Práctica No 6: Sistemas Multi-agente

Héctor Hernán Montes García

19 de septiembre de 2017

Resumen

En la presente práctica estudiamos una aplicación de sistemas multi-agente a un problema ya clásico en epidemiología: el modelo SIR (Susceptibles-Infecciosos-Removidos). En este modelo se contamina una población inicial con cierta cantidad de infecciosos, y se estudia la evolución del contagio a través del tiempo. El primer propósito de la práctica es evaluar la conveniencia de una implementación paralelizada del algoritmo vs una secuencial en lo que respecta a la reducción de tiempos computacionales. El segundo propósito es aprovechar la paralelización para evaluar el impacto de una estrategia de vacunación inicial de los individuos en el comportamiento de la epidemia. El tercer propósito es estudiar la forma en que las probabilidades iniciales de infección afectan la evolución de la epidemia. Los resultados respaldan la conveniencia de la paralelización del código. Luego, con esta implementación la experimentación evidencia la existencia de un umbral de infección por encima del cual la epidemia resulta inminente si una estrategia de vacunación no se ha aplicado, y en sentido inverso, se ratifica la importancia de cumplir con un umbral crítico de vacunación para prevenir la aparición de epidemia.

1. Planteamiento del problema

En esta práctica estudiaremos tres problemas independientes, a continuación una descripción de cada uno.

1. Investigar oportunidades de mejora en el código SIR, e implementar paralelización allí donde se justifique su conveniencia. Estudiar las mejoras en tiempos computacionales para una implementación paralelizada versus una secuencial. Para ello correremos varias réplicas del mismo modelo básico de experimento dado en la clase, con idénticos parámetros por defecto y naturalmente que sin incluir la estrategia de vacunación. Esto con el fin de probar la mejora en la versión más sencilla del algoritmo. De acuerdo con los comportamientos visualizados en un boxplot comparativo de tiempos computacionales, se decidirá si con una pequeña cantidad de réplicas es suficiente para concluir favorablemente por alguna estrategia o si se requiere aumentar la cantidad de réplicas. Finalmente se correrá un modelo un poco más complejo que incluya estrategia de vacunación con el fin de ratificar los resultados obtenidos en la primera fase.

2. Investigar el efecto de una estrategia de vacunación sobre el comportamiento de la epidemia. Para ello se procederá a usar la versión paralelizada del algoritmo para una búsqueda intensiva de los porcentajes máximos

de infectados que se logran en un mismo horizonte de tiempo límite (100 iteraciones) cuando se aplican diferentes probabilidades de vacunación iniciales (desde 0,00 a 0,40). Para facilitar la comparación se mantendrá fija la cantidad de agentes (50), y la probabilidad de infección se hará variar en el rango de 0.00 a 0.10 en pasos de 0.01. Se tiene presente que un experimento más completo podría desempeñarse para identificar los efectos conjuntos del tiempo límite de experimentación, de la probabilidad de recuperación, de la cantidad de agentes, o de otros valores de probabilidades de infección, pero un diseño experimental de dicha naturaleza sería oneroso computacionalmente, y por diversas limitaciones técnicas del equipo usado tan sólo se considerarán los escenarios antes planteados. A investigadores interesados en profundizar, se les recomienda el uso de una estrategia taguchi o de tamizado para recorrer diferentes especificaciones paramétricas, tener una idea preliminar del impacto de los parámetros y sus interacciones en la dinámica del fenómeno, y luego proceder con un diseño experimental focalizado en la investigación cuidadosa de los hallazgos preliminares.

3. Investigar el efecto de las probabilidades de infección en el porcentaje máximo de infectados durante la simulación. Este objetivo se ataca indirectamente con contribuciones del punto anterior, pero acá se amplía la investigación sobre el efecto del porcentaje inicial de infectados cuando ponemos a variar el tamaño de la población. La conjetura que queremos probar es si un mayor número de interacciones como las que se presentan con una población de agentes más densa impacta el porcentaje máximo de infectados. Nuevamente aclaramos que estudios más profundos pueden adelantarse para estudiar las interacciones de este factor junto con los demás, pero las limitaciones de tiempo hacen inviable un estudio más pormenorizado.

2. Resultados

Todas las experimentaciones se han realizado en R, y para el efecto se usó una computadora con sistema operativo Windows 10 Pro de 64bits y procesadorx64 Intel(R) core(TM) i5-4210U, CPU@ 1.70GHz 2.40GHz y memoria instalada de 6.00GB. Una llamada a la función *detectcores(logical=FALSE)* de R, permitió además constatar que la computadora posee dos núcleos físicos disponibles. A continuación se describe los fragmentos más importantes del código usado en cada experimentación, y los resultados numéricos y gráficos obtenidos.

2.1. Resultados para el problema 1

Los siguientes son los cambios efectuados sobre el código original, algunos de estos son cambios menores en la forma de plantear ciertas operaciones

- Cambio en la creación del data.frame de agentes inciales. Se hizo uso de la función *sample()* de R para muestrear aleatoriamente los agentes a infectar (y en el caso de la estrategia de vacunación a los agentes a vacunar con $p_v > 0$).

- Se paralelizó la construcción de los movimientos, es decir, que para un tiempo límite preestablecido, se llena con anterioridad y en forma paralela la matriz que contendrá todas las posiciones de los individuos en cualquier momento del tiempo dado. Note que acá no afectamos los estados de los agentes, sólo construimos sus posiciones para tenerlas disponibles para cualquier cálculo de distancias posterior. Se presume que esta mejora no ahorra mucho en los tiempos computacionales cuando la cantidad de agentes y la cantidad límite de tiempo es pequeño, y se reconoce también que la complejidad de la tarea es mucho menor que la de calcular los cambios de estado. No obstante se considera que por requerirse sólo un llamado a los núcleos de la máquina (y esto en forma previa al inicio de las labores pesadas de cómputo), no hay perjuicio alguno en paralelizar la actividad, y en cambio logramos algo de ganancia en tiempo sacrificando un poco de memoria disponible.

- Construcción de una matriz que contiene los diversos conteos de agentes por estados, y no solamente la cantidad de infectados en un momento dado. Esto con el fin de mejorar las salidas gráficas.

- Paralelización de contagios y actualización de estados, es decir, se diseña una función genérica que asigna a cada agente en el tiempo t su estado para el tiempo $t + 1$, siguiendo las lógicas de recuperación aleatoria con probabilidad p_r, e infección aleatoria basada en cercanía a infecciosos y distancia umbral de infección. La función es pasada como argumento a un comando *parSapply*, que recorre todo el vector de agentes en el tiempo t y actualiza en forma paralela su estado para el momento siguiente. Se considera que éste es el cálculo que más ahorro genera en tiempos computacionales puesto que en su versión secuencial habían involucrados varios ciclos *for*. Además la versión secuencial usaba el enfoque de dirigir la búsqueda de contagios desde los infecciosos a los susceptibles. En nuestro caso no hay preferencia de estados, simplemente un agente es tomado con independencia de su estado y se evalúa acorde a la lógica del proceso qué debe hacerse con él, en una forma paralela altamente eficiente.

- En las dos versiones del código se retira todo cálculo extraño al objeto mismo de simulación, es decir, se eliminan funciones que dibujan o controlan parámetros gráficos, la declaración de parámetros, y la carga de librerías. Esto con el fin de concentrarnos en medir específicamente la mejora en tiempos sobre aquellos cálculos importantes de la simulación[1].

Finalmente conviene mencionar la estrategia seguida para evaluar la ganancia en tiempo computacional de la versión paralelizada respecto a la no paralelizada:

- Ajustamos los parámetros de inicialización a los valores por defecto.

- Calculamos 10 réplicas de un escenario completo, es decir, corremos 10 veces la rutina para un $t_{max} = 100$, con $n = 50$ agentes.

[1] Como no se busca saturar el presente informe con fragmentos del código y la multiplicidad de cambios en él efectuados, se sugiere al lector consultar las versiones: *problema1ma.R* y *codigoor.R* para tener una idea de la implementación paralela y sencuencial respectivamente.

```
clusterExport(cluster,"tiempo")
clusterExport(cluster,"agentes")
estados_u<-as.character(agentes$estado)
clusterExport(cluster,"estados_u")
agentes$estado<-parSapply(cluster,1:n,function(i){
  a<-estados_u[i]
  if (estados_u[i]=="I" & runif(1) < pr)
  {
    a<-"R"
  }
  if (estados_u[i]=="S")
  {
    for (j in which(estados_u=="I"))
    {
      difx <- valores[i,tiempo+1]-valores[j,tiempo+1]
      dify <- valores[i+50,tiempo+1]-valores[j+50,tiempo+1]
      d <- sqrt(difx^2 + dify^2)
      if (runif(1)<((r-d)/r) & d<r)
      {
        a<-"I"
        break
      }
    }
  }
  return(a)
}
)
```

Figura 1: Rutina para asignar contagios en paralelo

- Aumentamos de 10 a 20 la cantidad de réplicas y estudiamos si hay diferencias en los resultados comparativos.
- Si las conclusiones extraídas por cada conjunto de réplicas no cambian significativamente, y los datos parecen alcanzar regularidad estadística, paramos el incremento en cantidad de réplicas, y nos decidimos a concluir.
- Corremos el experimento para 10 réplica más cada vez que no se alcance estabilidad en las conclusiones.

Seguir esta estrategia implicó correr primero 10 réplicas de cada versión del código, luego aumentar a 20 réplicas, hasta llegar a 40 finalmente, para finalmente concluir que la diferencia era tan notablemente alta que no era necesario realizar corridas adicionales para constatar la importante ganancia en tiempo de la versión en paralelo respecto a la versión secuencial. Los resultados se muestran en el boxplot comparativo de las figura 2. En promedio una corrida bajo implementación paralela se tarda 0.3 segundos en completarse, mientras que una implementación secuencial se tarda en promedio 13 segundos (43 veces más).

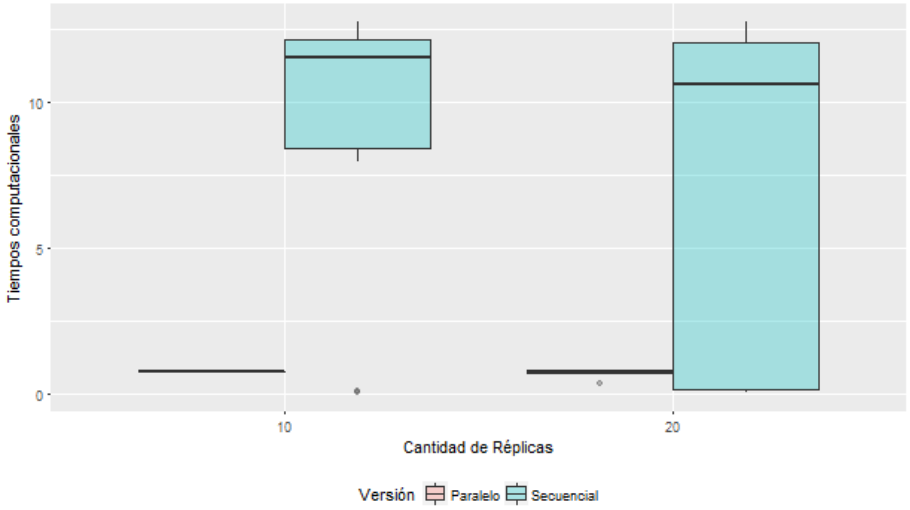

Figura 2: Comparación de tiempos entre implementaciones paralela y secuencial

2.2. Resultados para el problema 2

A continuación una descripción de la forma cómo se introdujo en el código la modificación del esquema de vacunación:

```
l <- 1.5
n <- 50
pi <-seq(0.00,0.10,0.01)
v <- l/30
pr<-0.02
pv<-seq(0.00,0.40,0.05)
tmax <- 100
r<-0.1

for (i in 1:length(pv))
{
   for (j in 1:length(pi))

   {
     probabilidades<-c(1-pi[j]-pv[i],pi[j],pv[i])
     tipos<-c("S","I","R")
     agentes<-data.frame(x = runif(n, 0, l), y = runif(n, 0, l),
                         dx = runif(n, -v, v), dy = runif(n, -v, v),
                         estado = sample(tipos,n,replace =T,
                             probabilidades))

     #...Las otras sentencias...
   }
}
```

Figura 3: Cambios en el código para involucrar probabilidad de vacunación

Como se puede apreciar, creamos dos vectores que contienen la información de las probabilidades de vacunación y probabilidades de infección que consideraremos para la experimentación. Y recorremos las posiciones de estos vectores

a través de ciclos *for* anidados. Las componentes de estos vectores son usadas dentro de un esquema de muestreo con reemplazamiento, vía construcción de un vector de probabilidades, el cual es pasado a la función *sample()* usada en la generación de la población inicial de agentes.

Los resultados de la experimentación, la cual usa 10 réplicas sobre cada combinación de los parámetros p_v y p_i, manteniendo constante los demás, se muestran en las gráficas **??** y **??**

Se concluye que para cada valor de probabilidad de infección inicial, hay un punto crítico de probabilidades de vacunación a partir del cual un porcentaje promedio muy mínimo infecciones se produce. Esto es de esperarse, en la medida en que los vacunados sirven de barrera para la propagación de la epidemia.

2.3. Resultados para el problema 3

A continuación se describe el código usado para estudiar el efecto de las probabilidades de infección sobre el porcentaje máximo de infectados en la población.

```
1  l <- 1.5
2  n <- c(50,100,150,200)
3  pi <-seq(0,0.40,0.05)
4  v <- l/30
5  pr<-0.02
6  pv<-0.02
7  tmax <- 100
8  r<-0.1
9
10 for (i in 1:length(n))
11 {
12    for (j in 1:length(pi))
13    {
14       probabilidades<-c(1-pi[j]-pv,pi[j],pv)
15       tipos<-c("S","I","R")
16       agentes<-data.frame(x = runif(n[i], 0, l), y = runif(n[i
             ],0,l),dx = runif(n[i], -v, v), dy = runif(n[i], -v, v
             ), estado = sample(tipos,n[i],replace = T,
             probabilidades))
17
18  #...Las otras sentencias...
19    }
20 }
```

Como puede notarse, la intención con la experimentación es evaluar el porcentaje máximo de infectados que se presentan ante diferentes niveles de probabilidad de infección y diferentes niveles de densidad poblacional (controlada vía valores diferentes de n). Los resultados del experimento se muestran en las gráficas **??** y **??**. Y permiten confirmar que la densidad poblacional, y no sólo la probabilidad inicial de infección, tienen impacto en los niveles máximos de infectados. Esto era de esperarse pues entre más densa sea la población mayor es la cantidad de interacciones posibles entre agentes que suceden a distancias cortas, y de ahí que aumente la probabilidad de que un agente en particular se contagie.

3. Conclusiones

Se concluye que las estrategias de paralelización pueden ahorrar en forma substancial esfuerzos de cómputo, y para el caso del sistema multiagente analizado, dicho ahorro es de hasta 43 veces el tiempo obtenido mediante programación secuencial. Por el lado del contexto de investigación, se concluye que la probabilidad de vacunación, puede disminuir substancialmente el porcentaje máximo de infectados, cuando las probabilidades de infección son bajas (no mayores a 0.1). Además, las probabilidades de infección así como el aumento de la densidad poblacional, son factores críticos a la hora de evaluar el porcentaje máximo de infectados. También se sugieren estudios más profundos para otras mezclas de valores paramétricos, pues las limitaciones tecnológicas impidieron estudios más exhaustivos.