baseline_file_name + extra_param + ".json").format(
98                  k, mu, cluster_size, seed, extpop)
99      if ext==2:
100         # >=60 in domovi 10 times more likely to be in quarantine and 10 times more likely not to go on trip.
101         config["simulation"]["initial_params"][2]["prob_goes_on_trip"] = p1/10
102         config["simulation"]["initial_params"][2]["prob_c_neighbour_trip_candidate"] = p2/10
103         config_file_name = ("tmp/Domovi" + baseline_file_name + extra_param + ".json").format(
104                 k, mu, cluster_size, seed, extpop)
105     return [config, config_file_name]
106
107 def save_json(ext,p1_dict, k, mu, cluster_size, extpop):
108     baseline_file_name1 = "_real_grid_search"
109     baseline_file_name2 = "_k_trip{}_mu{}_cluster_size{}.json"
110     extra_param = "_extpop{}"
111     if ext==0:
112         with open(("outputs/base_model/Base" + baseline_file_name1 + baseline_file_name2).format(
113                 k, mu, cluster_size), "w") as f:
114             json.dump(p1_dict, f, indent=4)
115     if ext==1:
116         with open(("outputs/superspreaders_model/Superspreaders" + baseline_file_name1 + extra_param +
                baseline_file_name2).format(
117                 extpop, k, mu, cluster_size), "w") as f:
118             json.dump(p1_dict, f, indent=4)
119     if ext==2:
120         with open(("outputs/domovi_model/Domovi" + baseline_file_name1 + extra_param + baseline_file_name2).format(
121                 extpop, k, mu, cluster_size), "w") as f:
122             json.dump(p1_dict, f, indent=4)
123
124 def f(cluster_size):
125     ext=parsed.ext
126     mu=parsed.mu
127     k=parsed.k
128     extpop=parsed.extpop
129     scale=1
130     scaledays=1
131     config=graph_generation(200,1000000,1200,scale,scaledays,ext,cluster_size,mu,k,extpop)
132     h=5 #put 1 for very precise grid
133     ptrip=np.arange(0,1.00001,0.01*h)
134     pdisobedient=np.arange(0,1.00001,0.01*h)
135     seeds=np.arange(0,1,1)
136     p1_dict={}
137     start = datetime.datetime.now()
```

```
138     devnull = open(os.devnull, 'w')
139     for p1 in ptrip:
140         p2_dict={}
141         for p2 in pdisobedient:
142             list_2len=[]
143             seed_dict={}
144             ratio_succ=-1
145             succ=0
146             for seed in seeds:
147                 config_list=grid_search_parameters(config,p1,p2,ext,k,mu,cluster_size,seed,extpop)
148                 config=config_list[0]
149                 config_file_name=config_list[1]
150                 with open(config_file_name, "w") as f:
151                     json.dump(config, f, indent=4)
152                 print("Running model with params: cluster_size = {:.3f}".format(cluster_size),
153                       ", prob_goes_on_trip = {:.3f}".format(p1),
154                       ", prob_c_neighbour_trip_candidate = {:.3f}".format(p2),
155                       "seed = {}".format(seed), file=sys.stderr)
156                 stdout = model_cluster_trip(config_file_name, seed, devnull)
157                 output=json.loads(stdout)
158                 os.remove(config_file_name)
159                 try:
160                     beginning_pandemic = next(x for x, val in enumerate(output["stats"]["infectious"]) if val > 0)
161                 except StopIteration:
162                     beginning_pandemic = -1
163                     end_pandemic = -1
164                     len_pandemic = -1
165                 else:
166                     beginning_pandemic = next(x for x, val in enumerate(output["stats"]["infectious"]) if val > 0)
167                     end_pandemic = len((output["stats"]["confirmed"])) - 1 - next(x for x, val
168                                                 in enumerate(reversed(output["stats"]["confirmed"])) if val > 0 )
169                     len_pandemic = end_pandemic - beginning_pandemic + 1
170                 peak_corona_total = max([sum(x) for x in zip( (output["stats"]["infectious"]),(output["stats"]["
                        confirmed"]),
171                                                                 (output["stats"]["icu"]))])
172                 peak_corona_system_load = max([sum(x) for x in zip( (output["stats"]["confirmed"]), (output["stats"]["
                        icu"]))])
173                 corona_deaths = output["stats"]["dead"][-1]
174                 no_corona_deaths = output["stats"]["nocorona_dead"][-1]
175                 total_immune = output["stats"]["immune"][-1]
176                 n_days_icu_overflow = output["num_days_icu_overflow"]
177                 first_day_icu_overflow = output["first_day_icu_overflow"]
178                 if n_days_icu_overflow==0:
179                     succ=succ+1
180                 seed_dict[str(seed)]=[beginning_pandemic,
181                                       len_pandemic,
182                                       peak_corona_total,
183                                       peak_corona_system_load,
184                                       corona_deaths,
185                                       no_corona_deaths,
186                                       n_days_icu_overflow,
187                                       first_day_icu_overflow,
188                                       total_immune]
189             ratio_succ=float(succ/len(seeds))
190             list_2len.append(seed_dict)
191             list_2len.append(ratio_succ)
192             p2_dict[str(p2)]=list_2len
193         p1_dict[str(p1)]=p2_dict
194     save_json(ext,p1_dict, k, mu, cluster_size, extpop)
195     devnull.close()
196     end = datetime.datetime.now()
197     print("Time elapsed during the calculation:", end - start)
198 if parsed.num_processes == 1:
199     print("Running ONE process", file=sys.stderr)
200     for cluster_size in parsed.cluster_sizes:
201         f(cluster_size)
202 else:
203     print("Running {} processes".format(parsed.num_processes), file=sys.stderr)
204     # This won't work on isabella, because python2 has different API for Pool
205     # object. However, that is fine since we run parallelize our code with SGE
206     with Pool(parsed.num_processes) as p:
207         p.map(f, parsed.cluster_sizes)
```

Zahvale

Posebno se zahvaljujem kolegi i prijatelju mag. ing. comp. Mislavu Bradaču na velikoj pomoći oko implementacije modela.
Hvala doc. dr. sc. Ani Meštrović na pruženoj pomoći u čitanju i razumijevanju stručnih medicinskih članaka i dr. sc. Tomislavu Lipiću na korisnim savjetima i na omogućenom pristupu računalnom klasteru Isabella, bez kojeg bi se većina simulacija još vrtila...
Konačno, jedno veliko hvala mentoru prof. dr. sc. Borisu Podobniku čije ideje, inicijativa i savjeti su glavni razlog postojanja ovog rada.

Bibliografija

[1] „A contribution to the mathematical theory of epidemics". *Proceedings of the Royal Society of London, Series A, Containing Papers of a Mathematical and Physical Character* 115.772 (1927), str. 700–721. DOI: 10.1098/rspa.1927.0118.

[2] P. Bak, K. Chen i C. Tang. „A forest-fire model and some thoughts on turbulence". *Physics Letters A* 147.5-6 (1990), str. 297–300. DOI: 10.1016/0375-9601(90)90451-s.

[3] N. G. Becker i K. Dietz. „The effect of household distribution on transmission and control of highly infectious diseases". *Mathematical Biosciences* 127.2 (1995), str. 207–219. DOI: 10.1016/0025-5564(94)00055-5.

[4] A. L. Barabási i R. Albert. „Emergence of Scaling in Random Networks". *Science* 286.5439 (1999), str. 509–512. DOI: 10.1126/science.286.5439.509.

[5] B. T. Grenfell, O. N. Bjørnstad i J. Kappey. „Travelling waves and spatial hierarchies in measles epidemics". *Nature* 414.6865 (2001), str. 716–723. DOI: 10.1038/414716a.

[6] F. Liljeros i dr. „The web of human sexual contacts". *Nature* 411.6840 (2001), str. 907–908. DOI: 10.1038/35082140.

[7] M. J. Keeling i K. Eames. „Networks and epidemic models". *Journal of The Royal Society Interface* 2.4 (2005), str. 295–307. DOI: 10.1098/rsif.2005.0051.

[8] S. R. Bapoje i dr. „Unplanned transfers to a medical intensive care unit: Causes and relationship to preventable errors in care". *Journal of Hospital Medicine* 6.2 (2010), str. 68–72. DOI: 10.1002/jhm.812.

[9] Andrés E. i dr. „Evolution of Mortality over Time in Patients Receiving Mechanical Ventilation". *ATS journals* (2013). DOI: 10.1164/rccm.201212-2169OC.

[10] A. Majdandzic i dr. „Spontaneous recovery in dynamical networks". *Nature Physics* 10.1 (siječanj 2013), str. 34–38. DOI: 10.1038/nphys2819.

[11] M. Bađun. „Financiranje domova za starije i nemoćne osobe u Hrvatskoj". *Revija za socijalnu politiku* 24.1 (2017). DOI: 10.3935/rsp.v24i1.1370.

[12] T. Britton, E. Pardoux i F. Ball. *Stochastic epidemic models with inference*. Springer, 2019.

[13] Pavan K. B. i dr. „Covid-19 in Critically Ill Patients in the Seattle Region — Case Series". *The New England Journal of Medicine* (svibanj 2020). DOI: 10.1056/NEJMoa2004500.

[14] *Broj kreveta intenzivne njege ključan je u borbi s koronom Evo kako Hrvatska stoji u tom pogledu u usporedbi s ostalim zemljama EU*. Ožujak 2020. URL: https://www.jutarnji.hr/vijesti/hrvatska/broj-kreveta-intenzivne-njege-kljucan-je-u-borbi-s-koronom-evo-kako-hrvatska-stoji-u-tom-pogledu-u-usporedbi-s-ostalim-zemljama-eu-10154305.

[15] D. K. Chu i dr. „Physical distancing, face masks, and eye protection to prevent person-to-person transmission of SARS-CoV-2 and COVID-19: a systematic review and meta-analysis". *The Lancet* (2020). DOI: 10.1016/s0140-6736(20)31142-9.

[16] *COVID-19 pandemic in South Korea*. Lipanj 2020. URL: https://en.wikipedia.org/wiki/COVID-19_pandemic_in_South_Korea.

[17] E. Edwards. *Family clusters: A common pattern for how the coronavirus spreads*. Ožujak 2020. URL: https://www.nbcnews.com/health/health-news/family-clusters-common-pattern-how-coronavirus-spreads-n1150646.

[18] *Herd immunity*. Lipanj 2020. URL: https://en.wikipedia.org/wiki/Herd_immunity.

[19] N. Lanese. *'Superspreader' in South Korea infects nearly 40 people with coronavirus*. Veljača 2020. URL: https://www.livescience.com/coronavirus-superspreader-south-korea-church.html.

[20] Y. Liu i dr. „The reproductive number of COVID-19 is higher compared to SARS coronavirus". *Journal of Travel Medicine* 27.2 (2020). DOI: 10.1093/jtm/taaa021.

[21] *Mathematical modelling of infectious disease*. Lipanj 2020. URL: https://en.wikipedia.org/wiki/Mathematical_modelling_of_infectious_disease.

[22] G. Onder, G. Rezza i S. Brusaferro. „Case-Fatality Rate and Characteristics of Patients Dying in Relation to COVID-19 in Italy". *Jama* (2020). DOI: 10.1001/jama.2020.4683.

[23] *Small-world experiment*. Travanj 2020. URL: https://en.wikipedia.org/wiki/Small-world_experiment.

[24] *Coronavirus Cases:* URL: https://www.worldometers.info/coronavirus/.

[25] *COVID-19 vs. previous pandemics*. URL: https://www.medicalnewstoday.com/articles/comparing-covid-19-with-previous-pandemics#Lessons-to-be-learned.

[26] *Hrvatske obitelji najbrojnije su u Europskoj uniji. Što nam to govori?* URL: https://www.tportal.hr/biznis/clanak/infografika-hrvatske-obitelji-najbrojnije-su-u-europskoj-uniji-sto-nam-to-govori-foto-20180604.

[27] *Population Pyramids of the World*. URL: https://www.populationpyramid.net/croatia/2019/.

[28] *U svijetu koronavirusa danas ih plaćaju suhim zlatom, ovi uređaji znače život: Hrvatska ima 800 respiratora, usporedili smo brojke s ostatkom Europe*. URL: https://slobodnadalmacija.hr/vijesti/hrvatska/u-svijetu-koronavirusa-danas-ih-placaju-suhim-zlatom-ovi-uredaji-znace-zivot-hrvatska-ima-800-respiratora-usporedili-smo-brojke-s-ostatkom-europe-1013607.

Sažetak

Tema ovog rada je analiza COVID-19 pandemije i njenih mogućih scenarija. Rad je podijeljen u tri logičke cjeline.

U prvoj cjelini (koja se sastoji od prvog i drugog poglavlja) spominju se i opisuju standardni epidemiološki modeli. Povijesno su se takvi modeli bazirali na običnim diferencijalnim jednadžbama poput SIR modela i njegovim varijantama. U zadnje vrijeme zbog sve većih računalnih kapaciteta, sve su češći u literaturi modeli bazirani na mrežama u kojima se svaka osoba poistovjećuje s jednim vrhom, a svaki kontakt među osobama s jednim bridom. Prednost takvih modela je očita jer omogućuje simulaciju stvarnih kretanja i kontakata osoba u populaciji.

U drugoj cjelini (koja se sastoji od trećeg poglavlja) detaljno se opisuje korišteni model u radu. Model je inovativan i baziran na jednostavnoj mreži u kojoj su eksplicitno ukomponirana kućanstva (dokazana kao glavni izvor širenja infekcije). Širenje bolesti po generalnoj populaciji ostvareno je preko posebnog slučajnog procesa koji uzima u obzir uvedene epidemiološke mjere i restrikcije na kretanja. U modelu se promatraju i ostale (respiratorne) bolesti i njihova interakcija s COVID-19.

U trećoj cjelini (koja se sastoji od četvrtog i petog poglavlja) prikazuju se rezultati simulacija. Osnovni korišteni princip je grafički prikaz značajki modela poput vrhunca ili duljine pandemije u ovisnosti o prostoru slobodnih parametara: parametar koji kontrolira strukturu grafa i parametri vezani za mobilnost i samoizolaciju osoba. Promatrane su i dvije ekstenzije osnovnog modela koje uvode specijalizirane jedinke čije ponašanje odskače od ostatka. Parametri modela kao i svi zaključci prilagođeni su za Republiku Hrvatsku.

Ključne riječi: COVID-19, model, simulacija, slobodan parametar, kolaps zdravstvenog sustava.

Summary

The topic of this work is the analysis of the COVID-19 pandemic and its possible scenarios. The work is divided into three logical parts.

In the first part (which consists of chapters one and two) the standard epidemiological models are described. Historically those models were based on ordinary differential equation like the SIR model and its variants. Lately, because of more available computing power, models based on graphs were each person is represented by a vertex and each contact between people by an edge are more and more common in literature. This approach has an obvious advantage because it is able to simulate real movements and contacts between people in a population.

In the second part (which consists of chapter three) the adopted model is described in detail. The model is innovative and based on a simple graph which explicitly implements the households (proved to be the main source of spread of the infection). The spread across the general population is accomplished by a particular random process which takes into account the adopted epidemiological measures and restrictions on movement. The model also considers other (respiratory) diseases and their interaction with COVID-19.

In the third part (which consists of chapter four and five) the results of the simulations are shown. The basic method used to convey this information is plotting the features of the model such as peak or length of the pandemic against the space of free parameters: one parameter which controls the structure of the graph and two parameters related to people's mobility and self-isolation. Two extensions are also introduced which include specialized individuals whose behaviour is different from the rest. The parameters of the model and all the conclusions are adapted to the Republic of Croatia.

Key words: COVID-19, model, simulation, free parameter, health system collapse.

Životopis

Rođen sam 23. rujna 1994. u Zagrebu. Godine 2001. preselio sam se s obitelji u Italiju, u Milano, gdje sam završio osnovnu školu (Istituto Comprensivo "Ilaria Alpi") i prirodoslovno-matematičku gimnaziju (Liceo Scientifico "Elio Vittorini"). Tijekom gimnazijskih dana doživio sam pomak u interesima s društvenih na prirodne znanosti te sam od 3. razreda sudjelovao na brojnim natjecanjima iz matematike i fizike s nekoliko visokih plasmana na školskoj i županijskoj razini a posebno bih istaknuo 10. mjesto na državnom natjecanju iz matematike u 4. razredu.

Godine 2014. vratio sam se u Zagreb i upisao preddiplomski studij Matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta. Titulu sveučilišnog prvostupnika stekao sam 2017. godine te upisao diplomski studij *Matematička Statistika* na istom fakultetu, te od iste godine primam i stipendiju za izvrsnost Privredne banke Zagreb.