Reporte 6:
Sistema Multiagente

Jorge Torres

20 de marzo de 2022

1. Objetivo

En esta práctica se implementa un sistema multiagente aplicado en epidemiología a través del modelo SIR, que consiste en definir tres tipos de agentes: susceptibles, infectados y recuperados. El objetivo es el de analizar el efecto que tiene el contener o reducir la velocidad de movimiento de los agentes infectados, tanto en la magnitud de la epidemia (la altura del pico en la curva del porcentaje de infectados por iteración), como en su velocidad (la iteración en la cual se llega por la primera vez al valor pico).

2. Desarrollo

Basando el desarrollo de la práctica en el código desarrollado por E. Schaeffer [1], en primer lugar, se definen los parámetros iniciales con los cuales actúa el sistema, como se observa en el código 1. Estos se conforman por 1) el largo de cada lado del sistema, `l`, 2) la cantidad de agentes que se crean, `n`, 3) la probabilidad inicial de infectados, `pi`, 4) la probabilidad de que cada agente infectado se recupere, `pr`, 5) la velocidad inicial de los agentes, `v`, 6) el umbral de cercanía para considerar una infección, `r`, 7) la cantidad de pasos que hacen los agentes, `tmax`, y 8) la cantidad de veces que se repite el experimento, `runs`. El desarrollo completo se puede observar en el repositorio en GitHub de J. Torres [2]

Código 1: Parámetros Iniciales

```
l = 1.5
n = 50
pi = 0.05
pr = 0.02
v = l / 30
r = 0.1
tmax = 100
runs = 100
```

Con la función `contagiados()` del código 2 se decide qué agentes son los próximos en infectarse de acuerdo a la probabilidad, p_c, descrita en la ecuación 1,

$$p_c = \begin{cases} 0, & \text{si } d(i,j) \geq r, \\ \frac{r-d}{r}, & \text{para cualquier otro caso,} \end{cases} \tag{1}$$

donde d es la distancia euclideana entre un par, (i, j), de agentes y r es el umbral de cercanía.

Código 2: Nuevos Agentes Infectados

```
def contagiados():
    for i in range(n):
        a1 = agentes.iloc[i]
        if a1.estado == 'I':
            for j in range(n):
                a2 = agentes.iloc[j]
                if a2.estado == 'S':
                    d = sqrt((a1.x - a2.x)**2 + (a1.y - a2.y)**2)
                    if d < r:
                        if random() < (r - d) / r:
                            contagios[j] = True
    return contagios
```

Utilizando el código 3, se crean los agentes, se distribuyen uniformemente a través del espacio y se decide su estado inicial de infección utilizando la probabilidad inicial, `pi`. Además, para cada agente creado, se decide de manera aleatoria un diferencial de movimiento que representa su velocidad, dado por los términos `dx` y `dy`.

Código 3: Creación de Agentes

```
agentes =  pd.DataFrame()
agentes['x'] = [uniform(0, l) for i in range(n)]
agentes['y'] = [uniform(0, l) for i in range(n)]
agentes['estado'] = ['S' if random() > pi else 'I' for i in range(n)]
agentes['dx'] = [uniform(-v, v) for i in range(n)]
agentes['dy'] = [uniform(-v, v) for i in range(n)]
```

Con las líneas del código 4 se lleva un conteo de la cantidad de infectados para cada paso iterado, y se manda a llamar a la función `contagiados()` del código 2 para decidir los nuevos agentes infectados. Si la cantidad de infectados llega a cero, el programa se detiene.

Código 4: Conteo de Infectados

```
epidemia = []
    for tiempo in range(tmax):
        conteos = agentes.estado.value_counts()
        infectados = conteos.get('I', 0)
        epidemia.append(infectados)
        contagios = [False for i in range(n)]
        if infectados == 0:
            break
        contagios = contagiados()
```

En seguida, se decide si un agente cambia de estado a infectado (por medio de la función `contagiados()`, descrita previamente), o a recuperado por medio de la probabilidad de recuperación, `pr`. En el código 5 se observan éstas funciones.

Código 5: Decisión de Agentes Infectados y Recuperados

```
        for i in range(n):
            a = agentes.iloc[i]
            if contagios[i]:
                agentes.at[i, 'estado'] = 'I'
            elif a.estado == 'I':
                if random() < pr:
                    agentes.at[i, 'estado'] = 'R'
```

Por último, se trasladan todos los agentes a su nueva posición dada por los diferenciales `dx` y `dy`, establecidos en el código 3. Las líneas `3` a `6` simplemente establecen que los agentes que lleguen a un límite del plano reaparecen en el límite opuesto, con la misma velocidad y dirección. Con las líneas `9` y `10` se crea una lista de la cantidad máxima de infectados y la iteración en la que se llegó por primera vez a esa cantidad, para todas las repeticiones del experimento. Esto se observa en el código 6.

Código 6: Movimiento de los Agentes

```
        x = a.x + a.dx
        y = a.y + a.dy
        x = x if x < l else x - l
        y = y if y < l else y - l
        x = x if x > 0 else x + l
        y = y if y > 0 else y + l
        agentes.at[i, 'x'] = x
        agentes.at[i, 'y'] = y
    maximos['Maximos Infectados'].append(max(epidemia))
    maximos['Posicion'].append(epidemia.index(max(epidemia)) + 1)
```

2.1. Reducción de Velocidad de Agentes Infectados

Para hacer una comparación entre el modelo epidemiológico con agentes infectados a velocidad normal y uno en donde éstos reducen su velocidad a la mitad, se tienen que realizar algunas modificaciones. Las líneas `5` y `6` del código 3 se modifican para reducir la velocidad de los agentes inicialmente infectados a la mitad, como se observa en el código 7.

Código 7: Modificación del Código 3

```
agentes['dx'] = [uniform(-v/2, v/2) if  agentes.at[i, 'estado'] == 'I'\
                 else uniform(-v, v) for i in range(n)]
agentes['dy'] = [uniform(-v/2, v/2) if  agentes.at[i, 'estado'] == 'I'\
                 else uniform(-v, v) for i in range(n)]
```

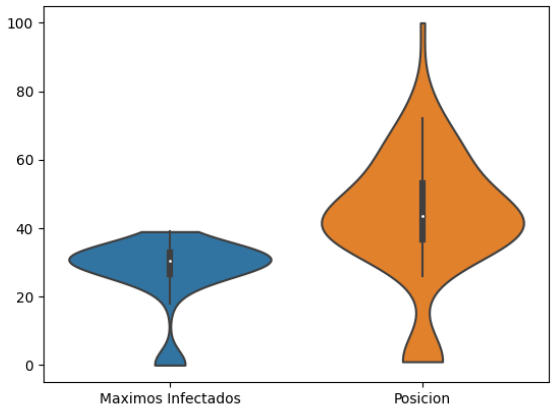
Por otro lado, al código 5 se agregan dos líneas para reducir la velocidad de los nuevos agentes infectados, de tal manera que resulta en el código 8.

Código 8: Modificación del Código 5

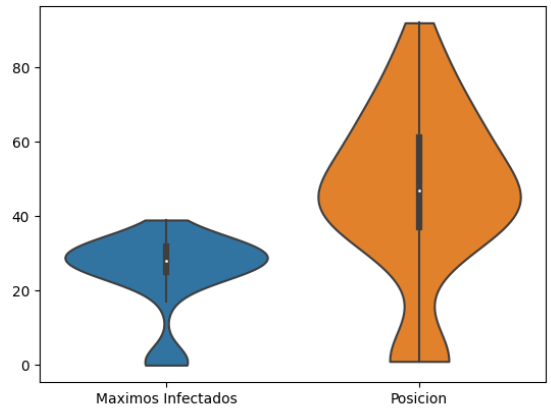
```
        for i in range(n):
            a = agentes.iloc[i]
            if contagios[i]:
                agentes.at[i, 'dx'] /= 2
                agentes.at[1, 'dy'] /= 2
                agentes.at[i, 'estado'] = 'I'
            elif a.estado == 'I':
                if random() < pr:
                    agentes.at[i, 'estado'] = 'R'
```

3. Resultados

En los diagramas de la figura 1 se puede observar una comparación entre las distribuciones de los picos de infectados y las iteraciónes en que se llega por primera vez a esos picos, para una epidemia con velocidad normal de agentes (figura 1a) y una en donde los agentes infectados reducen su velocidad a la mitad (figura 1b).



(a) Distribución de epidemia a velocidad normal.



(b) Distribución de epidemia con velocidad reducida.

Figura 1: Distribuciónes de máxima cantidad de infectados y posición del pico para cada tipo de epidemia.

4. Conclusiones

Al analizar la comparación de la figura 1, se puede observar que, al reducir la velocidad de agentes infectados, no hay una diferencia relevante en cuanto a la máxima cantidad de infectados, ambas epidemias llegando a picos de aproximadamente 30. Esto puede deberse a la cantidad limitada de agentes totales y el hecho de que los agentes recuperados ya no pueden infectarse ni propagar la infección.

Por otro lado, sí se observa una diferencia significativa en cuanto al comportamiento en el tiempo de la epidemia, observando que la velocidad de infección se reduce al hacerlo también la velocidad de los agentes infectados, propagándose más lentamente.

Referencias

[1] E. Schaeffer. Multiagent, 2019. URL https://github.com/satuelisa/Simulation/tree/master/MultiAgent.

[2] J. Torres. P6, 2022. URL https://github.com/FeroxDeitas/Simulacion-Nano/tree/main/Tareas/P6.