AIMS Bioengineering, 9(3): 1–8.
DOI: 10.3934/bioeng.2022016
Received: 05 May 2022
Revised: 08 July 2022
Accepted: 25 July 2022
Published: 29 Aug 2022

http://www.aimspress.com/journal/Bioengineering

Research article

Computational simulations of the effects of social distancing interventions on the COVID-19 pandemic

Sarbaz H. A. Khoshnaw* and Azhi Sabir Mohammed

Department of Mathematics, University of Raparin, Ranya, Kurdistan Region of Iraq

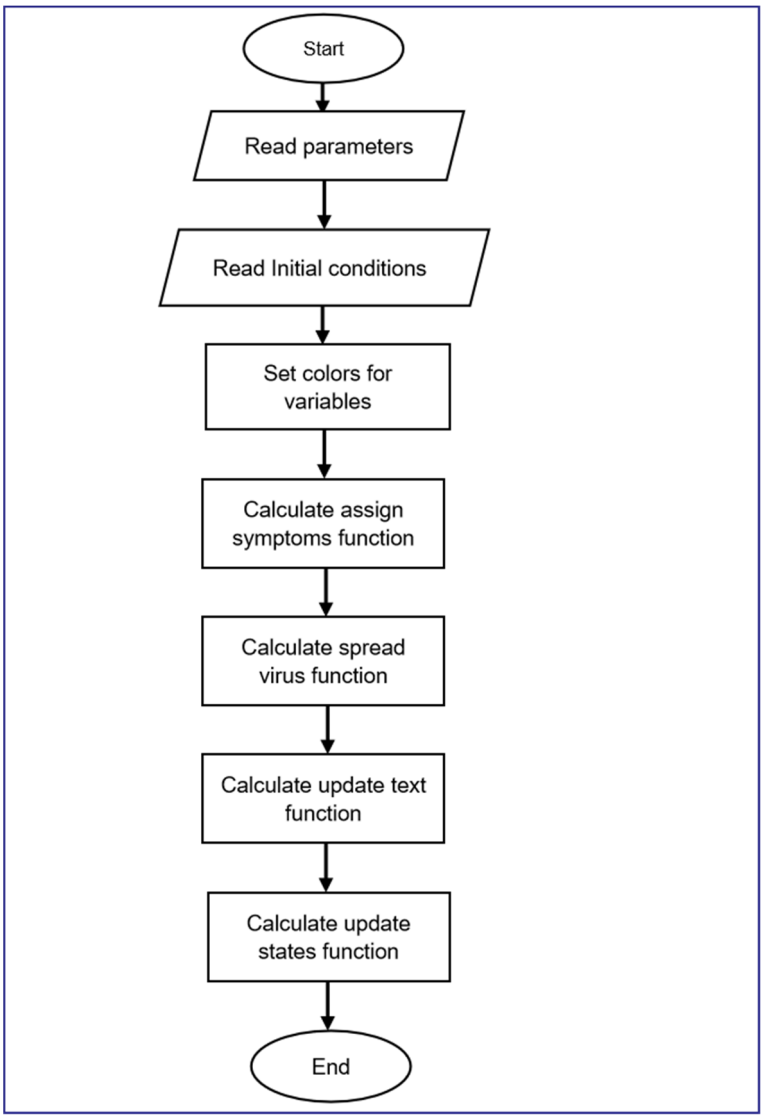
* **Correspondence:** Email: sarbaz.hamza@uor.edu.krd; Tel: +9647503161780.

Abstract: The spread of the COVID-19 pandemic has been considered as a global issue. Based on the reported cases and clinical data, there are still required international efforts and more preventative measures to control the pandemic more effectively. Physical contact between individuals plays an essential role in spreading the coronavirus more widely. Mathematical models with computational simulations are effective tools to study and discuss this virus and minimize its impact on society. These tools help to determine more relevant factors that influence the spread of the virus. In this work, we developed two computational tools by using the R package and Python to simulate the COVID-19 transmissions. Additionally, some computational simulations were investigated that provide critical questions about global control strategies and further interventions. Accordingly, there are some computational model results and control strategies. First, we identify the model critical factors that helps us to understand the key transmission elements. Model transmissions can significantly be changed for primary tracing with delay to isolation. Second, some types of interventions, including case isolation, no intervention, quarantine contacts and quarantine contacts together with contacts of contacts are analyzed and discussed. The results show that quarantining contacts is the best way of intervening to minimize the spread of the virus. Finally, the basic reproduction number R_0 is another important factor which provides a great role in understanding the transmission of the pandemic. Interestingly, the current computational simulations help us to pay more attention to critical model transmissions and minimize their impact on spreading this disease. They also help for further interventions and control strategies.

Keywords: infectious disease; COVID-19 pandemic; basic reproduction number R_0; computational simulations; networks

Appendix

Appendix A: This flowchart presents the main steps of the simulation of the COVID-19 spread using the Python program.



Flowchart 1. Main steps of COVID-19 pandemic simulation using Python.

The main steps in the flowchart can be explained and given in more detail. The first step is reading the model parameters that contain the initial model parameter values for COVID-19, such as R_0 = 2.8, incubation = 6, percent mild = 0.81, mild recovery = (8,12), percent severe = 0.14, severe recovery = (21,42), severe death = (14,56), fatality rate = 0.034 and serial interval = 5. The second step is reading the initial model conditions that include some initial model values such as day = 0, total number infected = 0, number of currently infected = 1, number of recoveries = 0, number of deaths = 0, exposed before =0, exposed after = 1 and total number in population = 4500. Then, we define the colors of compartments in polar coordinates, for example, uninfected is grey, infected is red, recovery is green and dead is black. After that, we use the following formulas to calculate some model quantities (signs and symptoms.):

Mild fast = incubation + first value of mild recovery
Mild slow = incubation + second value of mild recovery
Sever fast = incubation + first value of severe recovery
Severe slow = incubation + second value of severe recovery
Death fast = incubation + first value of severe death (2)
Death slow = incubation + second value of severe death
Number mild =round(percent mild × number new infected)
Number severe=round(percent severe × number new infected)
Percent severe recovery =1 - (fatality rate/percent severe)
Number severe recovery=round(percent severe recovery × number severe)

The next step is about calculating the spread of the virus function. In this step, the virus can be spread through the population using the given conditions.

exposed before = exposed after
if rem(day , serial interval) ==0 and exposed after < 4500 then
*number new infected = round(R_0 * total number infected)*
*exposed after = exposed after = round(number new infected * 1.1)*
if exposed after > 4500 then (3)
*number new infected = round((4500 - exposed after)*0.9)*
exposed after = 4500
number currently infected = number currently infected + number new infected
total number infected = total number infect + number new infected
day = day + 1

From the following step, we update the compartments using the update text functions given below:

indices = arrange(0 , population) + 0.5
*thetas = pi * (1 + sqrt(5)) * indices*
rs = sqrt (indices / population)
set day text = (" Day " , day) (4)
set infected text = (" Infected : " , number currently infected)
set deaths text = (" Deaths : " , number deaths)
set recovered text = (" Recovered : " , number recovered)

The compartment colors should be updated when we have red for infected individuals, green for recovered individuals and black for dead individuals. This is done via the update states function given below:

axes.scatter (thetas[0] , rs[0] , color = RED)
if day> = mild fast then
mild thetas = mild[day]["thetas"]
mild rs = mild[day]["rs"]
axes.scatter (mild thetas, mild rs , color = Green)
number recovered = number recovered +length(mils thetas)
number currently infected = number currently infected -length(mils thetas)
if day >= severe fast then
rec thetas = severe["recovery"] day["thetas"] (5)
rec rs = severe["recovery"] [day]["rs"]
axes.scatter(rec thetas, rec rs, color = GREEN)
number recovered = number recovered + length (rec thetas)
number currently infected = number currently infected - length(rec thetas)
if day >= death fast then

```
            death thetas = severe["death"]  [day]["thetas"]
            death rs = severe["death"]  [day]["rs"]
            axes.scatter(death thetas, death rs,color = BLACK)
    number deaths = number deaths + len(death thetas)
    number currently infected = number currently infected- len(death thetas)
```

Appendix B: Computational simulation using R for testing capacity. Testing capacity for COVID-19 is another important intervention for reducing the transmission of COVID-19. Different testing capacities are shown in Figure 5. It can be seen that the high capacity of testing plays a great role in reducing the infected people; see Figure 5b,c. Other methods such as optimal control can be used to maximize the number of people returning to normal life and minimize the number of active infected individuals [30].

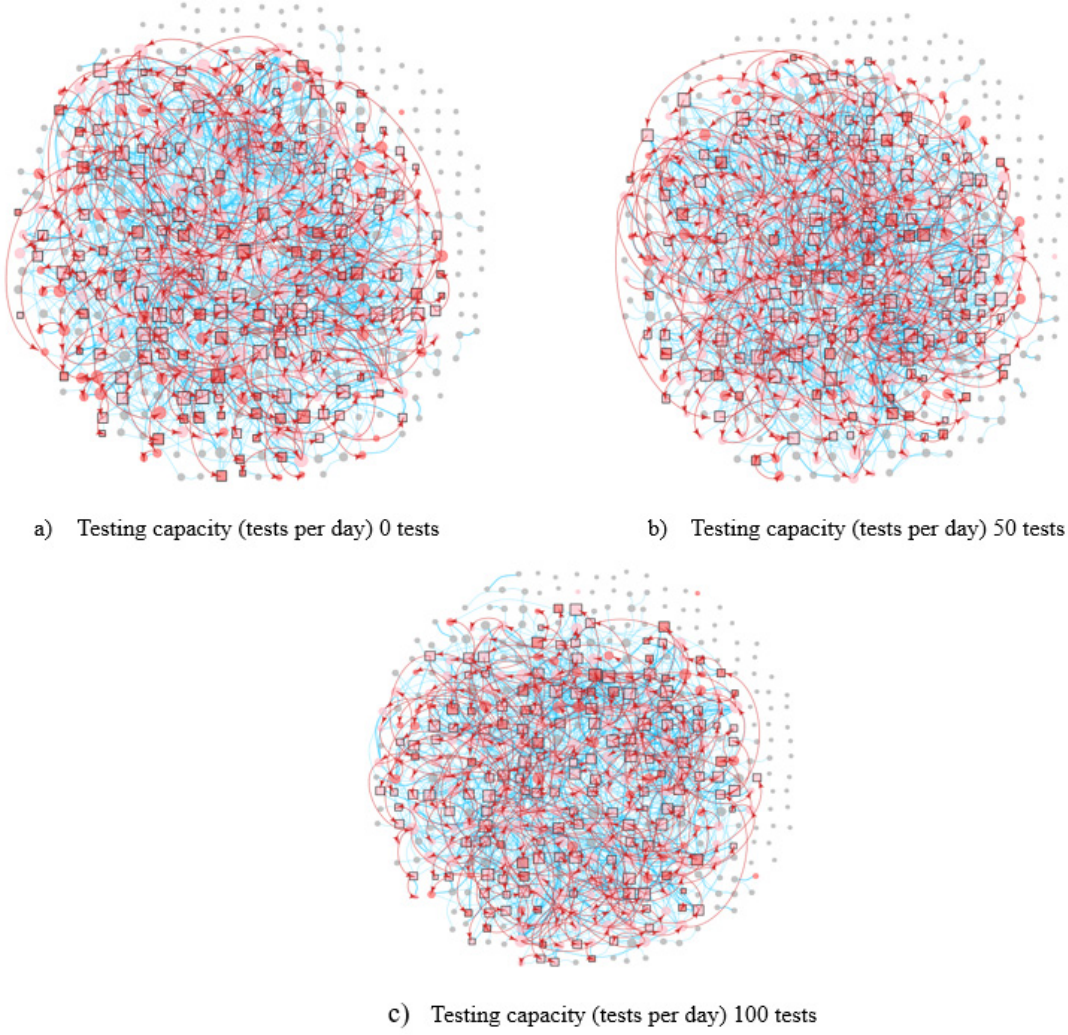


Figure 5. There are some testing capacities (tests per day): a) 0 tests, b) 50 tests and c) 100 tests. The gray circles, dark red circles, and light red circle show uninfected individuals, currently infected individuals and previously infected individuals, respectively. The blue lines show contacts between individuals. The red lines show infections between individuals. The squares represent currently isolated/quarantined individuals.

Appendix C: Computational simulation using R for case isolation with periods of delay to isolation. Isolation of the infected individuals can be used as a control strategy; see Figure 6. However, Figures 6a–c show that this technique may not decrease the virus spread significantly.

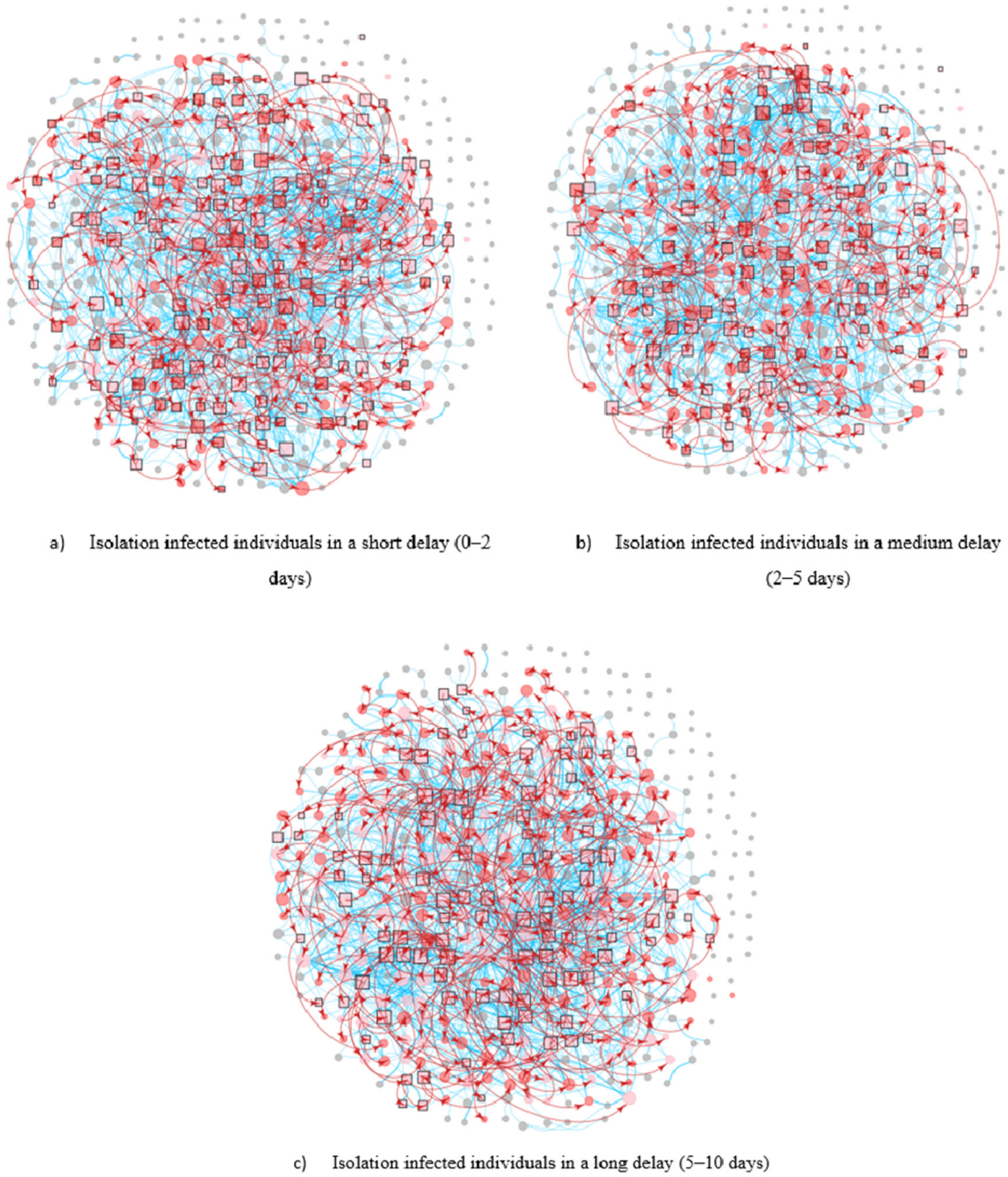


Figure 6. The interventions scenario is case isolation. The periods of delay to isolation are a) short (0–2 days), b) medium (2–5 days) and c) long (5–10 days). The gray circles, dark red circles and light red circles stand for uninfected individuals, currently infected individuals and previously infected individuals, respectively. The blue lines show contacts between individuals. The red lines show infections between individuals. The squares represent currently isolated/quarantined individuals.

Appendix D: Computational simulation using R. Types of intervention to control the spread of the virus presented in Figure 7.

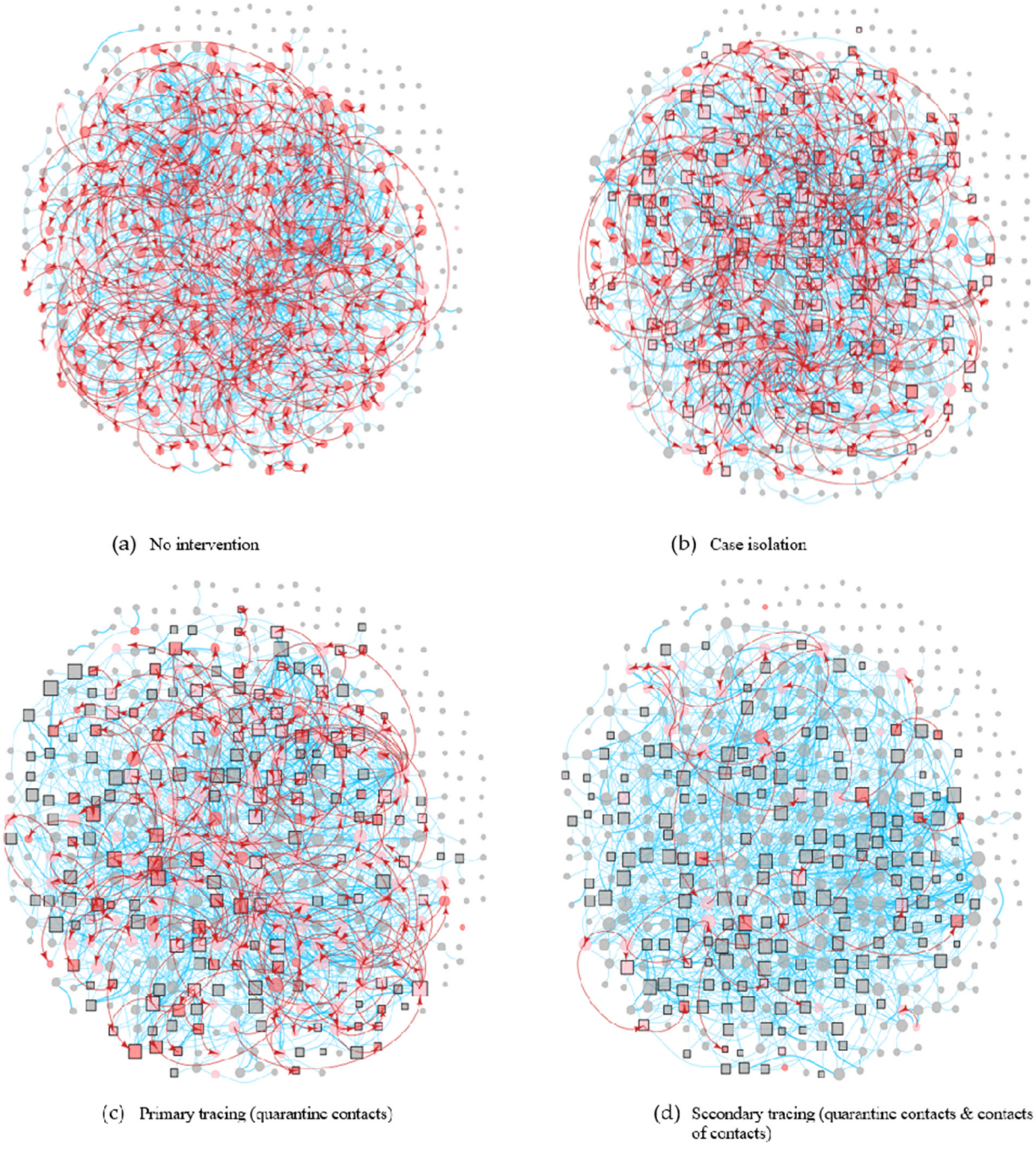
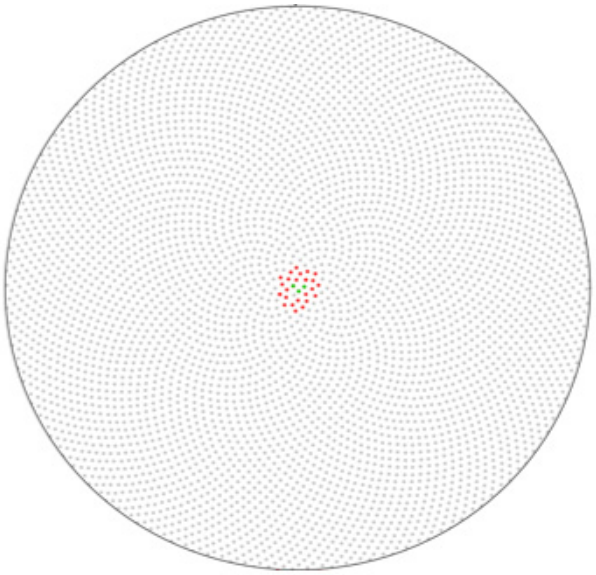
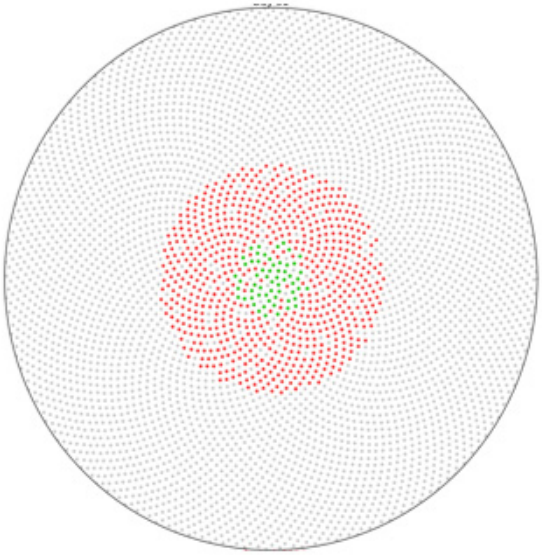


Figure 7. Types of interventions to control the spread of the virus: a) no intervention, b) case isolation, c) primary tracing (quarantine contacts) and d) secondary tracing (quarantine contacts and contacts of contacts). The gray circles, dark red circles and light red circles stand for uninfected individuals, currently infected individuals and previously infected individuals, respectively. The blue lines show contacts between individuals. The red lines show infections between individuals. The squares represent currently isolated/quarantined individuals.

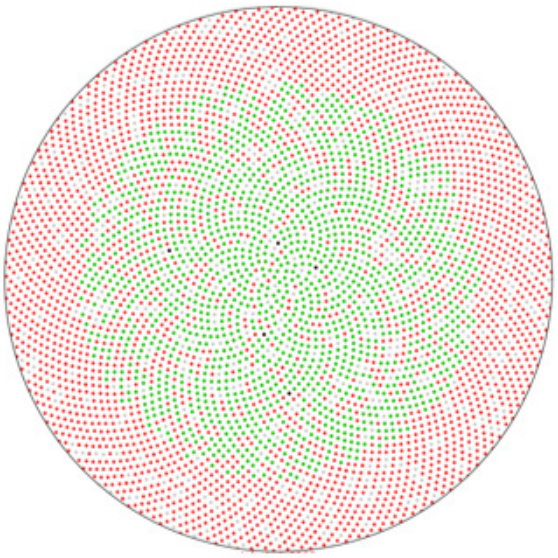
Appendix E: Computational simulation using Python. Figure 8 shows the topological changes of the sub-population when the basic reproduction number $R_0 = 2$.



(a) Computational simulation at day 15 when Infected = 78, Recovered = 3, Deaths = 0.



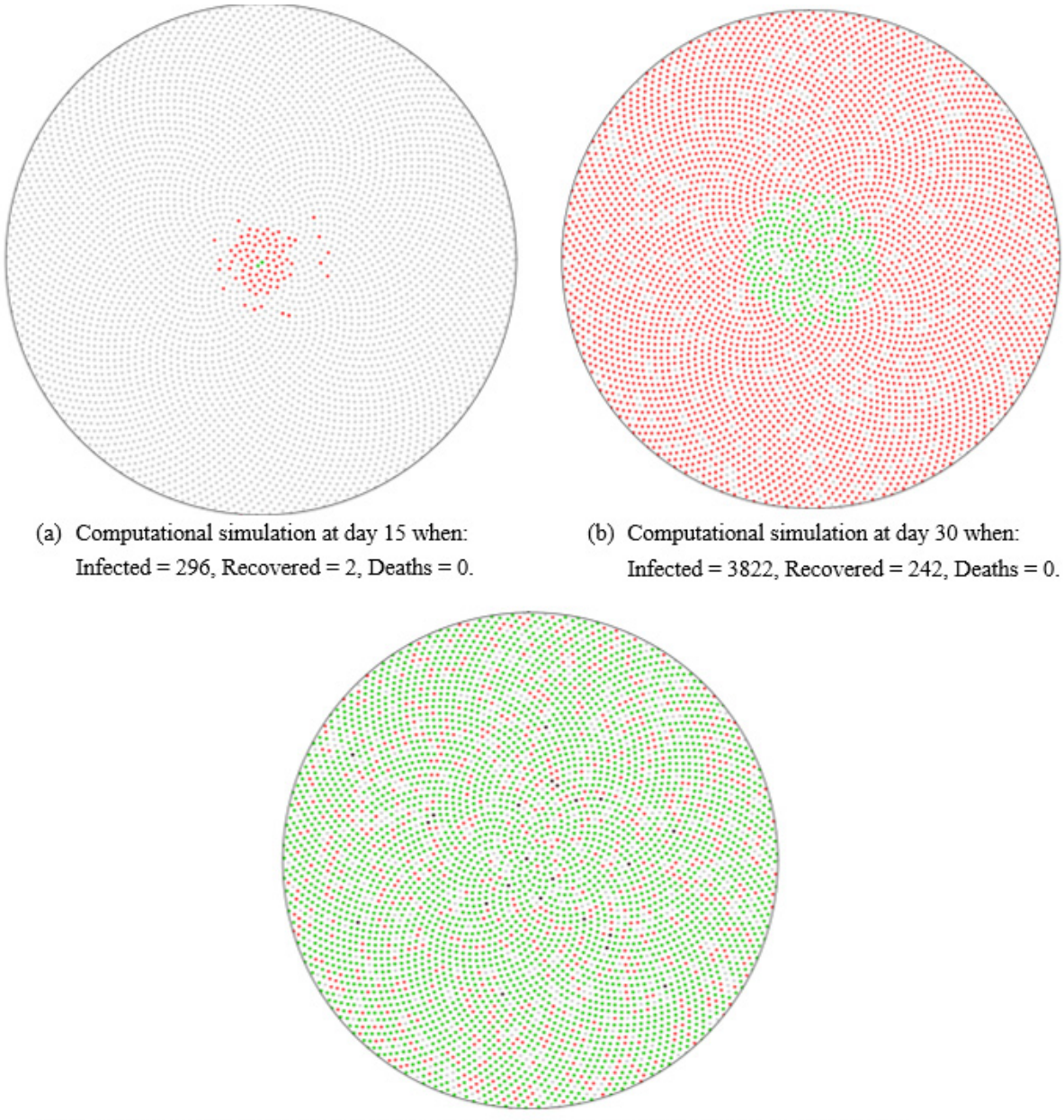
(b) Computational simulation at day 30 when: Infected = 2128, Recovered = 59, Deaths = 0.



(c) Computational simulation at day 45 when: Infected = 2588, Recovered = 1479, Deaths = 5.

Figure 8. Computational simulations calculated when the basic reproduction number (R_0) is 2. The gray circles, red circles, green circles and black circles stand for uninfected, currently infected, previously infected and dead individuals, respectively.

Appendix F: Computational simulation using Python. Figure 9 shows the topological changes of the sub-population when the basic reproduction number $R_0 = 3.2$.



(a) Computational simulation at day 15 when: Infected = 296, Recovered = 2, Deaths = 0.

(b) Computational simulation at day 30 when: Infected = 3822, Recovered = 242, Deaths = 0.

(c) Computational simulation at day 45 when: Infected = 741, Recovered = 3301, Deaths = 22.

Figure 9. Computational simulations calculated when the basic reproduction number R_0 is 3.2. The gray circles, red circles, green circles and black circles stand for uninfected, currently infected, previously infected and dead individuals, respectively.