Práctica 7: búsqueda local

26 de septiembre de 2017

1. Introducción

La parte primordial de la tarea consiste en maximizar la función *g(x,y)*, y ajustar el código previamente otorgado utilizando la misma lógica que en el ejemplo unidimensional.

El Reto uno consiste en mostrar de algún modo la forma en que se lleva a cabo la búsqueda local, es decir ejemplificar de modo gráfico como avanzan los puntos a través de las iteraciones. El reto dos es modificar el código de tal modo que la búsqueda local se convierta en un recocido simulado, es decir que acepte moverse la solución a valores peores con el fin de intentar salir de algún máximo local.

2. Parámetros de trabajo

La experimentación se realizó en una MacBook Pro la cual cuenta con cuatro núcleos disponibles, se establecieron los parámetros de tal modo que se obtuvieran cuatro vecinos para la búsqueda local, se decidió plasmar los puntos en un plano *(x, y)* para que la apreciación de los puntos fuera de forma adecuada y el paso que se le permitía a los agentes moverse se fijó en 0.005 con el fin de poder observar las diversas búsquedas en todos los puntos.

3. Modificaciones del código

De el código original se re formuló el modo de evaluación de la función que realiza las réplicas, ya que solo estaba adecuada para una función unidimensional, así como la forma de evaluar directo en la función g. Se conservaron los parámetros establecidos en el código original como lo son los valores en los cuales se va a mover la función y la cantidad de puntos se fijó en cien. La cantidad de réplicas se dejó fijo en tres valores distintos $10^2, 10^3, 10^4$.

```
replica <- function(t) {
curr <- c(runif(1, low, high),runif(1, low, high))
best <- curr
for (tiempo in 1:t) {
delta <- runif(1, 0, step)
pnts<- c(curr[1]+delta,curr[2],curr[1]-delta,curr[2],
curr[1],curr[2]+delta,curr[1] ,curr[2] -delta,curr[1],curr[2])
valores <- c(g(pnts[1],pnts[2]),g(pnts[3],pnts[4]),g(pnts[5],pnts[6]),
g(pnts[7],pnts[8]),g(curr[1], curr[2]))
v <- which.max(valores)
curr<- c(pnts[(v*2)-1],pnts[v*2])
if (g(curr[1],curr[2]) > g(best[1],best[2])) {
best <- curr
} }
return(best)
}
```

4. Resultados

Se optó por conservar la cantidad de réplicas que se realizaban y de ese modo se obtuvieron las figuras 1,2 y 3.

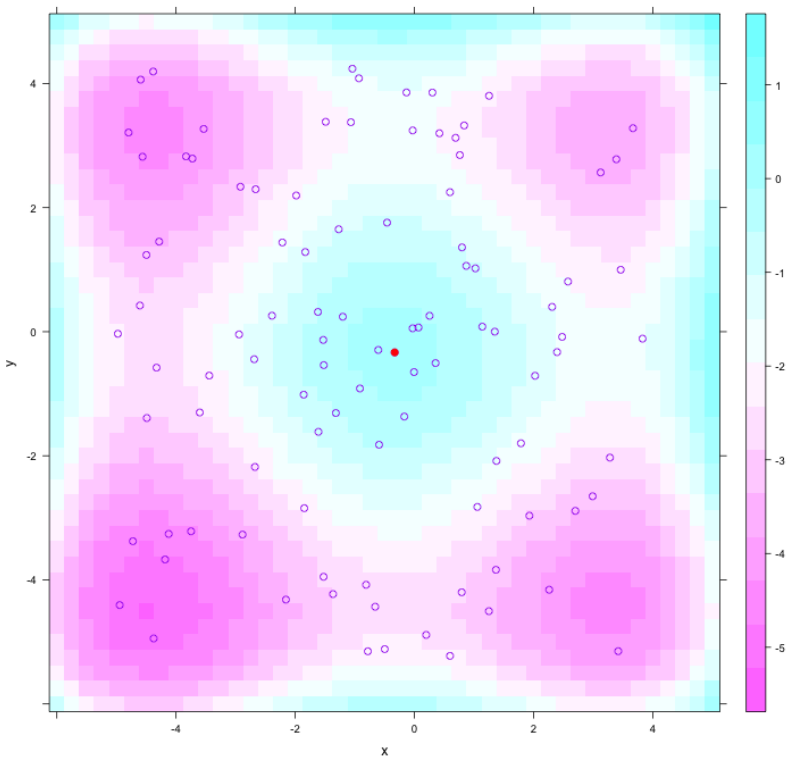

Figura 1: Prueba con 100 réplicas.

4.1. Interpretación

Podemos observar que si aumentamos la cantidad de pasos que la prueba realiza, es decir si dejas que la búsqueda local aumente la cantidad de iteraciones que se mueve, los puntos van avanzando a donde se esperaría, en nuestro caso a las zonas de color rosa.

5. Reto 1

Para el reto uno se realizó un GIF con el fin que se pudiera apreciar de una manera más clara lo que ocurre en una iteración de búsqueda local, de igual modo se incluyen la figura 4 y figura 5 con el fin de ilustrar que es la misma función y solo se habla de una búsqueda local la que se ejemplificará.

Dado que los pasos son en teoría largos se puede apreciar un poco el cambio de punto en la búsqueda.

6. Reto 2

El reto dos consiste en cambiar la búsqueda local por un recocido simulado, así como poder definir que parámetros serán los adecuados para la realización del recocido respecto a temperatura inicial y tamaño del paso de enfriamiento. para eso se realizaron varias pruebas de las cuales se llegó a la conclución que los parámetros adecuados para la búsqueda sería una temperatura inicial de 200 así como una multiplicación de la temperatura por un $\alpha = .99$.

En la figura 6 podemos observar la disminución de la temperatura con la cual se decidió el parámetro a mejor considerar para el recocido simulado.

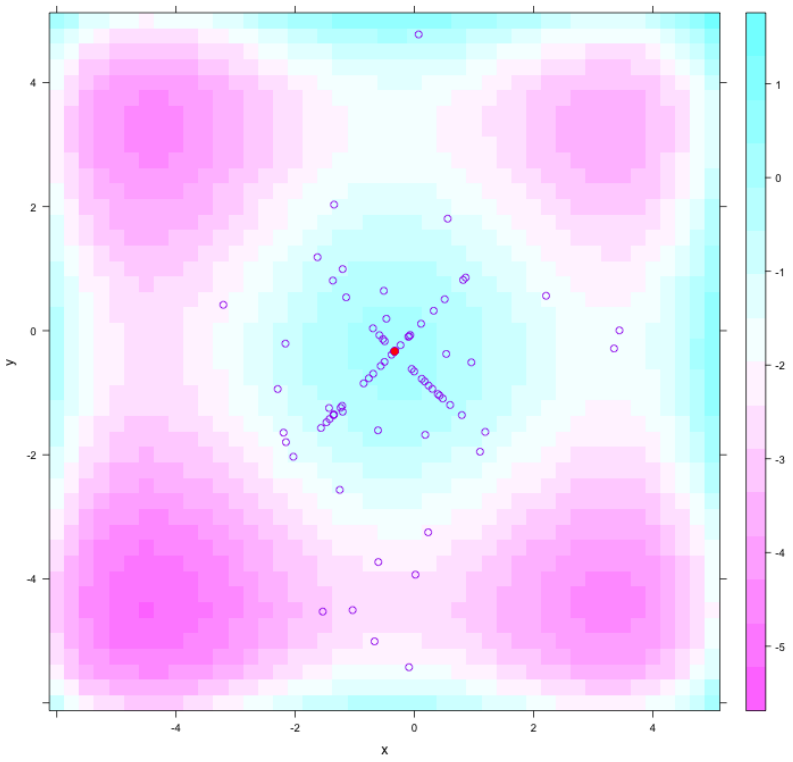

Figura 2: Prueba con 1000 réplicas.

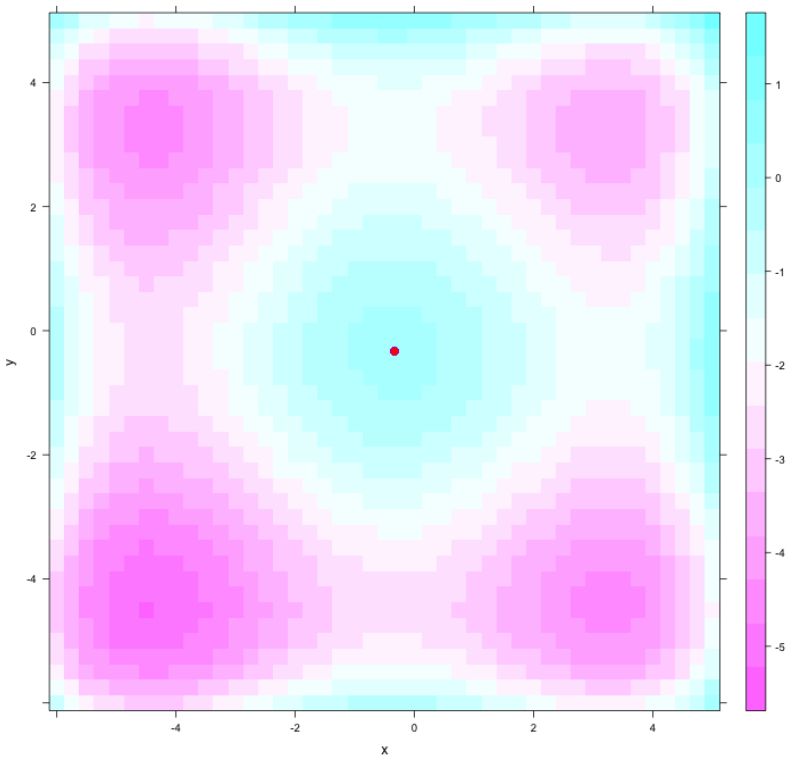

Figura 3: Prueba con 10000 réplicas.

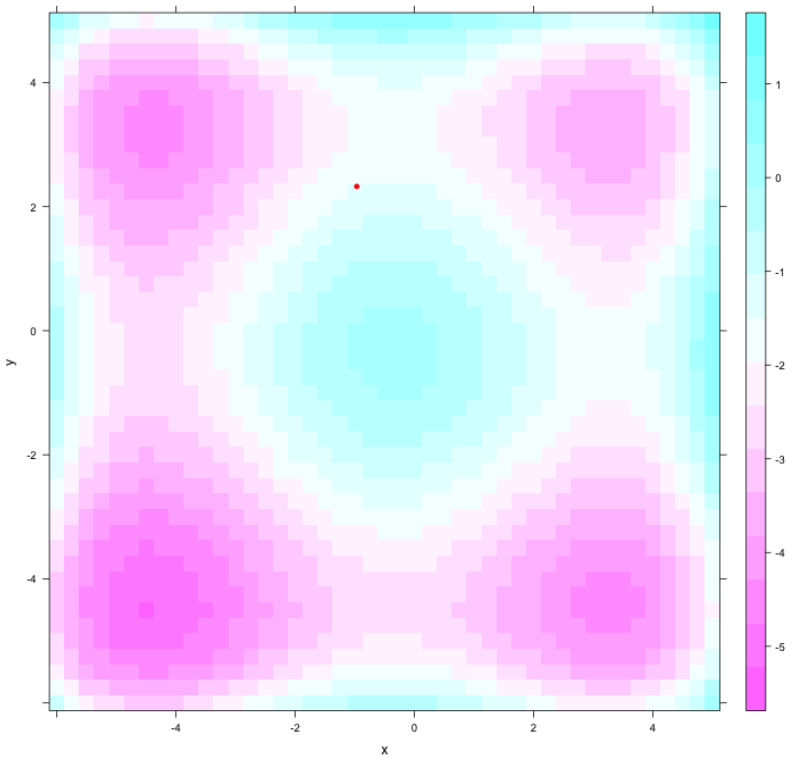

Figura 4: Paso número 4

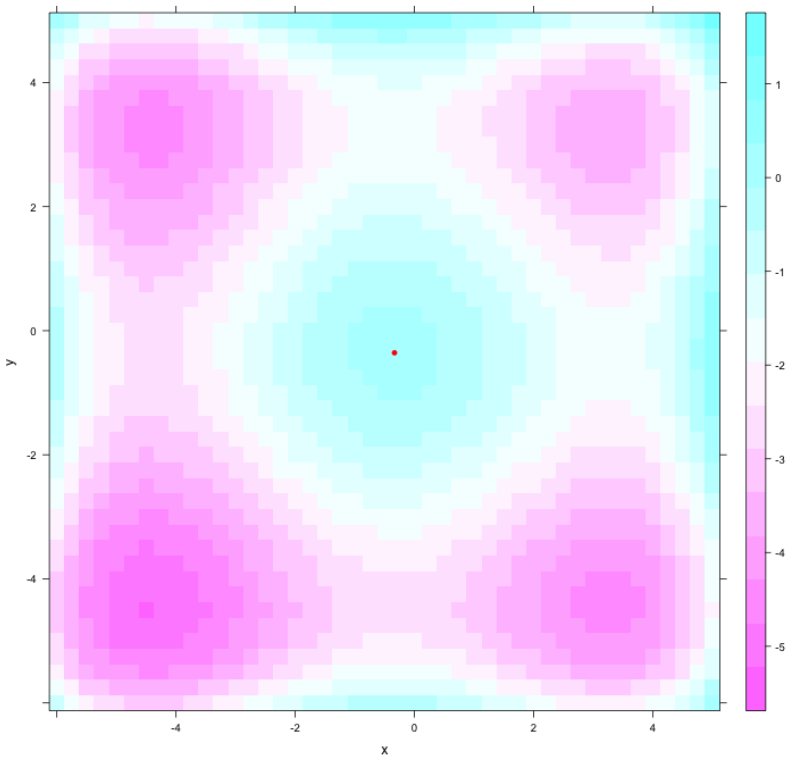

Figura 5: Paso número 47

7. Modificaciones del código para recocido simulado

Una característica por la cual se opta por trabajar con el recocido simulado es que dicho tipo de búsqueda te permite salir de máximos o mínimos, dependiendo del caso, locales. La modificación

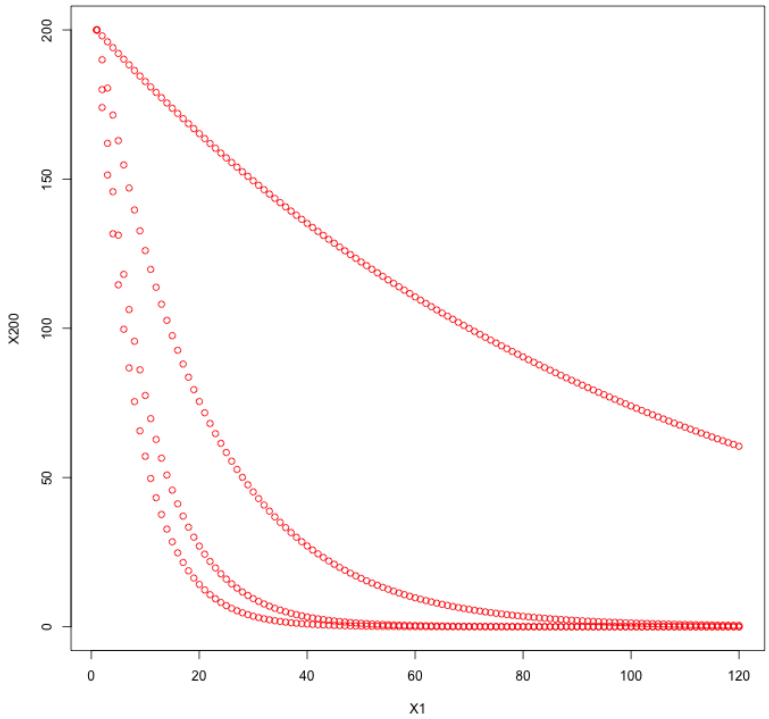

Figura 6: Iteraciónes respecto al valor de la función g.

que se le realizó al código se muestra a continuación, solo se incluye la parte propia del recocido con el fin de que quede claro el modo en que se definió la reducción de temperatura en cada iteración.

```
else{
e <- exp(del/temperatura)
if(runif(1,0,1) < e){
curr <- c(pnts[(v*2)-1],pnts[v*2])
}
}
if(g(curr[1],curr[2]) > g(best[1],best[2])){
best <- curr
}
temperatura <- temperatura*(.8)
varios <- c(best,curr,temperatura)
}
```

De igual forma se incluye un GIF con el fin de ilustrar a que se refiere que con cierta probabilidad se tomen soluciones no mejores para la búsqueda es decir, esto permite una mayor inspección en el área de las soluciones. Podemos observar en el GIF de recocido, como la solución parece irse a valores peores y al transcurso de las iteraciones vemos como regresa hacia las mejores soluciones ya que deja de tomar aquella que no le convengan en valor.