CHALMERS

GÖTEBORGS UNIVERSITET

Stokastisk modellering av smittspridning inom sjukvården med nätverksmodeller

Stochastic Modelling of Disease Transmission Within Healthcare using Network Models

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Harry Chu
Elin Esborn
Rohan Jamieson
Hugo Johansson

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Stokastisk modellering av smittspridning inom sjukvården med nätverksmodeller

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers

Harry Chu Rohan Jamieson Hugo Johansson

Kandidatarbete i matematik inom civilingenjörsprogrammet Kemiteknik vid Chalmers

Elin Esborn

Handledare: Philip Gerlee Matematiska vetenskaper

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Förord

Gruppens arbetsgång

Tidigt i arbetet läste hela gruppen in sig på ämnet och diskuterade i vilket riktning projektet skulle ta. Efter att beslutet togs att det skulle bli programmeringstungt delades arbetet till en början upp. Med hjälp av GitHub kunde vi se till att alla kunde ta del av samt modifiera alla ingående skript. Gruppen möttes regelbundet varje vecka, både med och utan handledare. Vi hade även en presentation under mitt-mötet för avstämning. Rapporten skrevs kontinuerligt under arbetets gång. Vi vill även tacka vår handledare Philip Gerlee som väglett oss från början till slut.

Individens arbetsgång

Harry: Arbetade med utveckling av koden till simulering i början vilket resulterade i utvecklingen av en enklare tidsdiskret modell. Utvecklade också vidare på Gillespie algoritmen och undersökte metoder för att ha med Gammafördelade tider i algoritmen. Jobbade sedan mest med implementation av screening, kontaktspårning och utvärdering av dessa metoder.

Elin: I början av arbetet behövde jag skola in mig i språket Python med VS code. Därefter var jag delaktig i grunderna vid skapandet av grafen, mer specifikt den första strukturen och därtill rummen tillhörande grafen. För modellen krävdes därefter litteraturundersökning för parametervärden, grafstruktur och möjliga interventioner vilket hela gruppen fick lägga tid på. Parameterskattningsmetoder undersöktes även här. Efter interventionerna hade bestämts utvecklade jag skriptet för handhygiensinterventionen. En av vägarna jag initialt ville gå var att smittspridningshastigheterna skulle bero på en dos-respons kurva. Idén användes inte efter diskussion med handledare, och metoden som presenteras i rapporten utfördes istället. För att se vad jag har producerat i skrift, se Tabell 1. För programmeringen användes AI-verktyg som stöd för tolkning, utveckling och felsökning av kod.

Rohan: Jag började arbetet med koden för att skapa grafen tillsammans med Elin. Sedan jobbade jag på att optimera körningstid på simuleringen med hjälp av profilingpaket i Python. Sedan började jag skriva på sprittspridningsdynamiken i teoridelen. Medan det lärde jag mig om reproduktionstalet och implementerade både en teoretisk beräkning på R_0 och simuleringar för att approximera R_0. Teoretiska beräkningen kom inte med i finalrapporten då den inte kunde ta hänsyn till screening. Jag undersökte också hur kontaktnätverket skulle vara byggd för att vara realistisk. Under tiden hjälpte jag också till andra med problem i koden och med Python och Git. Sedan skrev jag mycket av sammandraget och populärvetenskapliga texten. Jag använde inte AI i programmeringen eller skrivandet.

Hugo: Under arbetet har jag jobbat mycket med att utveckla koden för bland annat Gillespie-algoritmen som vi använder för att göra de stokastiska simuleringarna. Detta arbete inkluderade till en början att ta fram flertalet prototyper, som bland annat en tidsdiskret modell som föregick Gillespie-algoritmen. Efter den tidsdiskreta modellen skrev jag koden för vår första Gillespie-algoritm, som byggde på en homogen population med flöde. I detta arbete ingick även att skapa figurer för visualisering av de olika resultaten och att skapa ett första nätverk för simulering på. Implementationen av Gillespie-algoritmen har även inkluderat att sätta mig in i hur den fungerar och att undersöka möjligheterna att använda en mer generaliserad Gillespie-algoritm som bygger på andra fördelningar än exponentialfördelningen. På grund av att simuleringarna började ta lång tid att köra för större nätverk arbetade jag även med att skriva effektiv kod och att parallellisera den. Genom parallelliseringen kunde vi dels köra simuleringarna snabbare på våra egna datorer, men det visade sig också särskilt gynnsamt när vi började köra simuleringarna på Gantenbein som har ett stort antal processorer. När interventionerna hade implementerats arbetade jag även med att beräkna resultaten från simuleringarna, som det genomsnittliga antalet infekterade, och att visualisera det i olika figurer. Efter detta gick mycket tid till att implementera koden för grafen som användes för utvärderingen av olika typer av screeningmetoder, vilket var ett relativt stort projekt. Jag arbetade även med att implementera versionen av screening som bygger på kontaktspårning av grannar, genom att använda bredd-först-sökning. De sista veckorna spenderade jag

mycket tid på att felsöka diverse fel i resultaten och att producera ett stort antal resultat och figurer för rapporten.

Populärvetenskaplig presentation	Rohan
Sammandrag	Rohan, Hugo, Elin
Inledning	Elin
Teori - Smittspridningsdynamik	Rohan, Harry
Teori - Gillespie-algoritmen	Hugo
Teori - Nätverksmodeller	Rohan, Elin
Teori - Modeller för sjukhussmitta	Elin
Teori - Parametervärden	Harry
Metod - Nätverket	Rohan, Elin
Metod - Interventioner - Screening	Hugo, Harry
Metod - Interventioner - Handhygien	Elin
Metod - Implementation av Gillespie-algoritmen	Hugo
Metod - Utvärdering av effekter från interventioner	Hugo
Metod - Hantering av slump	Harry, Hugo
Resultat - Simulering utan interventioner	Rohan
Resultat - Simulering med interventionen handhygien	Elin, Hugo
Resultat - Utvärdering av kontaktspårningsmetoder	Hugo
Diskussion - R_0 och z_{250}	Rohan, Elin
Diskussion - Simulering utan interventioner	Elin
Diskussion - Handhygien	Elin, Hugo
Diskussion - Screening	Harry, Hugo
Diskussion - Begränsningar och framtida hänvisningar	Rohan
Samhälleliga och etiska aspekter	Elin

Tabell 1: Del av texten på vänster, vilka som har skrivit på den till höger.

Populärvetenskaplig presentation

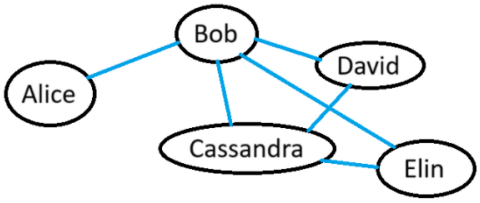
När man får en förkylning, och är frisk, så kan man ta det lugnt i några dagar, och sedan är man frisk igen. Men så är det inte för vissa av de som är inlagda på sjukhus. För dem så kan infektioner vara livsfarliga.

Ett exempel på en sådan infektion är MRSA, *Methicillin-resistant Staphylococcus aureus*, som kan hittas inom sjukvården. Att kunna förstå hur denna infektion sprids inom ett sjukhus och hur man kan förebygga att den gör så är viktigt för att kunna rädda liv. I detta arbete använder vi matematiska modeller för att simulera och utvärdera förebyggande strategier för att förhindra och avsluta utbrott av MRSA i sjukhus.

Matematisk modellering är en metod för att kunna förstå komplexa system. Man börjar med att göra antaganden om hur ett system fungerar, som förenklar systemet. Dessa ska vara verklighetstrogna men också enkla nog för att kunna skapa en modell. Ett viktigt antagande i detta arbete är att vi använder ett kontaktnätverk för att veta vilka individer som träffas och möjligen infekterar andra individer.

Ett exempel på en nätverk är ett vänskapsnätverk på Facebook. Tänk att varje person, så kallade noder, kan vara vän med flera andra personer. Om två personer är vänner med varandra så säger vi att det finns en kant mellan dem i nätverket. Med nätverket kan man då se hur en kattvideo, som i arbetets fall är en smitta, har möjlighet att sprida sig i nätverket. Tillsammans med kontaktnätverket har vi också antaganden om hur en smitta sprids i nätverket. Här tillkommer stokastiska delen av modellen. Tiden det tar för en person att bli smittad av en granne är slumpmässig. Med dessa antaganden skapar vi en modell som vi sedan använder för att simulera smittspridning med ett datorprogram. En simulering ger då ett utfall för hur smittspridningen kan ske. En kombination av flera simuleringar ger viktig statistik för den allmänna spridningen, såsom medelvärden på hur många personer som har blivit smittade efter en viss mängd tid, och hur många personer som en enstaka smittad individ smittar innan de blir friska.

Med modellen och programmet kan strategier för att förhindra smittspridning provas. En förbättrad handhygien, som innebär att smittan har svårare att spridas mellan personer, gav i våra simuleringar en tydlig förbättring genom att minska antalet smittade och sjukdomens smittsamhet. Även att vid regelbundna tider testa slumpmässigt utvalda personer för att se om de är infekterade och isåfall isolera dem gav ett liknande resultat. Med hjälp av kontaktnätverket kan vi prioritera testning av grannar till kända infekterade personer, som visade sig ge bättre resultat än när testningen var slumpad.



Figur 1: Ett exempel på en Facebook vänskapsnätverk. De svarta cirklarna är noder, och de blåa linjerna är kanter. Vi ser att Bob är vänner med alla och att Alice bara har Bob som kompis.

Sammandrag

Stokastisk modellering av sjukhusinfektioner är ett forskningsområde som har fått mer uppmärksamhet på senaste tiden. Tidigare arbeten har använt kontaktnätverk för att bestämma hur infektioner kan sprida sig i ett sjukhus [1]. Andra har utvecklat simulationsalgoritmer för sjukhus med in-och utflöde av patienter.

I detta arbete har stokastiska simulationer av *Methicillin-resistant Staphylococcus aureus* utförts på en nätverksmodell med patientflöde, där patienterna ges realistiska inskrivningsperioder på sjukhuset.

Interventioner har implementerats för att undersöka om det sker en minskningen av antalet infekterade och om reproduktionstalet påverkas. Resultaten visar på att både intervention handhygien och screening lyckas reducera smittsamheten av sjukhusinfektioner. Screeningparametrar som fördröjning av resultat och antalet test undersöktes. Resultatet visar att det är fördelaktigt att utföra ett större antal tester med ett större intervall mellan testningen i jämförelse med att testa färre personer oftare om totala antalet test hålls konstant. Dessutom gav screening baserad på närhet till nyliga fall, kontaktspårning, ökad effektivitet jämfört med att screena slumpmässiga individer.

Abstract

Stochastic modelling of nosocomial infections is a research area that has received more attention recently. Earlier works have used contact networks to define how infections can spread in a hospital. Others have developed simulation algorithms for hospitals with a flow of patients entering and exiting the hospital.

In this report, stochastic simulations of the transmission of *Methicillin-resistant Staphylococcus aureus* have been conducted on a network model with patient flow, with patients being given realistic length of stays at the hospital.

Interventions have been implemented in order to investigate if a reduction of infected individuals occur and whether the basic reproduction number is affected. The results indicate that both the hand hygiene and screening interventions are successfully reducing transmission of the nosocomial infection. Screening parameters such as test result delay, number of tests, and the time interval between test, are considered. Our results show that it is more effective to test more individuals with a larger time interval between tests than to test fewer individuals with a shorter time interval, given that the total amount of tests stay the same. Moreover, basing the screening on the viscinity to previous cases, contact tracing, showed a greater efficacy than randomised screening.

Innehåll

1 Inledning

Multiresistenta bakterier är ett växande globalt problem, och särskilt drabbade är patienter på sjukhus [2]. Nosokomiala sjukdomar (vårdrelaterade infektioner) är ett större hot mot patienter än den övriga populationen då de är försvagade och därav mer mottagliga. Bakterien *Methicillin-resistant Staphylococcus aureus* är resistent mot flertalet antibiotika och är en av de vanligare förekommande infektionsorsaker som uppstår inom sjukvården [3]. Omkring var tredje person bär MRSA på huden och konsekvenserna av en sådan infektion för en försvagad individ kan vara dödlig [4]. Antibiotikaresistensen uppstår av slumpmässiga mutationer i genomet av bakterier som ger dem en fenotypisk fördel. Med andra ord är de mer benägna att överleva i en miljö som annars hade varit dödlig [5]. Olika klasser av antibiotika har olika verkningsmekanismer och är effektiva mot särskilda bakterier. I modellen som presenteras undersöks hur interventioner påverkar smittspridning av bakteriefamiljen *Methicillin resistant Staphylococcus aureus* (MRSA). Familjen karaktäriseras av deras resistens mot beta-laktam antibiotika, den mest kliniskt använda antibiotikan vilken inhiberar bakteriernas cellväggssyntes [6, 7].

Inom sjukvården är modellering ett verktyg som kan användas för att öka förståelsen för hur smittor sprider sig. Ett sätt att tänka kring smittvägar är genom interaktioner som i sin tur kan representeras av kanter i nätverksmodeller. Ett citat av Statistikern George Box lyder *"Alla modeller är fel, men vissa är användbara"*. Med det sagt är den presenterade modellen en approximation av verkligheten i form av ett nätverk. Där representeras varje individ av en nod vilket kan skapa möjlighet för smittspårning. Exempelvis användes nätverksmodeller för att spåra COVID-19 i populationer och undersöka vilken karantänstrategi som hade störst effekt [8].

Eftersom nätverksmodeller är lämpliga för smittspridningsmodellering finns mängder av artiklar i litteraturen som utvärderar olika interventioners effektivitet i vården. En vanligt förekommande intervention är justeringen av antalet kontakter för läkare eftersom de är källor till hög spridning då de både kan röra sig mellan olika avdelningar och har många interaktioner med andra. Det görs exempelvis genom att dela upp läkarna i mindre lag eller att tilldela färre patientplatser per läkare, precis som i studien av Ueno et al [9].

Screening av patienter och sjukvårdspersonal för MRSA för att i sin tur isolera positiva fall är en annan strategi som visat sig effektiv i vården [10]. I praktiken innebär screening att ta prov, vanligtvis från slemhinnor då MRSA undersöks, för att avgöra om individen bär på bakterien [11]. För att välja typen av test vägs faktorer mot varandra som känslighet, tid, ekonomi, och frekvensen av falska positiva test. Det sistnämnda innebär att testet indikerar att individen är infekterad trots att så inte är fallet. Därefter behövs en strategi för hur testerna bör implementeras. I denna rapport undersöks strategier som beror på hur ofta screeningen bör tillämpas samt antalet individer som testas då strategin är slumpmässig testning eller kontaktspårning. Syftet med strategin är att minska antalet smittade så mycket som möjligt.

En ytterligare effektiv åtgärd för smittspridningsreducering är försök att öka följsamheten eller effektiviteten av handhygiensprocedurer. Exempelvis gjordes en studie av Pitet et al. i vilket en kampanj introducerades som innehöll påminnande affischer, handsprit i fickformat, samt handspritsflaskor på patienternas sängar [12]. Att använda handsprit i jämförelse med tvål har upprepade gånger visat sig minska antalet smittofall, och därför undersöks här hur stor effekt ett sådant produktbyte har i vår modell.

Den här rapporten presenterar en modell som gör det möjligt att undersöka effekterna av interventioner av MRSA smitta i sjukhusmiljö. Grafens storlek och struktur bygger på den presenterad i Rocha et al. för att avdelningar och rum ska vara verklighetsbaserade [13]. Till skillnad från redan existerande nätverksmodeller av sjukhus utnyttjas Gillespiealgoritmen med gammafördelningen för patienternas vistelsetid [14]. Med modellen utvärderas effekterna av interventionerna screening och handhygien.

1.1 Syfte

Arbetets syfte är att skapa en nätverksbaserad modell för smittspridning på sjukhus baserad på Gillespie-algoritmen men med en gammafördelning för vistelsetiden för patienter. Vidare ska interventioner motsvarande kontaktspårning och förbättrad handhygien utvärderas för sin förmåga att minska smittspridningen.

2 Teori

2.1 Smittspridningsdynamik

Matematisk modellering av smittspridning handlar om att först definiera en tillräckligt verklighetstrogen modell och sedan lära sig om modellen och dess egenskaper. Det finns olika varianter på modeller, till exempel deterministiska eller stokastiska; de kan beskriva en öppen eller sluten population; och även diskret tid i generationer eller kontinuerlig tid.

I en stängd population kan man vilja svara på om en infektion leder till ett utbrott [15]. Detta åstadkoms genom att definiera

$$R_0 = \text{Väntevärdet på antalet sekundära smittfall per primärt smittfall i en population med endast en smittad individ.}$$

Följande är ett beräkningsexempel som visar hur R_0 kan beräknas analytiskt för enkla modeller.

Beräkningsexempel

Vi modellerar en smitta i en stängd population deterministiskt och i diskreta tidsteg. Vi använder en SIR modell, som innebär att individer kan befinna sig i en av tre fack: mottagliga (S), smittsamma (I), och tillfrisknade (R). Vi antar vidare att individerna är oåtskiljbara förutom deras index, och att vid varje tidsteg finns det en sannolikhet p att en individ i har en kontakt med individ j som leder till att j skulle infektera i om j var smittad. En individ som blir smittad är smittsam i ett tidsteg, vilket innebär att antalet smittsamma är exakt lika med antalet friska individer som smittades av förra tidsstegets smittsamma. Detta leder till att antalet smittade vid ett tidssteg är binomialfördelat med parametrar $N = S_t$, $\rho = 1 - \mathbb{P}(\text{en frisk individ inte blev smittad}) = 1 - (1-p)^{I_t}$. Slutligen får vi då att vid varje tidssteg ändras antalet individer i varje fack enligt

$$\begin{aligned} I_{t+1} &= X_t, \\ S_{t+1} &= S_t - X_t \end{aligned}$$

där $X_t \sim \text{Bin}(S_t,\, 1-(1-p)^{I_t})$, $t \in \mathbb{Z}_{\geq 0}$. Uttrycket för antalet tillfrisknade R fås från att $S+I+R$ är konstant och lika med totala antalet individer. Denna modell kallas Reed-Frost modellen [16]. För att få en deterministisk modell ersätter vi X med väntevärdet $\mathbb{E}[X]$. Då ges I_t, S_t av

$$I_{t+1} = S_t(1-(1-p)^{I_t}), \tag{1}$$
$$S_{t+1} = S_t(1-p)^{I_t}. \tag{2}$$

Vi kan nu beräkna reproduktionstalet R_0 explicit genom att sätta $I_0 = 1$ och beräkna I_1:

$$R_0 = I_1 = S_0(1-(1-p)^{I_0}) = S_0(1-(1-p)^1) = pS_0 \qquad \square$$

För mer komplexa modeller som den som används i detta arbete är det inte möjligt att beräkna R_0 analytiskt som i Reed-Frost modellen. Däremot är det möjligt att simulera fram R_0 genom att skatta hur många sekundära fall som ett primärt fall orsakar. Sedan fås en skattning på R_0 från att ta medelvärdet över alla individer som börjar som primära fallet.

För deterministiska modeller med en stängd population bestämmer R_0 om infektionen blir en epidemi eller om den dör ut väldigt snabbt: om $R_0 < 1$ dör infektionen ut och om $R_0 > 1$ blir det en epidemi [15].

För stokastiska modeller har R_0 en annan innebörd än för deterministiska modeller. Om $R_0 > 1$ finns det ändå en positiv sannolikhet z_∞ att infektionen dör ut. R_0 kommer att användas som mått för smittsamhet för en infektion med en viss modell för att kunna jämföra olika interventioner.

Beräkningsexempel

Med stokastiska Reed-Frost modellen, är sannolikheten att den enstaka infekterade tillfrisknar utan att smitta andra är

$$\mathbb{P}(I_1 = 0) = \mathbb{P}(X_0 = 0) = \binom{S_0}{0}(1-(1-p)^1)^0(1-p)^{S_0} = (1-p)^{S_0},$$

ty X_0 är binomialfördelad. Då vet vi att $z_\infty > (1-p)^{S_0}$, då det också finns en positiv sannolikhet att infektionen dör ut vid två eller tre infekterade, och så vidare. □

I detta arbete används en stokastisk, öppen, kontinuerlig tid modell, som implementeras med en Markovprocess. Dessa används för modellering av slumpmässiga processer där nästkommande händelse endast beror på det nuvarande tillståndet [17]. Mängden av alla tillstånd S, fördelningen av starttillståndet Λ och hastighetsmatrisen Q definierar en tidskontinuerlig Markovprocess. Mängden S ges av ett diskret tillståndsrum som exempelvis kan representera antalet infekterade i ett system. I detta sammanhang har Λ liten betydelse då starttillståndet är deterministiskt. Matrisen Q är en $n \times n$-matris, där n är antalet tillstånd och där varje icke-diagonalt element $Q_{ij},\ i \neq j$ är icke-negativt och representerar hastigheten som processen övergår från tillstånd i till j. Diagonalelementen Q_{ii} definieras så att raderna summerar till 0. De icke-diagonala elementen representerar hastighet i den mening att både tiden T det tar för Markovprocessen att lämna tillstånd j är exponentialfördelad med parameter $\lambda = \sum_{j \neq i} Q_{ij}$ och att

$$\mathbb{P}(\text{nästa tillstånd är } j \mid \text{nuvarande tillstånd är } i) = \frac{Q_{ij}}{\sum_{j \neq i} Q_{ij}}.$$

Den här metoden är användbar då den kan modellera händelser i kontinuerlig tid och bevarar minneslösheten, alltså att sannolikheten för en övergång till ett annat tillstånd endast beror på det nuvarande tillståndet. Dessutom har vi matematiskt att tiden mellan övergångar T är exponentialfördelad och därav minneslös, alltså att

$$\mathbb{P}(T > t + s \mid T > s) = \mathbb{P}(T > t).$$

Detta tillåter hopp i tid som senare används i Gillespie-algoritmens implementation som beskrivs nedan.

Att populationen är öppen innebär att det finns ett inflöde och utflöde av individer. Detta gör så att en epidemi kan fortsätta att leva och inte dö ut, om tillräckligt med mottagliga individer flödar in i populationen. Man kan då få en jämvikt i antal smittade individer. Det innebär att antalet smittade individer varierar runt ett visst värde. Det varierar på grund av stokastiska effekter.

2.2 Gillespie-algoritmen

Gillespie-algoritmen erbjuder en metod för exakt simulering av stokastiska Markovprocesser [18]. Betrakta n händelser (i detta fall en överföring av smitta) $X_1, ..., X_n$ där tiden till dem är exponentialfördelade enligt $X_i \sim \text{Exp}(\lambda_i)$. Dessa hastigheter λ_i är vad som senare kallas smittspridningshastigheter. Låt vidare t_0 vara starttiden för simuleringen och $\mathbf{x}_0$ vara systemets tillstånd vid denna tid. Vi utnyttjar att slumpvariabeln $Y = \min(X_1, ..., X_n)$ därmed blir exponentialfördelad med parameter $\Lambda = \sum_{i=1}^{n} \lambda_i$. Tiden τ till nästa händelse kan därmed slumpas fram från Y. Sedan behöver även vilken händelse som sker vid τ slumpas fram, vilket görs från en likformig fördelning där varje händelse i inträffar med sannolikhet λ_i/Λ. Ett värde kan dras från denna fördelning enligt standardinversionsgenereringsmetoden genom att ta

$$j = \text{minsta heltalet som uppfyller } \sum_{k=1}^{j} \lambda_{i(\mathbf{x})} > r_2 \Lambda \tag{3}$$

där $r_2 \sim \text{U}(0,1)$. Låt även $\mathbf{v}_i$ vara förändringen av tillståndet $\mathbf{x}$ vid händelse i. Algoritmen utför de följande stegen

1. Initialisera systemet med värden $t = t_0$ och $\mathbf{x} = \mathbf{x}_0$
2. För tillstånd $\mathbf{x}$ och tid t, utvärdera λ_i samt summan Λ
3. Generera värden för τ och j genom att dra från Y respektive generera från (3)
4. Uppdatera systemets tillstånd enligt $t \leftarrow t + \tau$ och $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_j$
5. Spara $(\mathbf{x}, t)$. Gå till steg 1 eller avbryt simuleringen.

2.3 Nätverksmodeller

Nätverksmodeller grundar sig ursprungligen i grafteori där idén är att studera relationer mellan objekt representerat av en graf [19]. Metoderna kan appliceras på komplexa system som sociala, biologiska eller transportnätverk för nätverksanalys och refereras då till som nätverksmodeller [20]. Med tanke på dess vanliga förekomst finns variationer på modelltypen, där statiska/dynamiska grafer, homo-/heterogena noder, och gradfördelning är exempel på faktorer som tas i åtanke. Det sistnämnda, gradfördelning, beskriver hur kanterna är fördelade över alla noder. I extremerna finns så kallade skalfria nätverk där fördelningen är kraftigt skev (en minoritet av noderna har en majoritet av kanterna), och uniform fördelning i vilket noderna har lika många kanter. I detta projekt har konstruktionen av nätverket resulterat i en mer uniform gradfördelning. Med andra ord saknar nätverket superspridare vilka karaktäriseras av att ha signifikant fler kanter i nätverket eller en central lokalisering vilket gör att de orsakar signifikant fler sekundära infektioner [21].

2.4 Modeller för sjukhussmitta

Matematiska modeller av smittspridning är ett effektivt verktyg inom epidemiologin som används för att förstå sjukdomars utbredning och underlättar preventionsstrategier [22].

För en smittspridningsmodell på sjukhus krävs justeringar utifrån populationsbaserade modeller för att de ska vara användbara i praktiken. Till att börja med finns ett flöde då populationen är öppen vilket inte är lika vanligt för populationsbaserade modeller. För att beskriva sjukhusvistelsen används ibland en gammafördelning, som i Faddy et al [23]. Den används eftersom fördelningen passar datan, och dels för att sannolikhetstätheten kan maximeras efter att en viss tid har gått.

I sjukhusmodeller är en lämplig anpassning att inkludera heterogena noder. Vår modell använder inte individbaserad data, men noderna kategoriseras som patient eller sjukvårdspersonal vilket bidrar med viss heterogenitet. Kontaktnätverket är dessutom begränsat av sjukhusets uppbyggnad vilket gör systemet mer förutsägbarhet i jämförelse med modeller över större populationer där interaktioner vanligtvis är mer slumpmässiga [24]. Då flertalet patienter delar rum tillförs en grad av klustering, ett mått på hur noder i en graf grupperar sig, vilket bidrar till den mer organiserade strukturen [25].

2.5 Parametervärden

Olika parametrar som behöver beaktas under modellering av smittspridning med hjälp av Gillespies algoritm är infektionstakten som betecknas λ. Tolkningen av infektionstakten är det förväntade antalet smittor som sker per dag. En artikel om modellering av smittspridning av MRSA i nätverksmodeller använde till exempel värden av λ på runt 0.028/dag [26]. Då arbetet går ut på att modellera i sjukhusmiljöer behövs även parametrar som bestämmer hur länge patienterna stannar i sjukhuset. Tidigare forskning visar att dessa tider generellt följer en högerskev fördelning [23], och att de kan uppskattas med bland annat Gammafördelningar eller Log-normalfördelningar [14]. Det här kommer att användas i arbetet för att modellera flödet av patienter i sjukhuset, specifikt kommer gammafördelningen att användas.

3 Metod

3.1 Nätverket

För att skapa det riktade nätverket användes Python-modulen `networkx`. Att grafen är riktad innebär, för två noder A och B, att hastigheten som A smittar B kan skilja sig från hastigheten som B smittar A, då A respektive B är smittad. Detta är nödvändigt då infektionshastigheten inte är symmetrisk (patienter smittar sjukvårdspersonal med en annan hastighet än sjukvårdspersonal smittar patienter). För att effektivisera uppbyggnaden av strukturen på nätverket konstruerades rum med endast en typ av nod som alla sammankopplades. Nätverket G som används är en konstruktion baserad på beskrivningar av nätverket av Ueno och Masuda [1]. Det finns tio avdelningar där varje avdelning har åtta rum med fyra sängar. Tolv sjukvårdspersonal, uppdelade i två grupper på sex sjukhuspersonal, är kopplade till varje avdelning. Det finns även två personalrum med 30 sjukvårdspersonal vardera, där all sjukvårdspersonal i ett rum är kopplade med varandra. Sjukvårdspersonalgrupper som är kopplade till samma avdelning är i olika personalrum.

Nätverket	
Antal noder	380
Antal kanter	6540
Genomsnittsgrad på noder	17.211

Tabell 2: Egenskaper för nätverket G.

För att möjliggöra simulering på nätverket översattes det till en modifierad angränsningsmatris A där

$$A_{ij} = \begin{cases} \lambda_{ij} & \text{om nod } i \text{ och } j \text{ är grannar} \\ 0 & \text{annars} \end{cases}$$

och λ_{ij} är infektionshastigheten från nod i till nod j, som bestäms av nodernas respektive roller. Figur 2 är en enkel illustration av matrisen i Tabell 4 vilket i vårt fall har dimensionen 380×380. Detta definierar implicit matrisen Q för en stokastisk Markovprocess. Tillståndsmängden kan då ses som en mängd med binära vektorer som representerar vilka som är sjuka i nätverket med en etta och friska med en nolla.

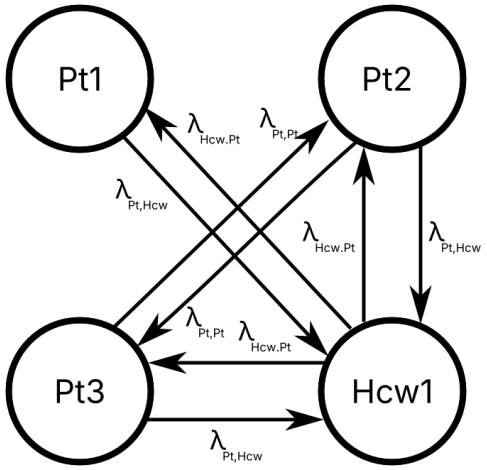
Smittspridningshastigheterna baseras på antalet kontakter per dag multiplicerat med sannolikheten att bli smittad per 30 sekunder. Värden för antalet kontakter kommer från en studie av Obadia et al. i vilken personalen och patienterna utrustades med trådlösa sensorer vilka registrerade en kontakt om distansen var under 1.5 meter [27]. Dessa värden visas i Tabell 3 och används i simuleringarna.

Parameter	Värde
$\lambda_{hcw,pt}$	$4.33 \cdot 10^{-3}/dag$
$\lambda_{hcw,hcw}$	$1.51 \cdot 10^{-3}/dag$
$\lambda_{pt,hcw}$	$1.07 \cdot 10^{-3}/dag$
$\lambda_{pt,pt}$	$1.8 \cdot 10^{-4}/dag$
Vistelsetid shape	3.5
Vistelsetid scale	2

Tabell 3: Värden på parametrar som användes i simuleringar utan interventioner. Index är av formen (smitta från, smitta till). Vistelsetiden beskrivs av en gammafördelning.

	Pt_1	Pt_2	Pt_3	Hcw_1
Pt_1	0	0	0	$\lambda_{Pt,Hcw}$
Pt_2	0	0	$\lambda_{Pt,Pt}$	$\lambda_{Pt,Hcw}$
Pt_3	0	$\lambda_{Pt,Pt}$	0	$\lambda_{Pt,Hcw}$
Hcw_1	$\lambda_{Hcw,Pt}$	$\lambda_{Hcw,Pt}$	$\lambda_{Hcw,Pt}$	0

Tabell 4: Diagram över hur en enkel modifierad angränsningsmatris A kan se ut.



Figur 2: Illustration av angränsningsmatrisen i Tabell 4 som ett nätverk.

3.2 Interventioner

Interventionerna som valdes var screening och handhygien. Dessa implementerades enskilt i modellen och beskrivs i detalj nedan.

3.2.1 Screening

Screening är en intervention som bygger på att det finns utrustning för att testa patienter eller sjukhuspersonal för en viss sjukdom. Testet ger ett positivt eller negativt resultat och baserat på resultatet kan den positivt testade individen isoleras från systemet vilket innebär att den inte kan bidra med ytterligare smitta och ersätts med en ny mottaglig individ. Med detta så introduceras faktorer och dynamik som kan analyseras. I arbetet betraktades fördröjning, vilket innebär att det dröjer en fixerad tid innan testresultaten kommer tillbaka och kan ageras på. Eftersom det kan finnas en begränsad tillgång till tester finns det även incitament till att utföra screening på det mest effektiva sättet.

Strategierna som undersöktes i arbetet var slumpmässig testning och kontaktspårning. Den slumpmässiga testningen gick ut på att i varje screeningomgång välja ut ett fixt antal individer slumpmässigt ur nätverket och utföra screening på dessa, sedan upprepa processen vid regelbundna tidsintervall. Viktiga parametrar som måste övervägas är hur många som ska testas och hur ofta testningen bör ske. Kontaktspårningen innebar att ett första test genomfördes med slumpmässigt urval, men att vid fortsatt testning utgick urvalet istället från de positivt testade individerna från det föregående testet. Detta urval rangordnade populationen på sjukhuset efter hur nära granne de är med någon av de som testats positivt och för individer på samma avstånd utnyttjades slumpmässigt urval för att välja det fixa antalet individer att testa. För att rangordna individerna på sjukhuset utifrån deras avstånd till de positivt testade användes en bredd-först-sökning med potentiellt flera startnoder (om flera föregående resultat var positiva) som grupperar noder efter avstånd. Om inga individer testade positivt vid den föregående omgången, eller om de som tes-

tade positivt hade blivit ersatta sedan dess, genomfördes en slumpmässig testning likt vid första omgången. I båda varianterna utfördes den första screeningen efter att sjukdomen hade fått fäste i populationen och nått ett jämviktsläge.

För att utvärdera hur effektiv en strategi för screening är jämförs andelen infekterade som testats med andelen av hela populationen som testas [26]. En testningsstrategi är alltså bättre om den har en lägre andel negativa test.

3.2.2 Handhygien

Folkhälsomyndigheten hävdar att handtvättning är den viktigaste åtgärden för att förhindra smittspridning [28]. Det finns även evidens för att handsprit har minst lika hög effektivitet som tvål när det kommer till avlägsnandet av bakterier från handflator [29]. För att undersöka effekten av ett sådant produktbyte implementerades parametervärden från en studie av Hornbeck et al [30]. Smittspridningshastigheterna i artikeln stämmer överens med dem rapporterade i Girou et al [29]. I studien av Girou et al. randomiserades personalen till grupper för tvål eller handsprit, och kontaminering på fingertoppar och handflator provtogs innan och efter rengöring. Medianreduceringen av bakterier uppmättes för de två metoderna. Hornbeck et al. rapporterar effektiviteten av handsprit och tvål för att reducera sannolikheten för smittspridning, vilket var 83 procent för handsprit och 58 procent för tvål [30]. På liknande sätt har värdena implementerats i vår modell för justering av smittspridningshastigheter. Smittspridningshastigheterna minskar därav proportionellt med handhygiensproceduren, och justeringen gäller för riktningen sjukvårdpersonal till patient eller mellan sjukvårdspersonal. Med metoden skapas en angränsningsmatris med likadan ursprungsmatris.

Till att börja med beräknas den procentuella förändringen som bytet från tvål till handsprit innebär på smittspridningen med följande beräkning.

$$\frac{0.83 - 0.58}{0.83} \approx 0.30$$

Därefter reduceras hastigheterna med förbättringen då riktningen är från sjukvårdspersonal.

$$\lambda_{\text{Hcw, Pt}} \cdot (1 - 0.30)$$

$$\lambda_{\text{Hcw, Hcw}} \cdot (1 - 0.30)$$

Det är på grund av att interventionen bara påverkar personalens beteende.

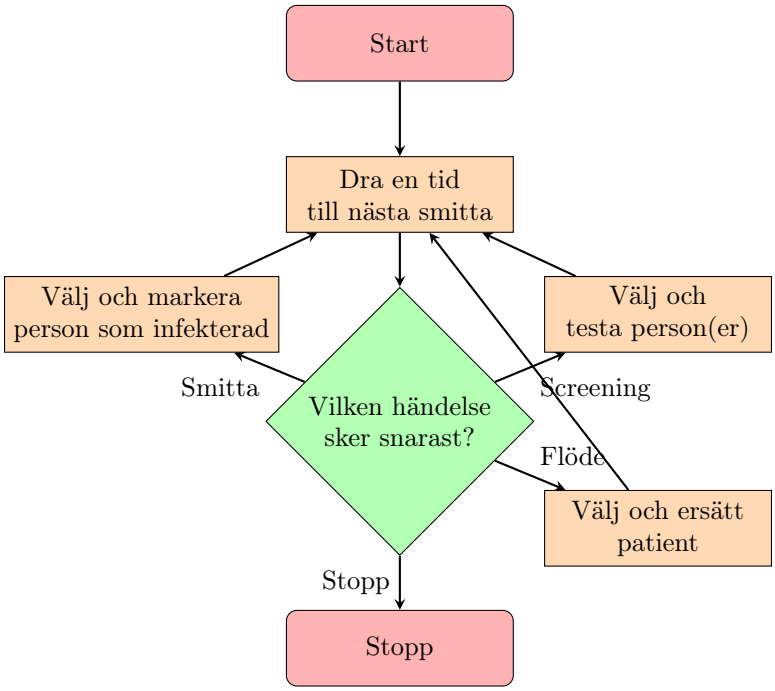
Antagandet om ett linjärt beroende mellan effektiviteten av handhygiensproceduren och smittspridningshastigheter behövde göras. Den beskrivna relationen som etablerats av Hornbeck et al. inkluderar däremot både sannolikheten att bli infekterad och personalens följsamhet. I vår modell antas följsamheten vara 100 procent och endast handhygienen påverkar sannolikheterna att bli smittad.

3.3 Implementation av Gillespie-algoritmen

För modellen användes en SI modell, en modifierad version av den mer etablerade SIR modellen, som bara innehåller de möjliga tillstånden Mottaglig (S) och Infekterad (I). Anledningen till att gruppen Tillfrisknad (R) saknas är för att kolonisationstiden är avsevärt mycket längre än vistelsetiden [31] och försummas därmed.

Modellen bygger på händelser, där varje händelse har en tid och en typ. Det finns tre olika typer av händelser: smitta, flöde och screening. Utöver dessa händelser kan även handhygien startas genom att angränsningsmatrisen ändras så att värdet på kanterna mellan patient och sjukvårdspersonal samt mellan två sjukvårdspersonal minskas permanent med en faktor.

- Smittohändelsen innebär att en mottaglig person har blivit smittad av en infekterad. När detta sker markeras denna nu smittade person som infekterad och bidrar i nästkommande iteration till smittspridning gentemot de friska individerna enligt de kanter som finns. Alltså aktiveras de utåtgående kanterna från personen samtidigt som de inåtgående inaktiveras. Det är denna del av simuleringen som utnyttjar Gillespie-algoritmen. Därmed är tiden för nästa smitta inte förutbestämd utan dras på nytt vid varje iteration och motsvarar att en smitta har skett över en okänd kontaktväg. Vilken denna kontaktväg är dras sedan när den här händelsetypen simuleras. Dessa parametrar kommer från angränsningsmatrisen.



Figur 3: Flödesschema för implementationen Gillespie-algoritmen.

- Flödeshändelsen innebär att en patient, oavsett tillstånd som mottaglig eller infekterad, blir utskriven och ersätts av en ny mottaglig person utifrån. Tiden för flödet dras inledningsvis från en gammafördelning vid simuleringens start. Den uppdateras sedan vid en flödeshändelse genom att ett ny vistelsetid dras från samma gammafördelning och adderas på den innevarande tiden.

- Screeningshändelsen innebär att ett specificerat antal personers infektionsstatus (om de är markerade som infekterade eller mottagliga) kollas. Beroende på om det är slumpmässig screening eller kontaktspårning anpassas metoden genom vilken urvalet av personer som testas sker. Närmare detaljer beskrivs under rubriken Screening. De som testas positivt isoleras med en fördröjning. Vid isoleringen introduceras omedelbart en ny mottaglig patient och en ny gammafördelad vistelsetid dras. Slutligen skapas en ny screeninghändelse som sker en fix tid framåt.

Vid varje iteration väljs alltså den tidigaste händelsen. Om stopptiden har passerats prioriteras den först. För att finna om flöde eller screening sker snarast kan deras tider jämföras för att hitta den minsta, eftersom dessa är kända. Tiden till nästa smitta är dock inte känd och måste därmed dras från exponentialfördelningen. Denna tid jämförs sedan med den minsta av de övriga tiderna för att se vilken som sker tidigast, varpå algoritmen går till den tiden. Om tiden för smitta inte är minst förkastas denna och en ny tid dras vid nästa iteration, vilket är möjligt genom minneslösheten. Se även flödesschemat i Figur 3.

3.4 Utvärdering av effekter från interventioner

För att utvärdera effekten av olika interventioner på smittspridningen användes flera mått.

1. Andelen infekterade efter att en stabil nivå har nåtts före interventionen påbörjats jämfördes med andelen infekterade efter att en stabil nivå har nåtts efter interventionen påbörjats. Detta beräknas med medelvärdesbildning.

2. Andelen simuleringar där smittan dör ut och inte tar fäste i populationen. Detta beräknas genom att ett stort antal simuleringar genomförs och andelen simuleringar där smittan dör ut innan en viss tidsgräns beräknas. I arbetet väljs tiden 250 och måttet kallas z_{250}.

3. Reproduktionstalet R_0' och R_0 beräknas med respektive utan interventionen för att utvärdera hur smittsamheten förändras. För att beräkna R_0 utförs simuleringar från varje nod i grafen där antalet infekterade efter 100 dagar, när endast den första noden är smittsam, beräknas. För att beräkna R_0' utfördes samma sorts simuleringar men med interventioner tillämpade. Värdet på R_0 och R_0' erhålles genom medelvärdesbildning av dessa resultat.

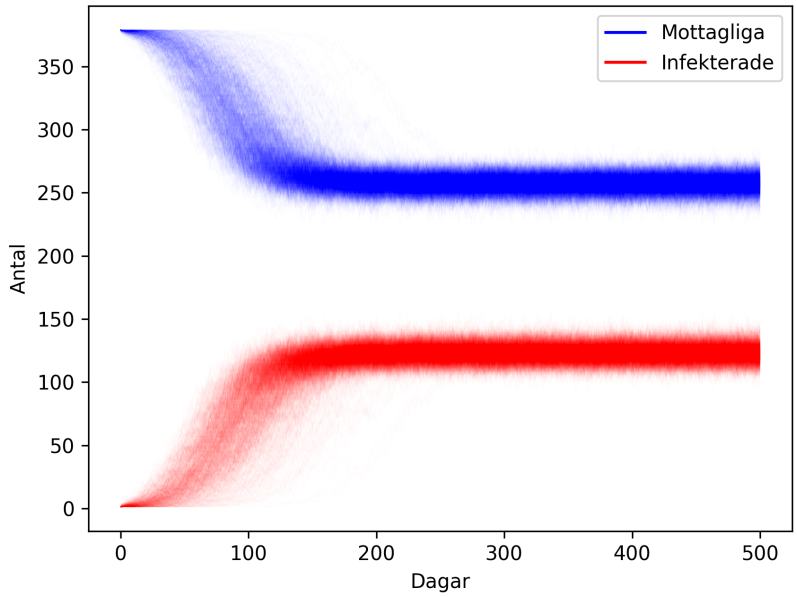
3.5 Hantering av slump

Eftersom simuleringarna involverar slump behövs upprepade iterationer av experimentet för att minska osäkerheten i resultatet genom medelvärdesbildning. Antalet simuleringar som gjordes var i storleksordningen 10^4-10^5. Att inte fler simuleringar gjordes berodde på begränsad tid och beräkningskraft. Under arbetet fanns tillgång till ett kluster med 112 processorer och koden parallelliserades för att göra simuleringar mer tidseffektiva. Parallelliseringen implementerades med en arbetarpool som hade samma antal arbetare som processorer och varje simulering lades till som ett arbete. Detta tillvägagångssätt var möjligt eftersom ett stort antal simuleringar kördes och alla simuleringar var oberoende av varandra. På så sätt fördelades alla simuleringar över de tillgängliga processorerna.

4 Resultat

4.1 Simulering utan interventioner

Smittspridning simulerades med modellen 10000 gånger utan interventioner aktiva med nätverket G från Tabell 2. Simuleringen börjar med en infekterad patient som är slumpmässigt vald, och körs i 500 dagar. De sista 50 dagarna var medelvärdet för antalet infekterade personer 122.8. I 91% av simuleringarna dog infektionen ut innan dag 250. Simulering av R_0 gav ett värde på cirka $R_0 = 5.02$.



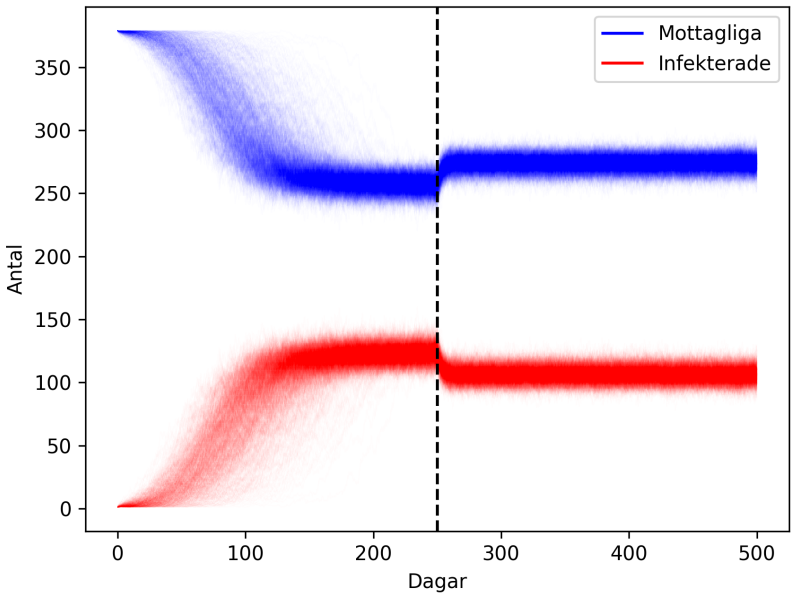
Figur 4: Graf med 10000 simuleringar utan interventioner.

4.2 Simulering med interventionen handhygien

Interventionen handhygien simulerades 10000 gånger över ett spann på 500 dagar med nätverket G från Tabell 2. Interventionen påbörjas efter att jämvikt ställt in sig vid tiden 250 dagar. Förändringen av jämviktsläget ger kunskap om hur stor effekt implementeringen haft på systemet då både antalet infekterade och mottagliga ställer in sig på ny jämvikt. Medelvärdet av antalet smittade de sista 50 dagarna innan interventionen påbörjas låg på 122.55 som i sin tur minskade till ett medelvärde på 106.09 de sista 50 dagarna av simuleringen. Skillnaden i antalet infekterade innan och efter interventionen är därav 13.43% vilket motsvarar det genomsnittliga antalet individer som inte blir smittade efter produktutbytet.

Reproduktionstalet R_0' simuleras från tiden noll med interventionen implementerad. Parametern ger nämligen information om det förväntade antalet grannar en individ smittar vid början av en epidemi. R_0' uppmättes till 3.56. Av alla simuleringar som gjordes dog smittan ut i 90% av fallen.

För att visa generaliserbarheten av interventionen har simuleringar med olika grad av effekten av interventionen gjorts, se tabell 5. Grafer över simuleringarna för dubbel och halv effekt kan hittas i Appendix A, Figur A.3 och A.4. Simuleringarna uppskattar hur ett sjukhus med likadan struktur hade reagerat om följsamheten från personalen varit lägre eller högre än utgångsläget eller om handspriten varit av annan effektivitet. För dubbel effekt minskade snittet av antalet infekterade individer de sista 50 dagarna med 28.46% och hade ett R_0' på 2.09. För halva effekten minskade snittet av antalet infekterade de sista 50 dagarna med 6.31% och hade ett genomsnittligt R_0' på 4.29.



Figur 5: Graf med 10000 simuleringar med interventionen handhygien

Faktor	Före	Efter
0.99	122.1	61.0
0.9	122.3	67.5
0.8	121.9	74.5
0.7	122.5	81.2
0.6	122.8	87.7
0.5	122.1	94.0
0.4	122.6	100.4
0.3	122.6	106.0
0.2	121.3	112.4
0.1	122.0	117.5

Tabell 5: Faktorn är den relativa minskningen av smittotakten och de andra kolonnerna visar det genomsnittliga antalet infekterade vid jämvikt före och efter interventionen med handhygien påbörjats. Alla simuleringar kördes 1000 gånger.

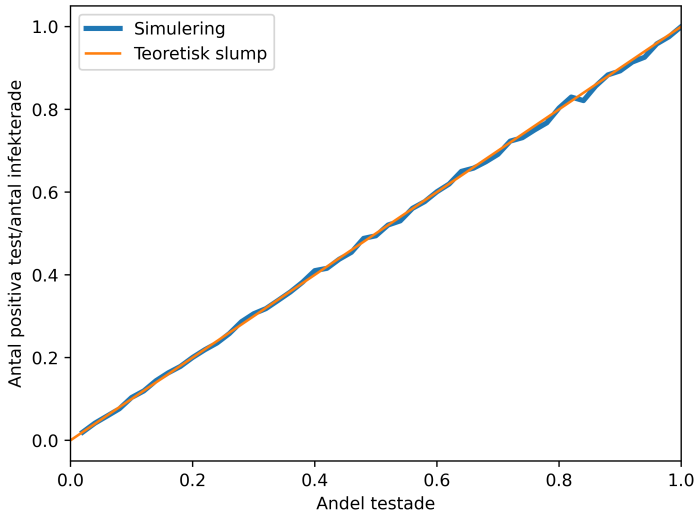
4.3 Utvärdering av kontaktspårningsmetoder

Dagar mellan testning / Antal test	1/2 dag	1 dag	2 dagar
10	1.01	1.82	2.86
20	0.53	1.01	1.84
40	0.27	0.53	1.01

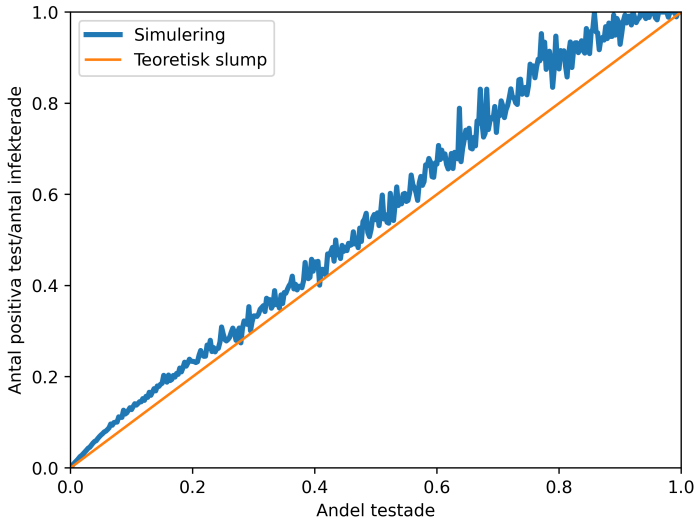
Tabell 6: R_0-värden för olika parametervärden på slumpmässig screening. Jämför med $R_0 = 5.00$ utan screening.

Dagar mellan testning / Antal test	1/2 dag	1 dag	2 dagar
10	0.97	1.74	2.77
20	0.50	0.93	1.68
40	0.26	0.48	0.88

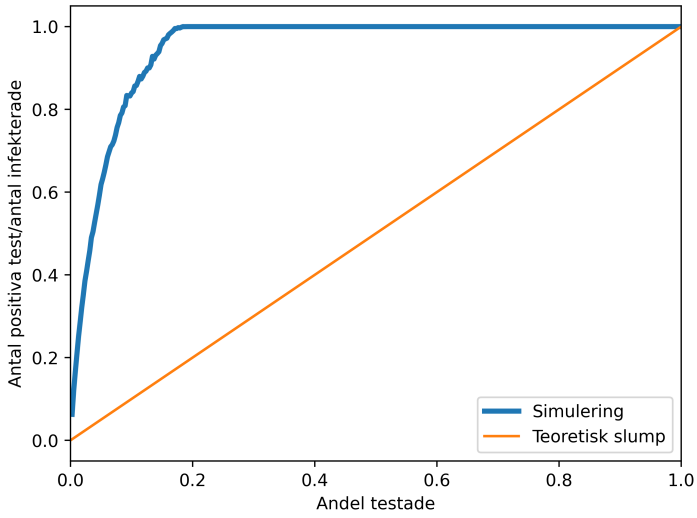
Tabell 7: R_0-värden för olika parametervärden på kontaktspårning. Jämför med $R_0 = 5.00$ utan screening.



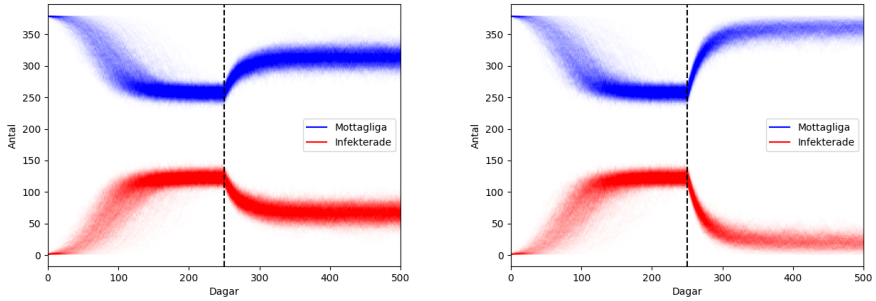
Figur 6: Screening med slumpmässigt urval varannan dag på den verklighetsbaserade grafen. Medelvärde av 100 000 simuleringar och teoretiskt resultat av slumpmässig screening.



Figur 7: Screening med kontaktspårning med testning varannan dag på den verklighetsbaserade grafen. Medelvärde av 100 000 simuleringar och teoretiskt resultat av slumpmässig screening.



Figur 8: Screening med kontaktspårning och perfekt urval. Medelvärde av 100 000 simuleringar och teoretiskt resultat av slumpmässigt urval.



(a) $\overline{I}_f = 122.42$, $\overline{I}_e = 65.26$, $z_{250} = 9134/10000$

(b) $\overline{I}_f = 122.20$, $\overline{I}_e = 15.79$, $z_{250} = 9208/10000$

Figur 9: Simuleringar med 10 tester med en dags mellanrum och a) slumpmässigt urval eller b) kontaktspårning. $\overline{I}_f$ = Medelvärde av infekterade 50 dagar innan start av intervention, $\overline{I}_e$ = Medelvärde av infekterade sista 50 dagarna, z_{250} = Andel utfall med inga utbrott innan start av intervention.

5 Diskussion

5.1 R_0 och z_{250}

I alla simuleringar, med och utan interventioner, dör en viss andel av simuleringarna ut. Det beskrivs med z_{250}, andel utfall med inga utbrott innan start av intervention. Exempelvis uppmättes z_{250} för simulering utan interventioner till 89%. Det beror på att infektionen börjar med att bara en patient är smittad vilken kan lämna avdelningen innan denna smittat någon annan. Beräkningarna för R_0 tar även med bidraget från sjukvårdspersonal eftersom de fortsätter att alltid vara smittsamma, och bidrar därav mycket till reproduktionstalet. Med det sagt är reproduktionstalet för endast patienter lägre än totala R_0. Det är anledningen till att många av våra simuleringar dör ut trots höga R_0.

5.2 Simulering utan interventioner

Spannet för reproduktionstalet R_0 är sjukdomsberoende där de minst smittsamma infektionerna kan ha värden nära noll och mycket smittsamma sjukdomar uppåt 30 [32]. I litteraturen är värden mellan 1 till 2 vanliga för MRSA, men det förekommer artiklar som rapporterat spann betydligt högre än så [33]. Exempelvis rapporterade Corkran et al. ett spann mellan 2.72-6.91 [34]. Med det sagt är det uppmätta reproduktionstalet från våra simuleringar på 5.02 högt, men inte orimligt.

5.3 Handhygien

Att byta från tvål till handsprit har en signifikant påverkan på jämvikten och parametern R_0. Resultatet indikerar en minskad sannolikhet för epidemi, men eftersom R_0' fortfarande är långt över 1 ökar smittan även efter interventionen men då långsammare. I Figur 5 kan man även se hur minskning av variansen, avvikelsen från medelvärdet för slumpvariabler, för båda grupperna Infekterade och Mottagliga inträffat. Den minskade spridningen är ett resultat av det minskade antalet infekterade patienter. Eftersom sjukvårdspersonal inte avkoloniseras efter infektion är antalet i vardera fack opåverkat vid beräkningen av snittantalet infekterade.

Resultaten i Tabell 5 uppvisar ett linjärt samband. Resultaten är något förvånande då denna typ av intervention upprepade gånger har visat sig följa lagen av avtagande avkastning [35]. Med andra ord är interventionen mer effektiv på sjukhus där handhygienen är sämre. Exempelvis gjordes en undersökning på smittspridning av en handhygiens interventionen vid olika grad av följsamhet av Hornbeck et al. Författarna kom fram till att den största förbättringen sker vid följsamhet under 20%, och vinsten avtar då följsamheten är hög [30].

Att interventionen i vår modell inte producerar ett sådant utfall kan bero på att all sjukvårdspersonal är smittad vid starten av interventionen och att vistelsetiden för patienterna är kort. Detta kan göra att smittspridningen mellan patienter får liten effekt på den totala smittspridningen, som istället till stor del utgörs av smitta från sjukvårdspersonal till patienter. Sjukvårdspersonalen har totalt sett även kontaktvägar till alla patienterna, vilket gör att smittan approximativt sprids från en homogen grupp av sjukvårdspersonal till en annan homogen grupp av patienter. Eftersom vistelsetiden även är kort beror därmed antalet infekterade patienter stark på smittotakten från sjukvårdspersonal till patienter. Därmed kan det anses rimligt att det finns ett linjärt samband mellan minskningen i denna smittotakt och antalet patienter som är infekterade.

5.4 Screening

Smittans egenskaper förändras tydligt när screening tillämpas, som framgår kvalitativt av figur 9, mot det önskade utfallet att färre är infekterade. Denna skillnad är också kvantitativt beskriven av skillnaden i genomsnittligt antal smittade före och efter interventionen påbörjas, som framgår av bildtexten i samma figur, och vidare av skillnaden i R_0' tal som återfinns i Tabell 6 och Tabell 7. Jämför exempelvis $R_0 = 5.00$ utan interventioner, med $R_0' = 1.01$ vid slumpmässig screening på 20 individer varje dag och $R_0' = 0.93$ för kontaktspårning på samma antal individer med samma frekvens. Detta tyder på att screening, för båda testade metoderna, bidrar till att färre individer på sjukhuset är infekterade och att sänka reproduktionstalet R_0'.

Värdena på R_0' som framgår av Tabell 6 tyder på att intervallet mellan testerna, så länge det genomsnittliga antalet tester per dag hålls konstant, inte påverkar effektiviteten av interventionen eftersom värdena inte förändras betydligt. Genom att betrakta diagonalerna i tabellerna kan värdena för olika strategier med samma totala antal tester jämföras. Detta överensstämmer med faktumet att alla slumpmässiga test är oberoende. Detta är i kontrast mot kontaktspårning, som framgår av Tabell 7, där vilka som testas beror på resultatet från det föregående testet. Dessa resultat tyder på att det är fördelaktigt att utföra ett större antal tester med ett större intervall mellan testningen i jämförelse med att testa färre personer oftare om totala antalet test hålls konstant. Denna slutsats kommer från att värdet på R_0' längs diagonalerna minskar.

Även figur 7 visar att kontaktspårning är marginellt bättre än slumpmässigt urval för den verklighetsbaserade grafen. Jämför med figur 8 som är en övre begränsning på hur bra screeningen kan bli då det visar scenariot där alla infekterade individer är kända och man kan välja ett perfekt urval. Att det inte blir större effekt beror antagligen på hur kontaktnätverket ser ut. På grund av att det inte är ett djupt nätverk, alltså att antalet kanter mellan två noder i nätverket inte är

så högt, så får kontaktspårning en liten effekt då metoden beter sig likt ett slumpmässigt urval. Simulationerna på ringgrafen, en graf specifikt konstruerad för att vara djup, i figur A.2 i Appendix visar att kontaktspårning får mycket högre effekt och presterar avsevärt bättre än slumpmässigt urval. Alltså varierar resultatet för kontaktspårning beroende på sammanhanget och om det ska ge effekt krävs ett djup på kontaktnätverket. Vidare simulerar vi endast kontakter som ger upphov till smitta och saknar de där överföring ej skedde. I en modell som simulerar alla kontakter kunde rangordningen av grannar ha skett endast för grannar en smittad individ har haft kontakt med, istället för att ta med alla potentiella kontakter (alla kanter).

Vidare kan det genomsnittliga antalet infekterade precis före en intervention påbörjas jämföras med det genomsnittliga antalet en tid efter att den startats. Figurer med tillhörande data över detta presenteras i Figur A.5, A.6 och A.7 i Appendix. Där framgår det att även sett till denna parameter så är kontaktspårning bättre än slumpmässigt urval. Vidare är denna skillnad avsevärt mycket större än de för R_0'.

5.5 Begränsningar och framtida utvidgningar

Projektet har följande begränsningar. Att smittspridningshastigheter inte var tillgängliga för både läkare och sjuksköterskor gjorde att dessa grupper slogs ihop till sjukvårdspersonal. Det är en begränsning då läkare är vanligt förekommande källor till hög spridning då de både kan röra sig mellan olika avdelningar och har många interaktioner. Kanterna i modellen tar också bara hänsyn till antalet interaktioner, men inte hur långa interaktionerna är då alla smittspridningshastigheter är sannolikheter per 30 sekunder. Att interagera med en smittad individ i längre tid ökar sannolikheten att bli smittad. Kanternas vikter är inte heller anpassade på individnivå, vilket innebär att personliga riskfaktorer ej tas i hänsyn. Beroende på sjukdomstillstånd kan en person bli infekterad med större sannolikhet än en annan. Sistnämnt ska nämnas att inkubationstiden för smittan MRSA inte beaktas i modellen utan att en smittad individ blir smittsam direkt.

Framtida studier som bygger på den presenterade nätverksmodellen för nosokomiala sjukdomar skulle kunna fokusera på de ekonomiska aspekterna av interventionerna. Baserat på resultaten i denna rapport finns potential för att minska utgifter om screeningmetoden används i optimal frekvens och mängd. En ytterligare utveckling hade varit att ta hänsyn till den infektiva perioden i vilket en individ kan vara smittsam.

En annan riktning är att variera vistelsetidens distribution då simuleringsalgoritmen tillåter den att vara allmän. Då kan sensitiviteten på resultaten med avseende på parametrar till vistelsetiden eller själva distributionen undersökas.

Sistnämnt är tillgång till patientdata, avdelningsstruktur, och en uppskattning på vistelsedistributionen från samma sjukhus förutsättningar för att ta beslut gällande dessa avdelningar.

6 Samhälleliga och etiska aspekter

Med studier relaterade till sjukvården är samhälleliga och etiska aspekter en central punkt eftersom det frekvent förekommer scenarion då de grundläggande etiska principerna måste vägas mot varandra. De grundläggande principerna innefattar *"göra gott, inte göra skada, vara rättvis, respektera självbestämmande och integritet"* [36].

Med resultaten från denna rapport finns mer underlag till att förebygga spridningen av nosokomiala sjukdomar vilket i sin tur skulle minska dödlighet och lidande. Däremot krävs datainsamling från både patienter och sjukvårdspersonal vilket medför risken att göra intrång på den personliga integriteten. Datainsamlingen för screeningmetoden är nämligen personlig information för vilket patienter behöver informeras om dess insamling, användning och delning. Detta följer lagstiftning av dataskyddsförordningen (GDPR) [37] som reglerar verksamheter som hanterar personuppgifter inom medlemsländer i Europeiska Unionen.

Det ska även tas i åtanke att modellen är en approximering av verkligheten med flertalet förenklingar. Därav ska slutsatser tolkas försiktigt innan åtgärder tillämpas i vården. Exempelvis hade ett högt R_0 värde påvisat att en epidemi utlöses, men att agera som så hade kunnat göra mer skada än gott för vårdkvalitén.

För vårdpersonalen kan projektet vara en ytterligare påfrestning på självbestämmandet. När en auktoritet tar beslut över hur screening implementeras på arbetsplatsen eller hur personalen

ska utföra handhygiensprocedurer kan det skapa obekväma känslor då gamla rutiner förändras. Omställningen kan vara energikrävande och förvirrande, och därför är det viktigt att personalen utbildas i varför interventionerna sannolikt minskar smittspridningen på lång sikt.

7 Referenser

Referenser

[1] Taro Ueno och Naoki Masuda. "Controlling nosocomial infection based on structure of hospital social networks". I: *Journal of Theoretical Biology* 254.3 (2008), s. 655–666. ISSN: 0022-5193. DOI: https://doi.org/10.1016/j.jtbi.2008.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0022519308003512.

[2] Luis Arturo Camacho Silvas. "[Bacterial resistance, a current crisis.]" spa. I: *Revista Espanola De Salud Publica* 97 (febr. 2023), e202302013. ISSN: 2173-9110.

[3] Clare Byrne m. fl. "The role of perioperative care in reducing rates of methicillin resistant Staphylococcus aureus". eng. I: *Journal of Perioperative Practice* 21.12 (dec. 2011), s. 410–417. ISSN: 1750-4589. DOI: 10.1177/175045891102101202.

[4] CDC. *Methicillin-resistant Staphylococcus aureus (MRSA) Basics*. en-us. Juni 2024. URL: https://www.cdc.gov/mrsa/about/index.html (hämtad 2025-02-13).

[5] N. Woodford och M. J. Ellington. "The emergence of antibiotic resistance by mutation". I: *Clinical Microbiology and Infection* 13.1 (jan. 2007), s. 5–18. ISSN: 1198-743X. DOI: 10.1111/j.1469-0691.2006.01492.x. URL: https://www.sciencedirect.com/science/article/pii/S1198743X14615500 (hämtad 2025-03-08).

[6] Harshad Lade och Jae-Seok Kim. "Molecular Determinants of β-Lactam Resistance in Methicillin-Resistant Staphylococcus aureus (MRSA): An Updated Review". I: *Antibiotics* 12.9 (aug. 2023), s. 1362. ISSN: 2079-6382. DOI: 10.3390/antibiotics12091362. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10525618/ (hämtad 2025-03-04).

[7] Jennifer Fishovitz m. fl. "Penicillin-binding protein 2a of methicillin-resistant *Staphylococcus aureus*". en. I: *IUBMB Life* 66.8 (aug. 2014), s. 572–577. ISSN: 1521-6543, 1521-6551. DOI: 10.1002/iub.1289. URL: https://iubmb.onlinelibrary.wiley.com/doi/10.1002/iub.1289 (hämtad 2025-02-27).

[8] Daniel Xu. "Modeling of network based digital contact tracing and testing strategies, including the pre-exposure notification system, for the COVID-19 pandemic". I: *Mathematical Biosciences* 338 (aug. 2021), s. 108645. ISSN: 0025-5564. DOI: 10.1016/j.mbs.2021.108645. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8214465/ (hämtad 2025-04-25).

[9] Taro Ueno och Naoki Masuda. "Controlling nosocomial infection based on structure of hospital social networks". I: *Journal of Theoretical Biology* 254.3 (okt. 2008), s. 655–666. ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2008.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0022519308003512 (hämtad 2025-05-07).

[10] Theodore Kypraios m. fl. "Assessing the role of undetected colonization and isolation precautions in reducing Methicillin-Resistant Staphylococcus aureustransmission in intensive care units". en. I: *BMC Infectious Diseases* 10.1 (febr. 2010), s. 29. ISSN: 1471-2334. DOI: 10.1186/1471-2334-10-29. URL: https://doi.org/10.1186/1471-2334-10-29 (hämtad 2025-04-21).

[11] G. L. French. "Methods for screening for methicillin-resistant *Staphylococcus aureus* carriage". I: *Clinical Microbiology and Infection* 15 (dec. 2009), s. 10–16. ISSN: 1198-743X. DOI: 10.1111/j.1469-0691.2009.03092.x. URL: https://www.sciencedirect.com/science/article/pii/S1198743X14606315 (hämtad 2025-04-25).

[12] D. Pittet m. fl. "Effectiveness of a hospital-wide programme to improve compliance with hand hygiene. Infection Control Programme". eng. I: *Lancet (London, England)* 356.9238 (okt. 2000), s. 1307–1312. ISSN: 0140-6736. DOI: 10.1016/s0140-6736(00)02814-2.

[13] Luis E. C. Rocha m. fl. "Dynamic contact networks of patients and MRSA spread in hospitals". en. I: *Scientific Reports* 10.1 (juni 2020), s. 9336. ISSN: 2045-2322. DOI: 10.1038/s41598-020-66270-9. URL: https://www.nature.com/articles/s41598-020-66270-9 (hämtad 2025-05-07).

[14] Alfio Marazzi m. fl. “Fitting the distributions of length of stay by parametric models”. I: *Medical care* 36.6 (1998), s. 915–927.

[15] Odo Diekmann, Hans Heesterbeek och Tom Britton. *Mathematical Tools for Understanding Infectious Disease Dynamics:* Princeton University Press, 2013. ISBN: 9780691155395. URL: http://www.jstor.org/stable/j.cttq9530 (hämtad 2025-02-23).

[16] A.D. Barbour och Sergey Utev. “Approximating the Reed–Frost epidemic process”. I: *Stochastic Processes and their Applications* 113.2 (2004), s. 173–197. ISSN: 0304-4149. DOI: https://doi.org/10.1016/j.spa.2004.03.013. URL: https://www.sciencedirect.com/science/article/pii/S0304414904000614.

[17] Umut Çetin och Albina Danilova. “Markov Processes”. en. I: *Dynamic Markov Bridges and Market Microstructure: Theory and Applications*. Utg. av Umut Çetin och Albina Danilova. New York, NY: Springer, 2018, s. 3–21. ISBN: 9781493988358. DOI: 10.1007/978-1-4939-8835-8_1. URL: https://doi.org/10.1007/978-1-4939-8835-8_1 (hämtad 2025-04-25).

[18] Daniel T Gillespie. “Stochastic simulation of chemical kinetics”. I: *Annu. Rev. Phys. Chem.* 58.1 (2007), s. 35–55.

[19] James DiFrisco, Johannes Jaeger och Andrea Loettgers. “Beyond Networks: Explaining Dynamics in the Natural and Social Sciences”. I: *Modeling the Possible*. Routledge, 2025. ISBN: 9781003342816.

[20] Sharmila Mary Arul m. fl. “Graph Theory and Algorithms for Network Analysis”. en. I: *E3S Web of Conferences* 399 (2023), s. 08002. ISSN: 2267-1242. DOI: 10.1051/e3sconf/202339908002. URL: https://www.e3s-conferences.org/articles/e3sconf/abs/2023/36/e3sconf_iconnect2023_08002/e3sconf_iconnect2023_08002.html (hämtad 2025-04-01).

[21] Brandon Lieberthal och Allison M. Gardner. “Connectivity, reproduction number, and mobility interact to determine communities’ epidemiological superspreader potential in a metapopulation network”. en. I: *PLOS Computational Biology* 17.3 (mars 2021), e1008674. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008674. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008674 (hämtad 2025-05-07).

[22] Sapana Yadav m. fl. *A Comprehensive Examination of Epidemiology: Its Terms, Types, and Methods*. English. Kap. Jan. 1. DOI: 10.4018/979-8-3693-2655-8.ch012. URL: https://www.igi-global.com/gateway/chapter/www.igi-global.com/gateway/chapter/351176 (hämtad 2025-02-22).

[23] Malcolm Faddy, Nicholas Graves och Anthony Pettitt. “Modeling length of stay in hospital and other right skewed data: comparison of phase-type, gamma and log-normal distributions”. I: *Value in Health* 12.2 (2009), s. 309–314.

[24] Collin M. McCabe och Charles L. Nunn. “Effective Network Size Predicted From Simulations of Pathogen Outbreaks Through Social Networks Provides a Novel Measure of Structure-Standardized Group Size”. English. I: *Frontiers in Veterinary Science* 5 (maj 2018). ISSN: 2297-1769. DOI: 10.3389/fvets.2018.00071. URL: https://www.frontiersin.orghttps://www.frontiersin.org/journals/veterinary-science/articles/10.3389/fvets.2018.00071/full (hämtad 2025-05-07).

[25] Oded Green och David A. Bader. “Faster Clustering Coefficient Using Vertex Covers”. I: *2013 International Conference on Social Computing*. Sept. 2013, s. 321–330. DOI: 10.1109/SocialCom.2013.51. URL: https://ieeexplore.ieee.org/document/6693348 (hämtad 2025-05-07).

[26] Sen Pei, Fredrik Liljeros och Jeffrey Shaman. “Identifying asymptomatic spreaders of antimicrobial-resistant pathogens in hospital settings”. I: *Proceedings of the National Academy of Sciences* 118.37 (2021), e2111190118.

[27] Thomas Obadia m. fl. “Detailed Contact Data and the Dissemination of Staphylococcus aureus in Hospitals”. I: *PLOS Computational Biology* 11.3 (mars 2015), s. 1–16. DOI: 10.1371/journal.pcbi.1004170. URL: https://doi.org/10.1371/journal.pcbi.1004170.

[28] *Om handhygien och handskar i vård och omsorg - Rena händer räddar liv*. sv. Mars 2022. URL: https://www.folkhalsomyndigheten.se/rena-hander-raddar-liv/handhygien-och-handskar/ (hämtad 2025-04-06).

[29] Emmanuelle Girou m. fl. "Efficacy of handrubbing with alcohol based solution versus standard handwashing with antiseptic soap: randomised clinical trial". I: *BMJ : British Medical Journal* 325.7360 (aug. 2002), s. 362. ISSN: 0959-8138. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC117885/ (hämtad 2025-04-06).

[30] Thomas Hornbeck m. fl. "On Hand Hygiene Compliance and Diminishing Marginal Returns: An Empirically-Driven Agent-Based Simulation Study". I: *Proceedings of the Computational Social Science Society of the Americas (CSSSA)*. Iowa City, IA, USA, 2011. URL: https://computationalsocialscience.org/wp-content/uploads/2011/10/CSSSA_hornbeck.pdf.

[31] Anna-Karin Larsson m. fl. "Duration of methicillin-resistant Staphylococcus aureus colonization after diagnosis: a four-year experience from southern Sweden". I: *Scandinavian journal of infectious diseases* 43.6-7 (2011), s. 456–462.

[32] Michiel Van Boven m. fl. "Estimation of measles vaccine efficacy and critical vaccination coverage in a highly vaccinated population". en. I: *Journal of The Royal Society Interface* 7.52 (nov. 2010), s. 1537–1544. ISSN: 1742-5689, 1742-5662. DOI: 10.1098/rsif.2010.0086. URL: https://royalsocietypublishing.org/doi/10.1098/rsif.2010.0086 (hämtad 2025-05-10).

[33] Mattia Prosperi m. fl. "Molecular Epidemiology of Community-Associated Methicillin-resistant Staphylococcus aureus in the genomic era: a Cross-Sectional Study". en. I: *Scientific Reports* 3.1 (maj 2013), s. 1902. ISSN: 2045-2322. DOI: 10.1038/srep01902. URL: https://www.nature.com/articles/srep01902 (hämtad 2025-06-04).

[34] Kiel Corkran m. fl. *Bayesian Inference of Nosocomial Methicillin-resistant Staphylococcus aureus Transmission Rates in an Urban Safety-Net Hospital*. en. Dec. 2024. DOI: 10.1101/2024.12.18.24319252. URL: https://www.medrxiv.org/content/10.1101/2024.12.18.24319252v1 (hämtad 2025-05-10).

[35] Clive B. Beggs, Simon J. Shepherd och Kevin G. Kerr. "Increasing the frequency of hand washing by healthcare workers does not lead to commensurate reductions in staphylococcal infection in a hospital ward". I: *BMC Infectious Diseases* 8.1 (sept. 2008), s. 114. ISSN: 1471-2334. DOI: 10.1186/1471-2334-8-114. URL: https://doi.org/10.1186/1471-2334-8-114 (hämtad 2025-05-13).

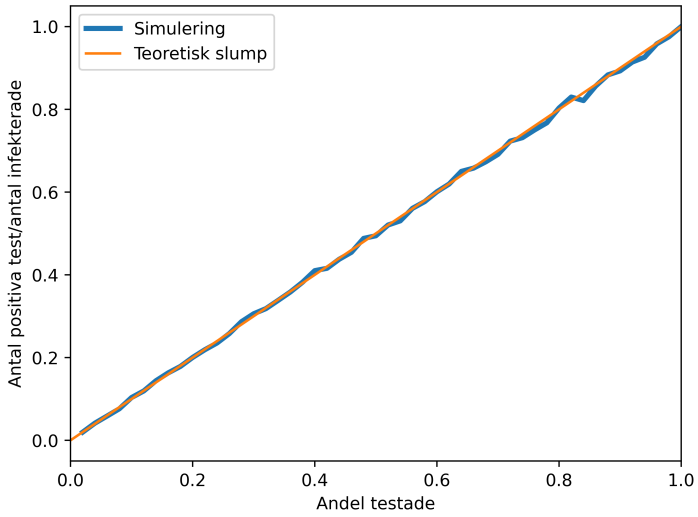
[36] Peter Herissone-Kelly. "Determining the common morality's norms in the sixth edition of Principles of Biomedical Ethics". en. I: *Journal of Medical Ethics* 37.10 (okt. 2011), s. 584–587. ISSN: 0306-6800, 1473-4257. DOI: 10.1136/jme.2009.030114. URL: https://jme.bmj.com/content/37/10/584 (hämtad 2025-05-08).

[37] Mary Donnelly och Maeve McDonagh. "Health Research, Consent and the GDPR Exemption". I: *European Journal of Health Law* 26.2 (2019), s. 97–119. DOI: 10.1163/15718093-12262427. URL: https://brill.com/view/journals/ejhl/26/2/article-p97_2.xml.

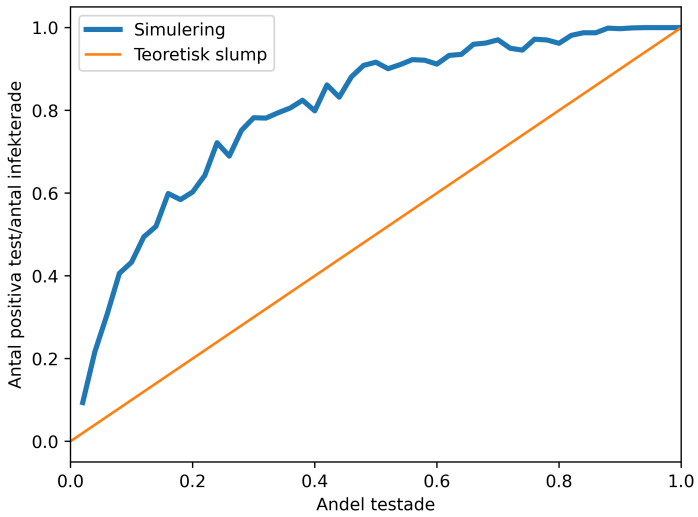
8 AI-användande

AI verktygen Microsoft Copilot och ChatGPT har utnyttjats som stöd i utvecklingen av vissa delar av koden. Scopus AI har även använts för sökning efter källor. Se bidragsrapport för tydligare beskrivning.

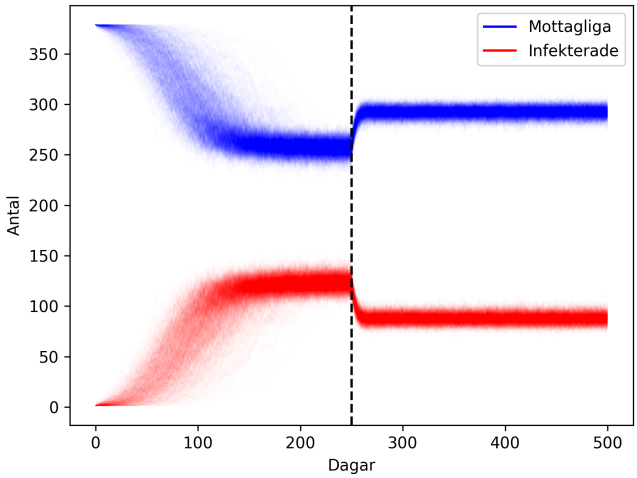
A Appendix 1 – teori



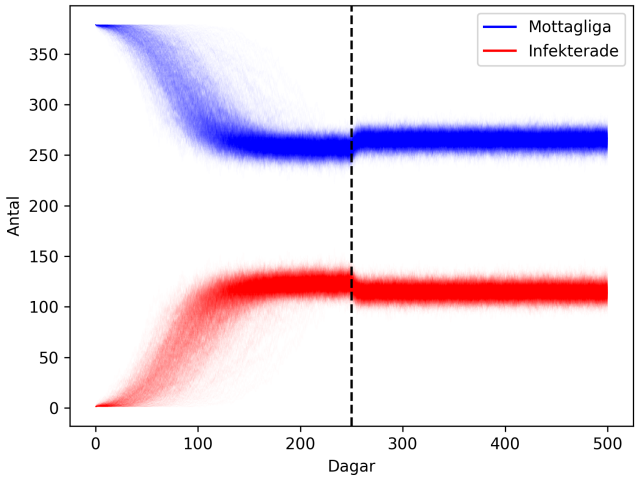
Figur A.1: Screening med slumpmässigt urval varannan dag på ringgraf. Medelvärde av 100 000 simuleringar och teoretiskt resultat av slumpmässig screening.



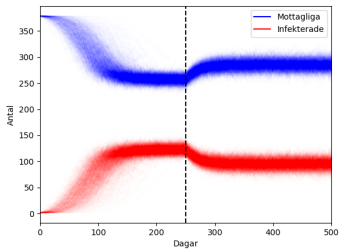
Figur A.2: Screening med kontaktspårning med testning varannan dag på ringgraf. Medelvärde av 100 000 simuleringar och teoretiskt resultat av slumpmässig screening.



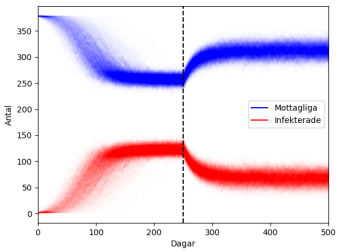
Figur A.3: Graf med 100 simuleringar med dubbel effekt av interventionen handhygien



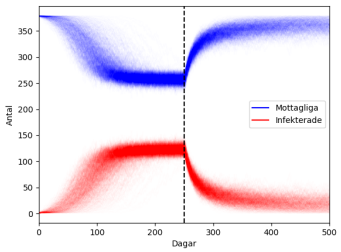
Figur A.4: Graf med 100 simuleringar med halv effekt av interventionen handhygien



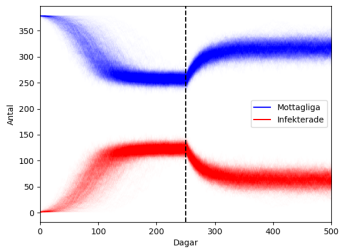
(a) $\overline{I}_f = 122.38$, $\overline{I}_e = 94.74$, $z_{250} = 9197/10000$



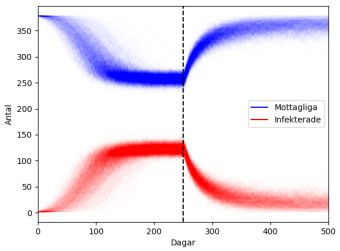
(b) $v_!$: 122.39, $\overline{I}_e = 66.46$, $z_{250} = 9126/10000$



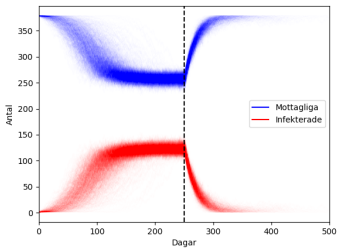
(c) $\overline{I}_f = 122.34$, $\overline{I}_e = 11.88$, $z_{250} = 9155/10000$



(d) $\overline{I}_f = 122.45$, $\overline{I}_e = 61.75$, $z_{250} = 9139/10000$

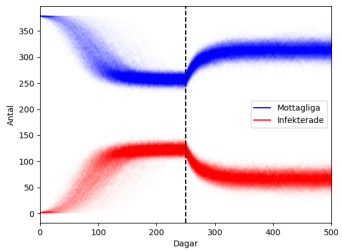


(e) $\overline{I}_f = 122.25$, $\overline{I}_e = 13.17$, $z_{250} = 9140/10000$

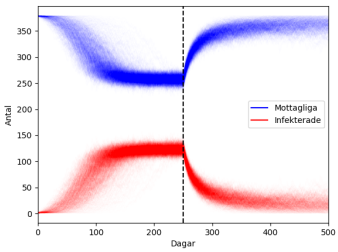


(f) $\overline{I}_f = 122.33$, $\overline{I}_e = 0.05$, $z_{250} = 9160/10000$

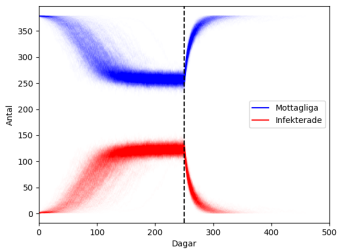
Figur A.5: Simuleringar av screening med slumpmässigt urval i a) - c) på 10, 20 respektive 40 individer och motsvarande för kontaktspårning i d) - f). Tid mellan screening var 2 dagar. $\overline{I}_f$ = Medelvärde av infekterade 50 dagar innan start av intervention, $\overline{I}_e$ = Medelvärde av infekterade sista 50 dagarna, z_{250} = Andel utfall med inga utbrott innan start av intervention.



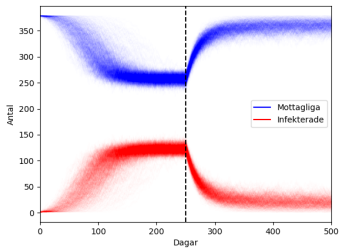
(a) $\overline{I}_f = 122.42$, $\overline{I}_e = 65.26$, $z_{250} = 9134/10000$



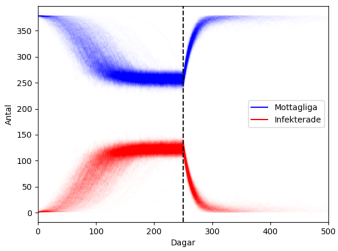
(b) $\overline{I}_f = 122.32$, $\overline{I}_e = 11.08$, $z_{250} = 9160/10000$



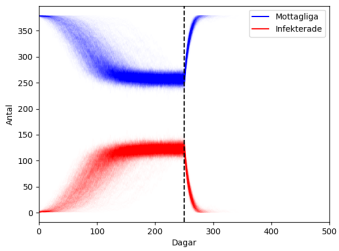
(c) $\overline{I}_f = 122.52$, $\overline{I}_e = 0.00$, $z_{250} = 9178/10000$



(d) $\overline{I}_f = 122.20$, $\overline{I}_e = 15.79$, $z_{250} = 9208/10000$

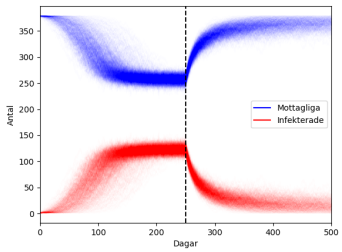


(e) $\overline{I}_f = 122.48$, $\overline{I}_e = 0.18$, $z_{250} = 9161/10000$

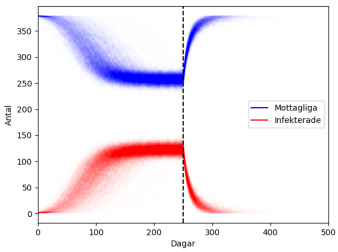


(f) $\overline{I}_f = 122.37$, $\overline{I}_e = 0.00$, $z_{250} = 9166/10000$

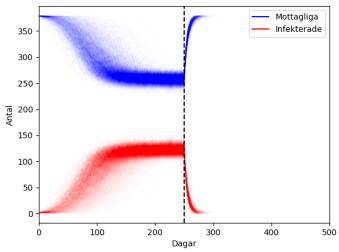
Figur A.6: Simuleringar av screening med slumpmässigt urval i a) - c) på 10, 20 respektive 40 individer och motsvarande för kontaktspårning i d) - f). Tid mellan screening var 1 dag. $\overline{I}_f$ = Medelvärde av infekterade 50 dagar innan start av intervention, $\overline{I}_e$ = Medelvärde av infekterade sista 50 dagarna, z_{250} = Andel utfall med inga utbrott innan start av intervention.



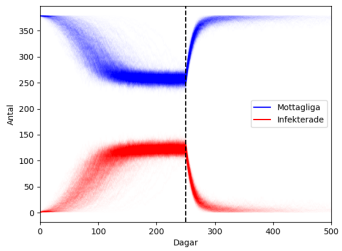
(a) $\overline{I}_f = 122.20$, $\overline{I}_e = 9.32$, $z_{250} = 9173/10000$



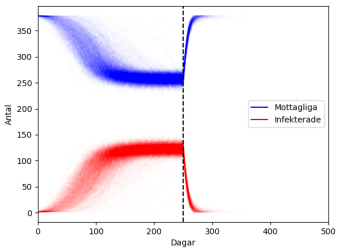
(b) $\overline{I}_f = 122.31$, $\overline{I}_e = 0.00$, $z_{250} = 9229/10000$



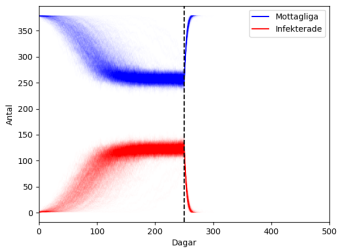
(c) $\overline{I}_f = 122.22$, $\overline{I}_e = 0.00$, $z_{250} = 9134/10000$



(d) $\overline{I}_f = 122.33$, $\overline{I}_e = 0.33$, $z_{250} = 9160/10000$



(e) $\overline{I}_f = 122.01$, $\overline{I}_e = 0.00$, $z_{250} = 9137/10000$



(f) $\overline{I}_f = 122.12$, $\overline{I}_e = 0.00$, $z_{250} = 9163/10000$

Figur A.7: Simuleringar av screening med slumpmässigt urval i a) - c) på 10, 20 respektive 40 individer och motsvarande för kontaktspårning i d) - f). Tid mellan screening var 0.5 dagar. $\overline{I}_f$ = Medelvärde av infekterade 50 dagar innan start av intervention, $\overline{I}_e$ = Medelvärde av infekterade sista 50 dagarna, z_{250} = Andel utfall med inga utbrott innan start av intervention.

B Appendix 2 – källkod

B.1 gillespie.py

```
import numpy as np
import heapq as hq
from enum import Enum
import networkx as nx

# Enumerator for event types
class EventType(Enum):
    TRACING_RANDOM = 1 # Random tests for contact tracing
    TRACING = 2 # Contact tracing
    RANDOM = 3 # Random testing
    FLOW = 4

    def __lt__(self, other):
        return self.value < other.value

# Event for priority queue in Gillespie function
class Event:
    def __init__(self, event_time, event_type, event_data=None):
        self.time = event_time
        self.type = event_type
        self.data = event_data

    def __lt__(self, other):
        return (self.time, self.type) < (other.time, other.type)

def Gillespie(sim_id: int, max_t: float, g_shape: float, g_scale: float, graph:
↪  nx.graph, n_pat: int, screening_start: float, screen_delay: float, n_screen:
↪  int, hh_start: float, hh_tr_mat: np.array, screening_method: EventType) ->
↪  tuple[int, list, list, list, dict]:
    """Stochastic simulation of infectious disease in a hospital.

    Args:
        sim_id (int): Identifier for simulation, used for identification during
        ↪  concurrency.
        max_t (float): Last day of the simulation.
        g_shape (float): Shape of gamma distribution for length of stay.
        g_scale (float): Scale of gamma distribution for length of stay.
        graph (nx.graph): Graph with transmission rates to run the simulation
        ↪  on.
        n_pat (int): Number of patients in the simulation
        screening_start (float): Day when the screening intervation starts.
        screen_delay (float): Duration (in days) between each screening event.
        n_screen (int): The number individuals to test during each screening.
        hh_start (float): Day when the handwashing should start
        hh_tr_mat (np.array): New transmission matrix, to be applied when
        ↪  handhygiene starts.
        screening_method (EventType): Either random or contact tracing screening
        ↪  method.

    Returns:
        tuple[int, list, list, list, dict]: Tuple of simulation id, susceptible
        ↪  data, infected data, time data and positive tests data.
```

```
"""

def matId2graphId(id: int) -> int:
    """Converts the matrix (row) id for a node into its corresponding
    ↪ networkx node id.

    Args:
        id (int): Node matrix id

    Returns:
        int: Networkx id for node.
    """
    if id < n_pat:
        return f"Patient {id}"
    return f"Staff {id-n_pat}"

def graphId2matId(id: int) -> int:
    """Converts the matrix (row) id for a node into its corresponding
    ↪ networkx node id.

    Args:
        id (int): Node networkx id.

    Returns:
        int: Matrix (row) id for node.
    """
    tp, nr = id.split(' ')
    if tp == "Patient":
        return int(nr)
    return n_pat + int(nr)

np.random.seed() # Base simulations on different seeds
n = len(graph.nodes) # Nr in simulation
trans_prob_mat = nx.to_numpy_array(graph, list(sorted(graph.nodes)))
hh_tr_mat = nx.to_numpy_array(hh_tr_mat, list(sorted(hh_tr_mat.nodes)))
I_days = [1]  # Save # of infected each time it changes
S_days = [n-1] # Save # of susceptible each time it changes
T = [0] # Save time that number of infected/susceptible changed
universe = {i for i in range(n)} # All ids in simulation
infec_status = set() # Infected ids
infec_status.add(np.random.randint(0,n_pat)) # Choose one patient at random
↪ as first case
flow_events = [] # List of events of type flow
event_pq = [] # Priority queue for all events
pos_tests=dict() # Nr of positive tests at each screening
t = 0  # Start time

# Add (same) flow event to event_pq and flow_events
for i in range(n_pat):
    # Important that they are pointers, allows finding by both id and event
    ↪ time
    patient_data = Event(np.random.gamma(g_shape, g_scale), EventType.FLOW,
    ↪ i)
    hq.heappush(event_pq, patient_data)
    flow_events.append(patient_data)
```

```
hq.heappush(event_pq, Event(screening_start, screening_method, set())) #
↪  First test event

while t < max_t: # Begin iterative algorithm
    # Switch to hand hygiene transmission matrix
    if t > hh_start:
        trans_prob_mat = hh_tr_mat

    rows = np.fromiter(infec_status, np.intp, len(infec_status))
    cols = np.fromiter(universe.difference(infec_status), np.intp, n -
    ↪  len(infec_status))
    if len(rows) == 0: # Check if no one infected
        break

    # Select only active rows and columns, depends on which nodes are
    ↪  infected
    infec_prob_mat = trans_prob_mat[rows, :][:, cols]
    Lambda = infec_prob_mat.sum() # Total exponential Lambda, as in Gillespie
    ↪  big Lambda

    if Lambda == 0:
        t = event_pq[0].time # Jump to next event if no one infected
    else:
        t += np.random.exponential(1 / Lambda) # Draw time for next infection
        ↪  event

    if event_pq[0].time <= t: # Another event happens before infection event
        event = hq.heappop(event_pq)
        t = event.time # Choose this event time instead, discard
        ↪  exponentially drawn t

        if event.type == EventType.FLOW: # Flow type event
            pat_nr = event.data
            # Updates info for patient so that it becomes a new patient
            event.time = t + np.random.gamma(g_shape, g_scale) # Update LoS
            ↪  for new patient
            hq.heappush(event_pq, event) # Add flow time for new patient
            if pat_nr in infec_status: # Make sure new patient susceptible
                infec_status.remove(pat_nr)
            else:
                continue # If nr of infected didn't change, don't log it
            if len(infec_status) == 0:
                continue # No one infected, quit and don't log

        if event.type == EventType.RANDOM: # No contact tracing
            for i in event.data: # Iterate over positive tests
                if i in infec_status:
                    infec_status.remove(i)  # Isolate individual
                    if i < n_pat: # If patient, also update LoS
                        flow_events[i].time = t + np.random.gamma(g_shape,
                        ↪  g_scale)
                        hq.heapify(event_pq)
            if len(infec_status) == 0:
                continue # No one is infected, quit and don't log
            screen_array = np.random.choice(range(n), n_screen,
            ↪  replace=False) # Select patients to screen
```

```
            positive_tests=set()
            for pat_nr in screen_array: # Collect all positive tests in a
            ↪  set
                pos_tests.setdefault(t, 0)
                if pat_nr in infec_status:
                    pos_tests[t] += 1 # Update nr of positive tests
                    positive_tests.add(pat_nr)
            hq.heappush(event_pq, Event(t+screen_delay, EventType.RANDOM,
            ↪  positive_tests)) # Push new random screening event

        if event.type == EventType.TRACING_RANDOM: # No positive tests last
        ↪  screening or first screening event, part of contact tracing
            screen_array = list(np.random.choice(range(n), n_screen,
            ↪  replace=False)) # Choose randomly to test
            positive_tests=set()
            for i in range(np.size(screen_array)): # Collect all positive
            ↪  tests in a set
                pat_nr = screen_array[i]
                if pat_nr in infec_status:
                    positive_tests.add(pat_nr) # Log tests as positive
            if (len(positive_tests) == 0):
                hq.heappush(event_pq, Event(t+screen_delay,
                ↪  EventType.TRACING_RANDOM, True)) # No one positive =>
                ↪  next screening is random
            else:
                hq.heappush(event_pq, Event(t+screen_delay,
                ↪  EventType.TRACING, positive_tests)) # Positive results =>
                ↪  Next event will be tracing

        if event.type == EventType.TRACING: # Contact tracing
            for i in event.data: # Iterate over positive tests
                if i in infec_status: # Isolate
                    infec_status.remove(i)
                    if i < n_pat: # If patient, also update LoS
                        flow_events[i].time = t + np.random.gamma(g_shape,
                        ↪  g_scale)
                        hq.heapify(event_pq)
            if len(infec_status)==0:
                continue # No one infected, break and don't log
            positive_tests=set()
            bfs_levels = dict(enumerate(nx.bfs_layers(graph,
            ↪  list(map(matId2graphId, event.data))))) # Neighbours of each
            ↪  order
            bfs_levels[list(bfs_levels.keys())[-1]+1] = bfs_levels.pop(0) #
            ↪  Put starting nodes last
            selected_set = set() # To be tested
            for level in bfs_levels.values():
                lvl = list(map(graphId2matId, level))
                if len(selected_set) + len(lvl) >= n_screen: # Too many nodes
                ↪  in levels
                    selected_set.update(np.random.choice(list(lvl),
                    ↪  n_screen-len(selected_set), replace=False)) # Select
                    ↪  random subset of correct size
                    break
                else:
```

```
                        selected_set.update(lvl) # Add all neighbours at this
                        ↪  level, continue adding more
                for i in selected_set: # Log positive tests
                    pos_tests.setdefault(t, 0)
                    if i in infec_status:
                        pos_tests[t] += 1
                        positive_tests.add(i)
                if len(positive_tests) == 0:
                    hq.heappush(event_pq, Event(t+screen_delay,
                    ↪  EventType.TRACING_RANDOM, True)) # Next screening is
                    ↪  random
                    # 0 positive tests already logged as default above
                else:
                    hq.heappush(event_pq, Event(t+screen_delay,
                    ↪  EventType.TRACING, positive_tests)) # Next screening is
                    ↪  tracing
            # Log nr of infected, susciptible and current time
            I_days.append(len(infec_status))
            S_days.append(n - I_days[-1])
            T.append(t)
            continue

        rand_event = np.random.uniform(0,Lambda) # Draw on which edge the
        ↪  infection was transmitted
        # Find first entry with at least the value rand_event (where transmission
        ↪  happened)
        event_pos = np.divmod(np.searchsorted(infec_prob_mat.cumsum(),
        ↪  rand_event), cols.shape)
        cols.sort()
        infected_idx = cols[event_pos[1]][0]
        if not infected_idx in infec_status:
            infec_status.add(infected_idx) # Set individual to infected
            # Log nr of infected, susciptible and current time
            I_days.append(I_days[-1] + 1)
            S_days.append(n - I_days[-1])
            T.append(t)
    return sim_id, S_days, I_days, T, pos_tests
```

B.2 pool.py

```
import multiprocessing as mp
from gillespie import Gillespie
from graph import *
from handhygiene import int_rates
import numpy as np
import pandas as pd
import time
from matplotlib import pyplot as plt
import matplotlib.lines as mlines
import bisect
from alive_progress import alive_bar
import networkx as nx
from gillespie import EventType

rates = {
    ('patient', 'patient'): 6*3e-5,
```

```
    ('staff', 'staff'): 6*2.51e-4,
    ('patient', 'staff'): 9*1.19e-4,
    ('staff', 'patient'): 6*7.22e-4,
}
graph, n_pat = realistic_graph(rates)
# graph, n_pat = example_graph(rates)
# graph, n_pat = ring_graph(rates, 50)
with_handhygiene = True
if with_handhygiene:
    graph, n_pat = realistic_graph(int_rates)
else:
    graph, n_pat = realistic_graph(rates)
tr_mat = nx.to_numpy_array(graph, list(sorted(graph.nodes)))
n = len(graph.nodes)

# Simulation parameters
n_tries = 100 # Simulation iterations
n_days_average = 50 # Number of days to take the average of the infected
days_before_int = 250 # Days before intervention starts
sim_days = 500 # Total days to simulate
screening_freq = 1 # Days between each screening
n_screen = range(n_pat+1) # Nr of patients to screen each time
hh_start = 800  # When to start hand hygiene
hh_tr_mat, _ = realistic_graph(int_rates)  # transmission matrix for hand
↪  hygiene
mod_n = n # For performing simulations using different x values, eg screening
↪  evaluation
test_type = EventType.RANDOM
screening_delay = 0.5  # Days until result from screening is received
n_to_test = 40

start = time.time() # Run time measurements

# Make sure there is enough time for intervention and final averages
assert days_before_int-n_days_average > 0 and
↪  sim_days-days_before_int-n_days_average > 0

def gel_wrapper(args):
    return Gillespie(*args)

def pool_handler ():
    last_avg = []
    before_int_avg = []
    pos_tests_tot = {}
    infec_tot = {}
    time_tot = {}
    n_no_epidemic = 0 # Number of results with quick death of process, ignored
    pool = mp.Pool(mp.cpu_count())
    fig, ax1 = plt.subplots()

    # Perform concurrent simulations
    # res = pool.imap_unordered(gel_wrapper, [(i%mod_n+1, sim_days, 3.5, 2,
    ↪  graph, n_pat, days_before_int, screening_delay, i%mod_n+1, hh_start,
    ↪  hh_tr_mat, test_type) for i in range(n_tries)])
```

```
res = pool.imap_unordered(gel_wrapper, [(n_to_test, sim_days, 3.5, 2, graph,
↪  n_pat, days_before_int, 2, n_to_test, hh_start, hh_tr_mat, test_type) for
↪  i in range(n_tries)])
with alive_bar(n_tries) as bar:
    bar.text("Performing simulations")
    for res_part in res:
        bar()
        sid, S, I, T, pos_tests = res_part # Get results from Gillespie
        ↪  algorithm
        df = pd.DataFrame({"T": T, "S": S, "I": I}).set_index("T")
        if T[-1] <= days_before_int or len(pos_tests) == 0: # Ingore
        ↪  simulations which didn't last long enough, or for screening
        ↪  simulations which didn't result in any positive tests
            n_no_epidemic += 1
        else:
            # Find data before intervention started and the last selected
            ↪  days (after intervention)
            n_before_intervention = df.loc[days_before_int - n_days_average:
            ↪  days_before_int]
            n_after_intervention = df.loc[df.index[-1] - n_days_average:
            ↪  df.index[-1]]

            # Calculate average values, taking into account the uneven
            ↪  spacing between data points by weighing data points with the
            ↪  spacing between them
            i_before = np.array(n_before_intervention['I'])
            t_before = np.array(n_before_intervention.index)-(days_before_in
            ↪  t-n_days_average)
            dt_before = np.diff(t_before,prepend=0) # Intervals, including
            ↪  first value/interval
            mean_before = np.sum(i_before*dt_before)/n_days_average #
            ↪  Average, weighed for total amount of days (includes 0s for
            ↪  simulations which ended early)
            before_int_avg.append(mean_before)

            i_after = np.array(n_after_intervention["I"])
            t_after = np.array(n_after_intervention.index)-(df.index[-1]-n_d
            ↪  ays_average)
            dt_after = np.diff(t_after, prepend=0)
            mean_after = np.sum(i_after * dt_after)/n_days_average
            last_avg.append(mean_after)

            # Add data to dictionaries
            pos_tests_tot.setdefault(sid, [])
            pos_tests_tot[sid].append(pos_tests) # Add simulation data to
            ↪  corresponding nr of tests
            infec_tot.setdefault(sid, [])
            infec_tot[sid].append(I)
            time_tot.setdefault(sid, [])
            time_tot[sid].append(T)
        ax1.plot(T, I, c="r", alpha=1, lw=0.25, label="I")
        ax1.plot(T, S, c="b", alpha=1, lw=0.25, label="S")
print(f"Average value for {n_days_average} days before intervention:",
↪  np.average(before_int_avg))
print(f"Average value for last {n_days_average} days:", np.average(last_avg))
```

```
print(f"No outbreaks before {days_before_int} days/total:
↪  {n_no_epidemic}/{n_tries}={n_no_epidemic/n_tries}")

plt.xlabel("Dagar")
plt.ylabel("Antal")
blue_line = mlines.Line2D([], [], color='blue', label='Mottagliga')
red_line = mlines.Line2D([], [], color="red", label="Infekterade")
plt.legend(handles=[blue_line, red_line])

plt.axvline(days_before_int, c="black", linestyle="dashed")
plt.savefig("pool.png", dpi=300)
plt.clf()
test_infecs = {}
mn = {}

# Calculate values for evaluation graph
with alive_bar(len(time_tot.keys())) as bar:
    bar.text("Processing data")
    for j1 in time_tot.keys():
        bar()
        j1_tots = []
        for j2 in range(len(time_tot[j1])):
            test_event_pos = {}
            for x in pos_tests_tot[j1][j2].keys():
                test_event_pos[x] = bisect.bisect_left(time_tot[j1][j2], x) #
                ↪  Find event corresponding to each time for a positive
                ↪  test
            test_infecs.setdefault(j1, {})
            test_infecs[j1].setdefault(j2, {})
            for k in pos_tests_tot[j1][j2].keys():
                test_infecs[j1][j2][k] = infec_tot[j1][j2][test_event_pos[k]]

            # Invalid point detection, for debugging
            """invalid_points = [(k, pos_tests_tot[j1][j2][k],
            ↪  test_infecs[j1][j2][k],
            ↪  pos_tests_tot[j1][j2][k]/test_infecs[j1][j2][k]) for k in
            ↪  list(pos_tests_tot[j1][j2].keys()) if
            ↪  pos_tests_tot[j1][j2][k]/test_infecs[j1][j2][k] > 1]
            if len(invalid_points) != 0:
                print("INVALID POINT DETECTED")
                print(invalid_points)"""

            plt.xlim(0, 1)
            # plt.ylim(0, 1)
            try:
                j1_tots.extend(list(np.array(list(pos_tests_tot[j1][j2].valu⌋
                ↪  es()))/np.array(list(test_infecs[j1][j2].values()))))
            except RuntimeWarning:
                print("failed to calculate mean")
                print("Pos tests:", list(pos_tests_tot[j1][j2].values()))
                print("Test infecs", test_infecs[j1][j2].values())
        mn[j1] = np.mean(j1_tots)

# Plot evaluation graph
time_tot = dict(sorted(time_tot.items()))
pos_tests_tot = dict(sorted(pos_tests_tot.items()))
```

```
    test_infecs = dict(sorted(test_infecs.items()))
    mn = dict(sorted(mn.items()))
    ks = [k for k in test_infecs.keys()]
    xs = [j/n for j in time_tot.keys()]
    ys = mn.values()
    print("xs len", len(xs))
    # plt.plot(xs, ys, c="w", linewidth=5)
    plt.plot(xs, ys, linewidth=3, label="Simulering")
    plt.plot((0,1),(0,1), label="Teoretisk slump")
    plt.xlabel("Andel testade")
    plt.ylabel("Antal positiva test/antal infekterade")
    plt.legend()
    plt.savefig("screening_eval.png", dpi=500)

if __name__ == "__main__":
    pool_handler()

# Print total time
print("Total time:", time.time() - start)
print("Time/n:", (time.time() - start)/n_tries)
```

B.3 handhygiene.py

```
# Normal infection rates
rates = {
    ('patient', 'patient'): 6 * 3e-5,
    ('staff', 'staff'): 6 * 2.51e-4,
    ('patient', 'staff'): 9 * 1.19e-4,
    ('staff', 'patient'): 6 * 7.22e-4,
}

# Percentual improvement of rubbing alcohol
dif=(0.83-0.58)/0.83

# Double the effect of the intervention
with_double_effect=False
if with_double_effect:
    dif=dif*2
else:
    dif=dif

# Half the effect of the intervention
with_half_effect=False
if with_half_effect:
    dif=dif/2
else:
    dif=dif

# Hand hygiene infection rates. Staff -> Patient and Staff -> Staff are
↪  effected.
int_rates = {
    ('patient', 'patient'): 6 * 3e-5,
    ('staff', 'staff'): 6 * 2.51-4*(1-dif),
    ('patient', 'staff'): 9 * 1.19e-4,
    ('staff', 'patient'): 6 * 7.22e-4*(1-dif),
}
```

B.4 r0.py

```
import numpy as np
import networkx as nx
import multiprocessing as mp
from alive_progress import alive_bar
import heapq as hq
from gillespie import Event, EventType


MAX_CPUS = 12  # Parameter for parallelisation


def run(node, sims_per_node, n, max_t, g_shape, g_scale, graph, n_pat,
↪  screen_delay, screening_start, screening_method, n_screen, hh_start,
↪  hh_tr_mat):
    current_r0 = 0
    for _ in range(sims_per_node):
        current_r0 += approx_r0_for_node(node, n=n, max_t=max_t, g_shape=g_shape,
        ↪  g_scale=g_scale, graph=graph, n_pat=n_pat, screen_delay=screen_delay,
        ↪  screening_start=screening_start, screening_method=screening_method,
        ↪  n_screen=n_screen, hh_start=hh_start, hh_tr_mat=hh_tr_mat)
    return current_r0 / sims_per_node


def run_wrap(args):
    return run(*args)

# def approx_r0(sims_per_node, n, max_t, g_shape, g_scale, graph, n_pat,
↪  screen_delay, screening_start, screening_method, n_screen, hh_start,
↪  hh_tr_mat):
def approx_r0(sims_per_node: int, n: int, max_t: float, g_shape: float, g_scale:
↪  float, graph: nx.graph, n_pat: int, screening_start: float, screen_delay:
↪  float, n_screen: int, hh_start: float, hh_tr_mat: np.array, screening_method:
↪  EventType):
    """ Approximates R_0 for a given graph and parameters in parallel

    Args:
        sims_per_node (int): Number of simulations per node
        n (int): Size of network
        max_t (float): Last day of the simulation
        g_shape (float): Shape of gamma distribution for length of stay.
        g_scale (float): Scale of gamma distribution for length of stay.
        graph (nx.graph): Graph with transmission rates to run the simulation
        ↪  on.
        n_pat (int): Number of patients in the simulation
        screening_start (float): Day when the screening intervation starts.
        screen_delay (float): Duration (in days) between each screening event.
        n_screen (int): The number individuals to test during each screening.
        hh_start (float): Day when the handwashing should start
        hh_tr_mat (np.array): New transmission matrix, to be applied when
        ↪  handhygiene starts.
        screening_method (EventType): Either random or contact tracing screening
        ↪  method.

    Returns:
```

```
        float: approximated R_0

    """
    r0 = 0

    pool = mp.Pool(max(mp.cpu_count(), MAX_CPUS))

    # For each node, call run with arguments
    res = pool.imap_unordered(run_wrap, [(i, sims_per_node, n, max_t, g_shape,
    ↪  g_scale, graph, n_pat, screen_delay, screening_start, screening_method,
    ↪  n_screen, hh_start, hh_tr_mat) for i in range(n)])
    with alive_bar(n) as bar:
        for res_part in res:
            bar()
            r0 += res_part

    return r0 / n  # Average over all nodes


def approx_r0_for_node(node, n, max_t, g_shape, g_scale, graph, n_pat,
↪  screen_delay, screening_start, screening_method, n_screen, hh_start,
↪  hh_tr_mat):
    """ Approximate R_0 for a given node once.

    Note:
        Very similar to `Gillespie` in `gillespie.py`. Differences are
        ↪  commented.

    """


    def mat2graph(id):
        if id < n_pat:
            return f"Patient {id}"
        return f"Staff {id-n_pat}"

    def graph2mat(id):
        tp, nr = id.split(' ')
        if tp == "Patient":
            return int(nr)
        return n_pat + int(nr)

    new_infections = 0

    np.random.seed()
    n = len(graph.nodes)
    trans_prob_mat = nx.to_numpy_array(graph, list(sorted(graph.nodes)))
    hh_tr_mat = nx.to_numpy_array(hh_tr_mat, list(sorted(graph.nodes)))

    universe = {i for i in range(n)}

    infec_status = set()
    infec_status.add(node) # Start with `node` infected

    patient_flow_objects = []
    event_pq = []
```

```
for i in range(n_pat):
    patient_data = Event(np.random.gamma(g_shape, g_scale), EventType.FLOW,
    ↪  i)
    hq.heappush(event_pq, patient_data)
    patient_flow_objects.append(patient_data)

hq.heappush(event_pq, Event(screening_start, screening_method, set())) #
↪  First test
t = 0
while t < max_t:
    if not node in infec_status:
        break

    # Switch to hand hygiene
    if t > hh_start:
        trans_prob_mat = hh_tr_mat

    # Select only `node` row and active columns, depends on which nodes are
    ↪  infected
    trans_prob_mat_specific = trans_prob_mat[np.array([node]), :]
    rows = np.array([node])  # only `node` can infect
    cols = np.fromiter(universe.difference(infec_status), np.intp, n -
    ↪  len(infec_status))

    if len(rows) == 0:
        break # `node` is not infected

    infec_prob_mat = trans_prob_mat_specific[:, cols]

    Lambda = infec_prob_mat.sum() # Total exponential Lambda, as in Gillespie
    ↪  big Lambda
    if Lambda == 0:
        t = event_pq[0].time # Jump to next event if no one infected
    else:
        t += np.random.exponential(1 / Lambda) # Take a time step

    if event_pq[0].time <= t:
        event = hq.heappop(event_pq)
        t = event.time

        if event.type == EventType.FLOW: #If event is flow
            pat_nr = event.data
            event.time = t + np.random.gamma(g_shape, g_scale)
            hq.heappush(event_pq, event) #add next flow time
            if pat_nr in infec_status: #New patient
                infec_status.remove(pat_nr)
            else:
                continue # Don't log if susceptible flowed
            if len(infec_status)==0:
                continue # No one infected, quit


        if event.type == EventType.TRACING_RANDOM: #First order screening
            screen_array = list(np.random.choice(range(n), n_screen,
            ↪  replace=False))
```

```
            positive_tests=set()
            for i in range(np.size(screen_array)): #Collect all positive
            ↪  tests in a set
                pat_nr = screen_array[i]
                if pat_nr in infec_status:
                    positive_tests.add(pat_nr)
            if (len(positive_tests) == 0):
                hq.heappush(event_pq, Event(t+screen_delay,
                ↪  EventType.TRACING_RANDOM, True)) #Next screening time
            else:
                hq.heappush(event_pq, Event(t+screen_delay,
                ↪  EventType.TRACING, positive_tests)) #Push data and
                ↪  results to heap

        if event.type == EventType.RANDOM: #No contact tracing
            for i in event.data: #iterate over positive tests
                if i in infec_status: #Isolate
                    infec_status.remove(i)
                    if i < n_pat: #Flowtime, if applicable
                        patient_flow_objects[i].time = t +
                        ↪  np.random.gamma(g_shape, g_scale)
                        hq.heapify(event_pq)

            if len(infec_status) == 0:
                continue # No one is infected

            screen_array = np.random.choice(range(n), n_screen,
            ↪  replace=False) #Select patients to screen
            positive_tests=set()
            for pat_nr in screen_array: #Collect all positive tests in a set
                if pat_nr in infec_status:
                    positive_tests.add(pat_nr)
            hq.heappush(event_pq, Event(t+screen_delay, EventType.RANDOM,
            ↪  positive_tests)) #Next screening time

        if event.type == EventType.TRACING: #Any result event
            for i in event.data: #iterate over positive tests
                if i in infec_status: #Isolate
                    infec_status.remove(i)
                    if i < n_pat: #Flowtime, if applicabloe
                        patient_flow_objects[i].time = t +
                        ↪  np.random.gamma(g_shape, g_scale)
                        hq.heapify(event_pq)

            if len(infec_status)==0:
                continue

            positive_tests=set()
            bfs_levels = dict(enumerate(nx.bfs_layers(graph,
            ↪  list(map(mat2graph, event.data)))))
            bfs_levels[list(bfs_levels.keys())[-1]+1] = bfs_levels.pop(0)
            selected_set = set()
            for level in bfs_levels.values():
                lvl = list(map(graph2mat, level))
                if len(selected_set) + len(lvl) >= n_screen: # Too many nodes
                ↪  in levels
```

```
                        selected_set.update(np.random.choice(list(lvl),
                        ↪  n_screen-len(selected_set), replace=False))
                        break
                    else:
                        selected_set.update(lvl)

                for i in selected_set:
                    if i in infec_status:
                        positive_tests.add(i)


                if len(positive_tests) == 0:
                    hq.heappush(event_pq, Event(t+screen_delay,
                    ↪  EventType.TRACING_RANDOM, True)) #Next screening time
                    # 0 already logged as default above
                else:
                    hq.heappush(event_pq, Event(t+screen_delay,
                    ↪  EventType.TRACING, positive_tests)) #Push neighbour test
                    ↪  results

            continue

        rand_event = np.random.uniform(0,Lambda) # Randomize event
        # Find first entry with at least the value rand_event

        event_pos = np.divmod(np.searchsorted(infec_prob_mat.cumsum(),
        ↪  rand_event), cols.shape)
        cols.sort()
        infected_idx = cols[event_pos[1]][0]
        if not infected_idx in infec_status:
            infec_status.add(infected_idx) # Set individual to infected
            new_infections += 1 # Total count increases instead of logging
            ↪  current counts for time `t`

    return new_infections

if __name__ == '__main__':
    from graph import realistic_graph
    from handhygiene import rates, int_rates

    graph, n_pat = realistic_graph(rates)
    hh_graph, _ = realistic_graph(int_rates)

    n_calcs = 100
    sn = 0
    sr = 0
    st = 0
    n_to_test = 40
    when_to_test = 2
    hh_start = 200 # Value abouve `max_t` means it is inactive
    for i in range(n_calcs):
        print("Iteration", i + 1, "/", n_calcs)
        # Random screening
```

```
        sr += approx_r0(10, n=len(graph.nodes), max_t=100, g_shape=3.5,
        ↪  g_scale=2, graph=graph, n_pat=n_pat, screen_delay=when_to_test,
        ↪  screening_start=0, screening_method=EventType.RANDOM,
        ↪  n_screen=n_to_test, hh_start=hh_start, hh_tr_mat=hh_graph)
        # Screening with tracing
        st += approx_r0(10, n=len(graph.nodes), max_t=100, g_shape=3.5,
        ↪  g_scale=2, graph=graph, n_pat=n_pat, screen_delay=when_to_test,
        ↪  screening_start=0, screening_method=EventType.TRACING_RANDOM,
        ↪  n_screen=n_to_test, hh_start=hh_start, hh_tr_mat=hh_graph)
    print("sn", sn/n_calcs)
    print("sr", sr/n_calcs)
    print("st", st/n_calcs)
```

B.5 graph.py

```
import numpy as np
import networkx as nx


def fully_connect(g: nx.DiGraph, nodes: list, rates: dict[(str, str), float]):
    """
    For a group of nodes `nodes` in the network `g`, connect each node to
    every other node.
    """

    data = g.nodes.data()
    for na in nodes:
        role_a = data[na]['role']
        for nb in nodes:
            if na == nb:
                continue
            role_b = data[nb]['role']
            g.add_edge(na, nb, weight=rates[(role_a, role_b)])


def bi_connect(g: nx.DiGraph, nodes1: list, nodes2: list, rates: dict[(str, str),
↪   float]):
    """
    For two groups of nodes `nodes1` and `nodes2` in the network `g`, connect
    each node in `nodes1` to every node in `nodes2`, and vice versa.
    """

    data = g.nodes.data()
    for na in nodes1:
        role_a = data[na]['role']
        for nb in nodes2:
            role_b = data[nb]['role']
            g.add_edge(na, nb, weight=rates[(role_a, role_b)])
            g.add_edge(nb, na, weight=rates[(role_b, role_a)])


num_created = {role : 0 for role in ['patient', 'staff']}

def patient_room(g, num_patients, rates):
    """
    Creates a room with `num_patients` patients in the graph `g`, connecting
```

```
    them with rates attained from `rates`
    """

    global num_created
    patientnr = [f"Patient {num_created['patient'] + i}" for i in
    ↪  range(num_patients)]
    num_created['patient'] += num_patients

    for node in patientnr:
        g.add_node(node, role='patient')
    fully_connect(g, patientnr, rates)
    return patientnr


def staff_room(g, num_staff, rates):
    """
    Creates a room with `num_staff` staff in the graph `g`, connecting
    them with rates attained from `rates`
    """
    global num_created
    nodes = [f"Staff {num_created['staff'] + i}" for i in range(num_staff)]
    num_created['staff'] += num_staff

    for node in nodes:
        g.add_node(node, role='staff')
    fully_connect(g, nodes, rates)
    return nodes


def connect_rooms_to_room(g, rooms: list[list], room: list, rates) -> nx.DiGraph:
    """
    Connect, to `room`, every room in `rooms`

    rooms: list[list[node label]]
    room: list[node label]
    """
    for r in rooms:
        bi_connect(g, r, room, rates)


def example_graph(rates) -> tuple[np.ndarray, int]:
    """
    Returns an example graph and the amount of patients. Patients are the first
    rows and columns.
    """
    g = nx.DiGraph()

    room1 = patient_room(g, 5, rates)
    room2 = patient_room(g, 5, rates)
    room3 = patient_room(g, 5, rates)

    nurses = staff_room(g, 3, rates)
    doctors = staff_room(g, 2, rates)

    connect_rooms_to_room(g, [room1, room2, room3], nurses, rates)
    connect_rooms_to_room(g, [room1, room2, room3], doctors, rates)
```

```
    # !!!  OBS  !!!
    # resetta `num_created` till 0, så att den inte stör när vi skapar flera
    num_patients = num_created['patient']
    for key in num_created.keys():
        num_created[key] = 0
    return g, num_patients

def giant_graph(rates):
    g = nx.DiGraph()

    n_groups = 8
    n_patient_rooms = 6
    n_patients_per_room = 5
    n_nurses_per_room = 6
    n_doctors = 40

    patient_groups = []
    for _ in range(n_groups):
        group = []
        for _ in range(n_patient_rooms):
            group.append(patient_room(g, n_patients_per_room, rates))
        patient_groups.append(group)

    nurse_groups = []
    for i in range(n_groups):
        nurses = staff_room(g, n_nurses_per_room, rates)
        connect_rooms_to_room(g, patient_groups[i], nurses, rates)
        nurse_groups.append(nurses)

    doctors = staff_room(g, n_doctors, rates)
    for i in range(n_groups):
        connect_rooms_to_room(g, patient_groups[i], doctors, rates)
    connect_rooms_to_room(g, nurse_groups, doctors, rates)

    num_patients = num_created['patient']
    for key in num_created.keys():
        num_created[key] = 0
    return g, num_patients

def realistic_graph(rates):
    g = nx.DiGraph()

    n_wards = 10
    n_rooms_per_ward = 8
    n_beds_per_room = 4
    n_staff_per_ward = 6

    wards = []
    for _ in range(n_wards):
        ward = []
        for _ in range(n_rooms_per_ward):
            room = patient_room(g, n_beds_per_room, rates)
            ward.append(room)
        wards.append(ward)
```

```
    staff_rooms = [[], []]
    for i in range(n_wards // 2):
        group1 = staff_room(g, n_staff_per_ward, rates)
        group2 = staff_room(g, n_staff_per_ward, rates)
        connect_rooms_to_room(g, wards[i], group1, rates)
        connect_rooms_to_room(g, wards[i], group2, rates)
        connect_rooms_to_room(g, wards[n_wards - i - 1], group1, rates)
        connect_rooms_to_room(g, wards[n_wards - i - 1], group2, rates)
        for s in group1:
            staff_rooms[0].append(s)
        for s in group2:
            staff_rooms[1].append(s)

    for sr in staff_rooms:
        fully_connect(g, sr, rates)

    num_patients = num_created['patient']
    for key in num_created.keys():
        num_created[key] = 0
    return g, num_patients

def ring_graph(rates, n):
    node_names = [f"Patient {i}" for i in range(n)]
    g=nx.cycle_graph(node_names)
    return g, n


def split_realistic_graph(rates):
    g = nx.DiGraph()

    n_wards = 10
    n_rooms_per_ward = 8
    n_beds_per_room = 4
    n_staff_per_ward = 6

    wards = []
    for _ in range(n_wards):
        ward = []
        for _ in range(n_rooms_per_ward):
            room = patient_room(g, n_beds_per_room, rates)
            ward.append(room)
        wards.append(ward)

    # Five staff rooms instead of two
    staff_rooms = [[], [], [], [], []]
    for i in range(n_wards // 2):
        group1 = staff_room(g, n_staff_per_ward, rates)
        group2 = staff_room(g, n_staff_per_ward, rates)
        connect_rooms_to_room(g, wards[i], group1, rates)
        connect_rooms_to_room(g, wards[i], group2, rates)
        connect_rooms_to_room(g, wards[n_wards - i - 1], group1, rates)
        connect_rooms_to_room(g, wards[n_wards - i - 1], group2, rates)

        for s in group1:
            staff_rooms[i % 5].append(s)
        for s in group2:
```

```
            staff_rooms[(i + 1) % 5].append(s)

    for sr in staff_rooms:
        fully_connect(g, sr, rates)

    num_patients = num_created['patient']
    for key in num_created.keys():
        num_created[key] = 0
    return (nx.to_numpy_array(g, list(sorted(g.nodes)))), num_patients


if __name__ == '__main__':
    rates = {
        ('patient', 'patient'): 12 * 3e-5,
        ('staff', 'staff'): 9 * 1.19e-4,
        ('patient', 'staff'): 15 * 7.22e-4,
        ('staff', 'patient'): 6 * 2.51e-4,
    }
    g, n_patients = split_realistic_graph(rates)
    print("Nodes:", g.shape[0])
    print("Patients:", n_patients)
```