Práctica 7: Búsqueda local

Simulación de sistemas

Marco Antonio Guajardo Vigil 2095

19 de marzo, 2019

1. Introducción

En la séptima práctica se implementa una optimización huerística sencilla para encontrar máximos locales de funciones. Se busca máximizar la función bidimensional $g(x, y)$ 1 :

$$g(x, y) = \frac{(x + 0{,}5)^4 - 30x^2 - 20x + (y + 0{,}5)^4 - 30y^2 - 20y}{100}, \tag{1}$$

a partir de un punto seleccionado al azar, realizando movimientos locales.

2. Objetivo

Se maximiza la función bidimensional, $g(x, y)$, con restricciones $-3 \leq x, y \leq 3$. La posición actual es un par x, y y se ocupan dos movimientos aleatorios, Δx y Δy, cuyas combinaciones posibles proveen ocho posiciones vecinos, de los cuales aquella que logra el mayor valor para g es seleccionado. Dibujando $g(x, y)$ en tres dimensiones obtenemos la figura 1.

Se crea una visualización animada de cómo proceden **15** réplicas simutáneas de la búsqueda encima de una gráfica de proyección plana.

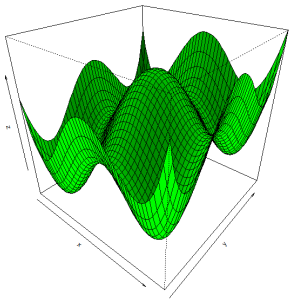


Figura 1: Modelo en tres dimensiones de la función $g(x, y)$.

2.1. Implementación de R

Para la elaboración de este experimento, se hace uso de un software libre para computación estadística y gráficos llamado R [1], el cual nos permite realizar los cálculos necesarios para dicho experimento. Con él, se pueden controlar los datos estadísticos que se ocupan para dar seguimiento con la práctica, se necesita graficarlos para así poder compararlos mejor, ya que se maneja una cantidad de datos considerable y trabajaremos con ellos en forma estadística.

2.2. Experimentación

Se crea un `data.frame()` llamado `coordenadas`, este almacena la mejor posición (x, y) obtenida entre los ocho vecinos, también se almacena el número de pasos en el cual se obtuvo ese valor, se establecen los límites (`low` y `high`) mencionados anteriormente en el objetivo, el punto avanza en pasos de 0.25 declarado en `step`.

```
low <- -3
high <- -low
step <- 0.25
replicas <- 15
coordenadas <- data.frame("X"=0, "Y"=0, "T"=0, "R" = 0)
animacion <- FALSE
```

Se modifica el código obtenido de la Dr. Elisa Shaeffer [3] que inicialmente obtenia los mínimos de una función unidimensional, de tal modo que permita encontrar los máximos locales de la función bidimensional $g(x, y)$, estas modificaciones se presentan en el siguiente código:

```
replica <- function(t, coor) {
  currX <- runif(1, low, high)
  currY <- runif(1, low, high)
  bestX <- currX
  bestY <- currY
  for (tiempo in 1:t) {
    deltaX <- runif(1, 0, step)
    deltaY <- runif(1, 0, step)
    izq <- currX - deltaX
    der <- currX + deltaX
    abajo <- currY - deltaY
    arriba <- currY + deltaY
    while(sum(c(izq, der, abajo, arriba) < low) !=0 || sum(c(izq, der, abajo, arriba) > high) != 0){
      currX <- runif(1, low, high)
      currY <- runif(1, low, high)
      deltaX <- runif(1, 0, step)
      deltaY <- runif(1, 0, step)
      izq <- currX - deltaX
      der <- currX + deltaX
      abajo <- currY - deltaY
      arriba <- currY + deltaY
     b = data.frame("X" = bestX, "Y" = bestY, "T" = tiempo, "R" = i)
     coor <- rbind(coor, b)
    }
  best = c(bestX, bestY)
  return(coor)
}
```

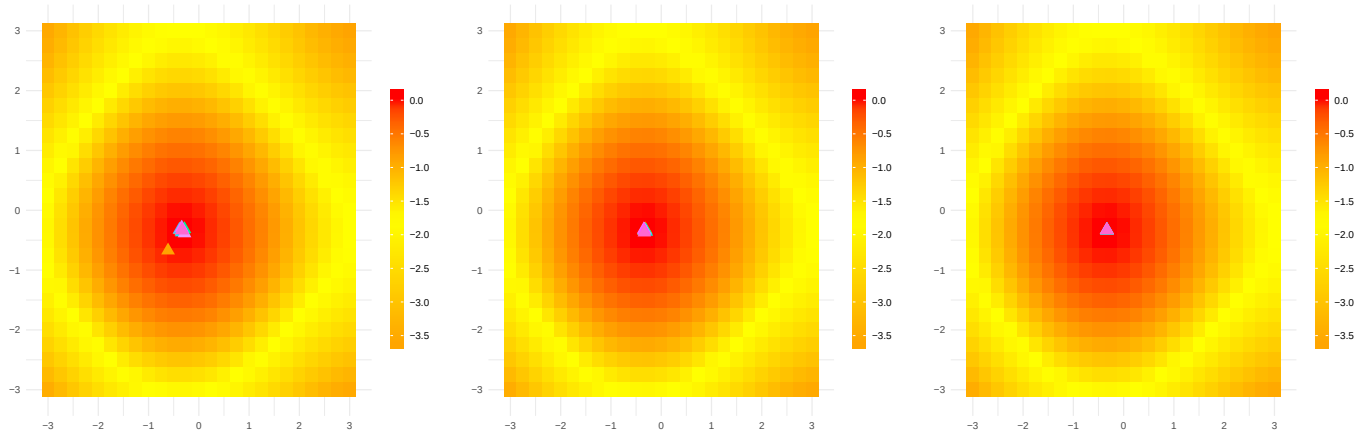
Se paralelizan las réplicas para realizarlas simultáneamente y se crean figuras en donde se muestran cada una de las réplicas buscando el mayor vecino, hasta llegar a la máxima posición.

```
suppressMessages(library(doParallel))
registerDoParallel(makeCluster(detectCores() - 1))
values <- seq(low, high, by = step)
x <- rep(values, each = length(values))
y <- rep(values, length(values))
z <- foreach(i = x, j = y, .combine=c) %dopar% g(i,j)
resultados2 <- data.frame(x, y, z)
tmax <- 10^4
resultados <- foreach(i = 1:replicas, .combine="rbind") %dopar% replica(tmax, coordenadas)
resultados <- data.frame(resultados)
stopImplicitCluster()
resultados <- resultados[!(resultados$T == 0),]
library(ggplot2)
for (i in 1:tmax){
  sS <- subset(resultados, T == i)
  if(animacion){
    ggsave(paste("paso", i, ".png", sep=""))
    ggplot(resultados2, aes(x = x, y = y)) + geom_tile(aes(fill=z)) + ggtitle(paste("Paso ", i, sep="")
        ) +
      scale_fill_gradient2(name = "",low = "orange", mid = "yellow", high = "red", midpoint=(min(z)+max
          (z))/2, breaks = c(seq(floor(min(z)), 0, by = 0.5))) +
      scale_x_continuous("", breaks = seq(low, high, by = 1)) + scale_y_continuous("", breaks = seq(low
          , high, by = 1)) +
      guides(fill = guide_colorbar(barwidth = 1, barheight = 20)) + geom_point(data = sS, aes(x= X, y=
          Y, color = as.factor(R)), size = 4, shape = 17, stroke = 2) + scale_color_hue(l=80, c=150,
          guide = FALSE) +
      theme_minimal(base_size = 14)
    graphics.off()
  }
}
```

2.3. Resultados y conclusiones

Se obtuvo un `gif` ("busqueda_animacion.gif") donde muestra el movimiento de cada réplica en un período de 100 pasos, localizado en el repositorio donde se encuentra esta práctica [2].

En la figura 2, tomando en cuenta que lo naranja indica los mínimos de la función $g(x, y)$ y lo rojo el máximo, se observa que con 100 pasos la mayoría de las réplicas logra acercarse al máximo de la función, cuando se realizan 10,000 pasos casi todas las réplicas coinciden en el punto máximo.



(a) Busqueda local de máximos en una proyección plana (x, y), con 100 pasos.

(b) Busqueda local de máximos en una proyección plana (x, y), con 1000 pasos.

(c) Busqueda local de máximos en una proyección plana (x, y), con 10000 pasos.

Figura 2: Comparación de las réplicas a mayor cantidad de pasos.

Referencias

[1] The R Project for Statistical Computing. 2019. URL `https://www.r-project.org/`.

[2] Guajardo Vigil Marco Antonio. Simulación de sistemas. 2019. URL `https://sourceforge.net/projects/simulaciondesistemas/`.

[3] Schaeffer, E. Práctica 7: Búsqueda local. 2019. URL `https://elisa.dyndns-web.com/teaching/comp/par/p7.html`.