Simulación de Sistemas
Práctica 6

Sistema multiagente
1455175: Ángel Moreno

18 de septiembre de 2018

Resumen

En esta práctica se simula un sistema multiagente secuencial y luego compararlo con una simulación en paralelo, se observar y prueba que no cambia la distribución de la simulación en el efecto de un variable respuesta.

1. Introducción

En un sistema multiagente se tiene un conjunto de entidades con estados internos que reaccionan entre ellos mismos cambiando sus estados. Se maneja tres estados internos, conocido como el modelos **SIR**: **s**usceptibles, **i**nfectados y **r**ecuperados.

Para los n agentes se tiene una probabilidad de inicio de contagio p_i y una probabilidad de recuperación p_r. Supongo que, por simplicidad, la infección produce inmunidad en los recuperado, solamente los susceptibles serán infectados. La probabilidad de contagio p_c es proporcional a la distancia euclidiana entre dos agentes $d(i, j)$ como sigue:

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

donde r es un umbral.

Los agentes se encuentran en el subespacio $l \times l \subset \mathbb{R}^2$ con velocidad y dirección. Se posicionan de forma uniforme al azar en el subespacio mencionado considerándolo como un torus.

2. Tarea

Identifica partes de los cálculos que se pueden implementar de formas más eficientes y en particular paraleliza todo lo que conviene con cualquiera de los dos paquetes que hemos visto. Asegura con pruebas estadísticas adecuadas que no se haya modificado el comportamiento del modelo matemático de la simulación. Estudia además el efecto de la probabilidad p_i (de 0.1 a 0.9 en pasos de 0.1) en el porcentaje máximo de infectados durante la simulación.

3. Simulacion

Los parámetros que se utilizan son: el tamaño de la cuadricula de $l = 1{,}5$, un total de $n = 50$ agentes, para cada agente una velocidad entre $(-l/30, l/30)$ para cada componente (x, y), probabilidades para inicio de infección del agente de $0{,}1, 0{,}2, ..., 0{,}9$, el radio del umbral de $r = 0{,}1$, la probabilidad de recuperación de $p_r = 0{,}02$ y un tiempo de 100 iteraciones.

Primero muestro el pseudocodigo de la simalucion:

Pseudocodigo Sistema Multiagente

Entrada: Conjunto de datos de n agentes con sus componentes (x, y), velocidad y estado.
Salida: Porcentajes de infeccion en 100 tiempos.

1. **for** $i \leftarrow 1$ to $tmax = 100$ **do**
2. % epidemia $\leftarrow$ agregamos porcentaje de infectados.
3. Exploramos quienes se van a contagiar.
4. Actualizamos.

La simulación secuencial y la simulación trabajando en paralelo se ejecutaron con códigos donde la mayor parte son similares excepto la parte de exploración de los contagios. El siguiente código es como se ejecuto en la simulación secuencial.

```
contagios <- rep(FALSE, n)
   for (i in 1:n) { # posibles contagios
       a1 <- agentes[i, ]
       if (a1$estado == "I") { # desde los infectados
           for (j in 1:n) {
               if (!contagios[j]) { # aun sin contagio
                   a2 <- agentes[j, ]
                   if (a2$estado == "S") { # hacia los susceptibles
                       dx <- a1$x - a2$x
                       dy <- a1$y - a2$y
                       d <- sqrt(dx^2 + dy^2)
                       if (d < r) { # umbral
                           p <- (r - d) / r
                           if (runif(1) < p) {
                               contagios[j] <- TRUE
                           }
                       }
                   }
               }
           }
       }
   }
```

Esta parte del código se observa que tiene complejidad $\mathcal{O}(n^2)$. El siguiente código se ejecuto en la simulación en paralelo.

```
contagios <- rep(FALSE, n)
    suceptibles <- dim(agentes[agentes$estado == "S",])[1]
    if (suceptibles != 0) {
      posibles <- as.numeric(row.names(agentes[which(agentes$estado == "I"),])) #solo los infectados
      for (i in posibles) { # propagar sobre los infectados
        a1 <- agentes[i, ]
        candidatos <- agentes[which(contagios %in% F),] # solo propagar con los FALSE
        elegidos <- as.numeric(row.names(candidatos[which(candidatos$estado == "S"),])) #solo los suceptibles
        for (j in elegidos) {
          a2 <- agentes[j, ]
          dx <- a1$x - a2$x
          dy <- a1$y - a2$y
          d <- sqrt(dx^2 + dy^2)
          if (d < r) { # umbral
            p <- (r - d) / r
            if (runif(1) < p) {
              contagios[j] <- TRUE
            }
          }
        }
      }
    }
```

Esta modificación del código empieza preguntado si existen susceptibles a cuales infectar es eficaz para probabilidades de inicio de infección grandes y haciedo exploración solo desde los agentes infectados hacia los susceptibles sin hacer la exploración sobre todos los agentes.

Se ejecuto 40 replicas de cada probabilidad en la simulación secuencial y en paralelo, se calculo una media para cada tiempo sobre el porcentage de infeccion de las 40 replicas obteniendo la figura 1

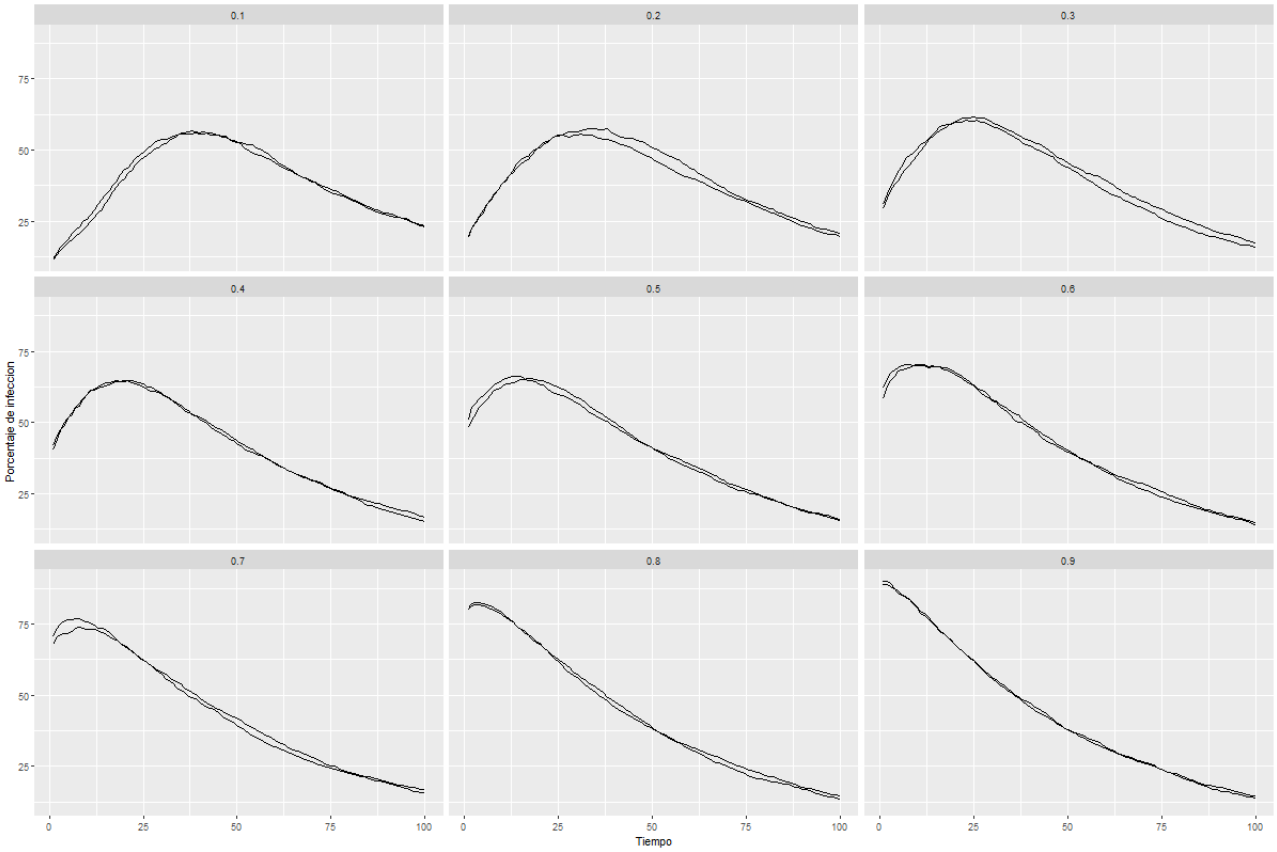

Figura 1: Gráfica de lineas de la simulación secuencia junto a el paralelo para cada probabilidad .

Se concluye gráficamente de la figura 1 que ni la simulación secuencial y la del paralelo no afectan la simulación, ademas se observan porcentajes máximos de infección alcanzado en tiempos de entre (30, 50) en probabilidades de 0.1, 0.2 y 0.3, para las demás probabilidades los porcentajes máximos se alcanzan en tiempos menores a 30.

Se verifica bajo la prueba estadística no parametrica de Kruskal-Wallis con H_0: los datos vienen de la misma distribución, es decir, la ejecución secuencial y en paralelo no afectan la simulación.

```
> kruskal.test(PorInfectados ~ Categorias, data = Resultados)

    Kruskal-Wallis rank sum test

data:  x by Group.2
Kruskal-Wallis chi-squared = 0.032494, df = 1, p-value = 0.8569
```

Para un nivel de significación $\alpha = 0{,}05$ la hipótesis nula se acepta, es decir la simulación no se ve afectada por lo secuencial y lo paralelo.

Referencias

[1] Schaeffer E. *R paralelo: simulación and análisis de datos, 2018.* https://elisa.dyndns-web.com/teaching/comp/par/

[2] Kurt Will. *6 Neat Tricks with Monte Carlo Simulations — Count Bayesie; Probably a probability blog, Marzo 24, 2015.* https://www.countbayesie.com/blog/2015/3/3/6-amazing-trick-with-monte-carlo-simulations

[3] Valdes Eduardo. *Repository of Github, 2017.* https://github.com/eduardovaldesga/SimulacionSistemas

[4] Garcia Beatriz. *Repository of Github, 2017.* https://github.com/BeatrizGarciaR/SimulacionSistemas

[5] Saus Liliana. *Repository of Github, 2018.* https://github.com/pejli/simulacion