Reporte práctica 7
Búsqueda local

1. INTRODUCCIÓN

En el modelo de búsqueda local se evalúa la calidad de una solución en un punto determinado. La presente práctica realiza la búsqueda de máximos y mínimos de una función partiendo de un punto al azar para sus variables dadas.

El código ejemplo busca minimizar una función unidimensional, a partir de un punto seleccionado al azar, realizando movimientos locales en el eje *x*. Compara la posición actual con una posible posición determinada mediante un delta al azar. Si la función en el valor actual es mayor que el posible cambio entonces se reemplaza el valor actual con el cambio, si sucede lo contrario el cambio es rechazado y el valor actual no se mueve.

2. SIMULACIÓN

Se inicia la simulación cambiando de una función unidimensional base a una bidimensional, para esto es necesario cambiar los parámetros de decisión del movimiento ya que ahora tendremos cuatro posibles cambios o vecinos (arriba y abajo en el eje *x*, o izquierda y derecha en el eje *y*).

En el código que se muestra es de la función que controla la decisión del movimiento en la que se comparan todos los posibles vecinos determinando el mejor candidato, al final compara contra su posición misma y al resultar una mejora toma el cambio, de lo contrario no se mueve, esto se repite un número de pasos determinado en la variable *t*.

```
> po<-function () {
+       datos=data.frame()
+       resultados=data.frame()
+       currx <- runif (1, low, high)
+       curry <- runif (1, low, high)
+       best <- c (currx, curry)
+     for (tiempo in 1:t) {
+         delta <- runif(1, 0, step)
+         left <- currx - delta
+         right <- currx + delta
+         down <- curry - delta
+         up <- curry + delta
+         if (f(left,curry) > f(right,curry)) {
+           bestx <- c(left,curry)
+         } else {
+           bestx <- c(right,curry)
+         }
+         if (f (currx, up) > f(currx, down)) {
```

```
+           besty <-c(currx,up)
+         } else {
+           besty <- c(currx,down)
+         }
+         if(f (bestx[1],bestx[2])> f(besty[1], besty[2])){
+           currx<-bestx [1]
+           curry<-bestx [2]
+         }else{
+           currx<-besty [1]
+           curry<-besty [2]
+         }
+         if (f (currx, curry) > f(best [1] ,best[2])) {
+           best <- c(currx,curry)
+         }
+         datos=cbind (i,tiempo, currx, curry,f(best[1], best[2]))
+         resultados=rbind(resultados,datos)
+       }
+       return(resultados)
+   }
```

Se grafica mediante la librería *lattice* diez corridas de treinta pasos cada una, agregando al gráfico una línea base del máximo de la función representado por la variable *WA*, obtenido de la página web wolframalpha.

3. RESULTADOS

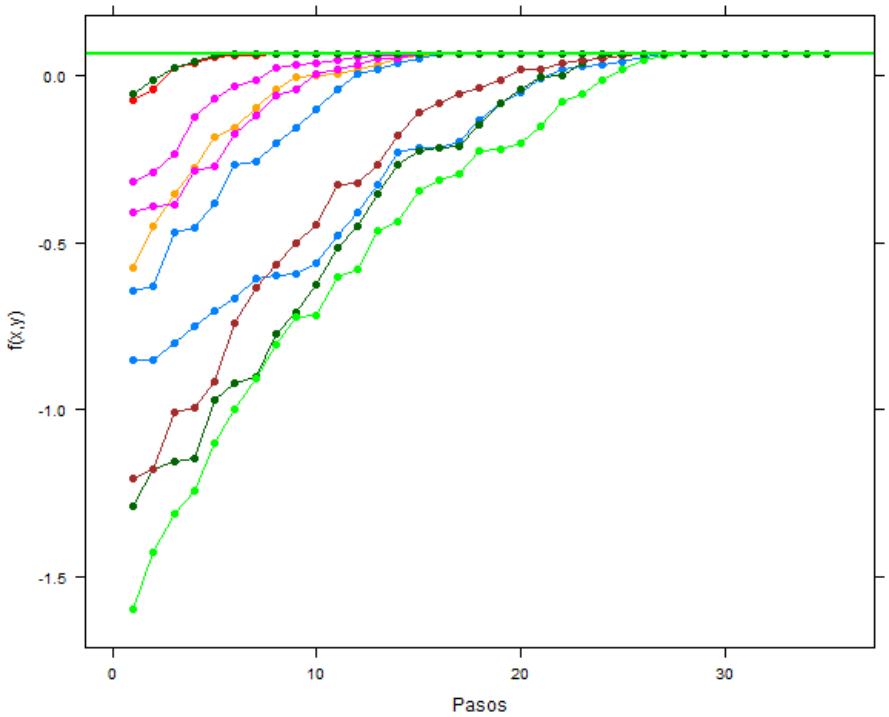


Figura 1. Gráfica de puntos de las diez réplicas del código de treinta pasos cada una, marcando con una línea verde el máximo dado por wolframalpha.

Cada réplica o paso inicia en un punto al azar y va aumentando en base a las condiciones dadas en el código hasta llegar al punto máximo como se muestra en la figura 1 en dónde todas las réplicas llegan al valor dado en wolframalpha.

4. RETOS

- Reto 1

Para el reto uno se solicita que la búsqueda del máximo local implementada en la tarea base esquematice en una proyección plana de la función. Se tomó como base la función anteriormente implementada y se graficó en plano con la ayuda de la librería *ggplot2* (código en azul), posteriormente se utilizó las reglas de movimiento impuestas en la tarea base (código en rojo) y se sobrepuso al gráfico en plano, los puntos de posición de los buscadores en cada paso (código en verde).

```
> library(ggplot2)
> g1<-ggplot(d, aes(x, y)) +
+   geom_raster(aes(fill =z))+
+ scale_fill_gradient (low="red", high="white")

> names(y)=c ("corrida","tiempo","posx","posy", "f(x)")
> y$corrida=as.factor(y$corrida)
> for (tiempo in 1:t {
+   g<-y[y$tiempo==tiempo,]
+   g1+geom_point (data=g, aes (g$posx,g$posy, color=corrida))
+   #ggsave (paste ("p7R1", tiempo,".png"))
+  }
```

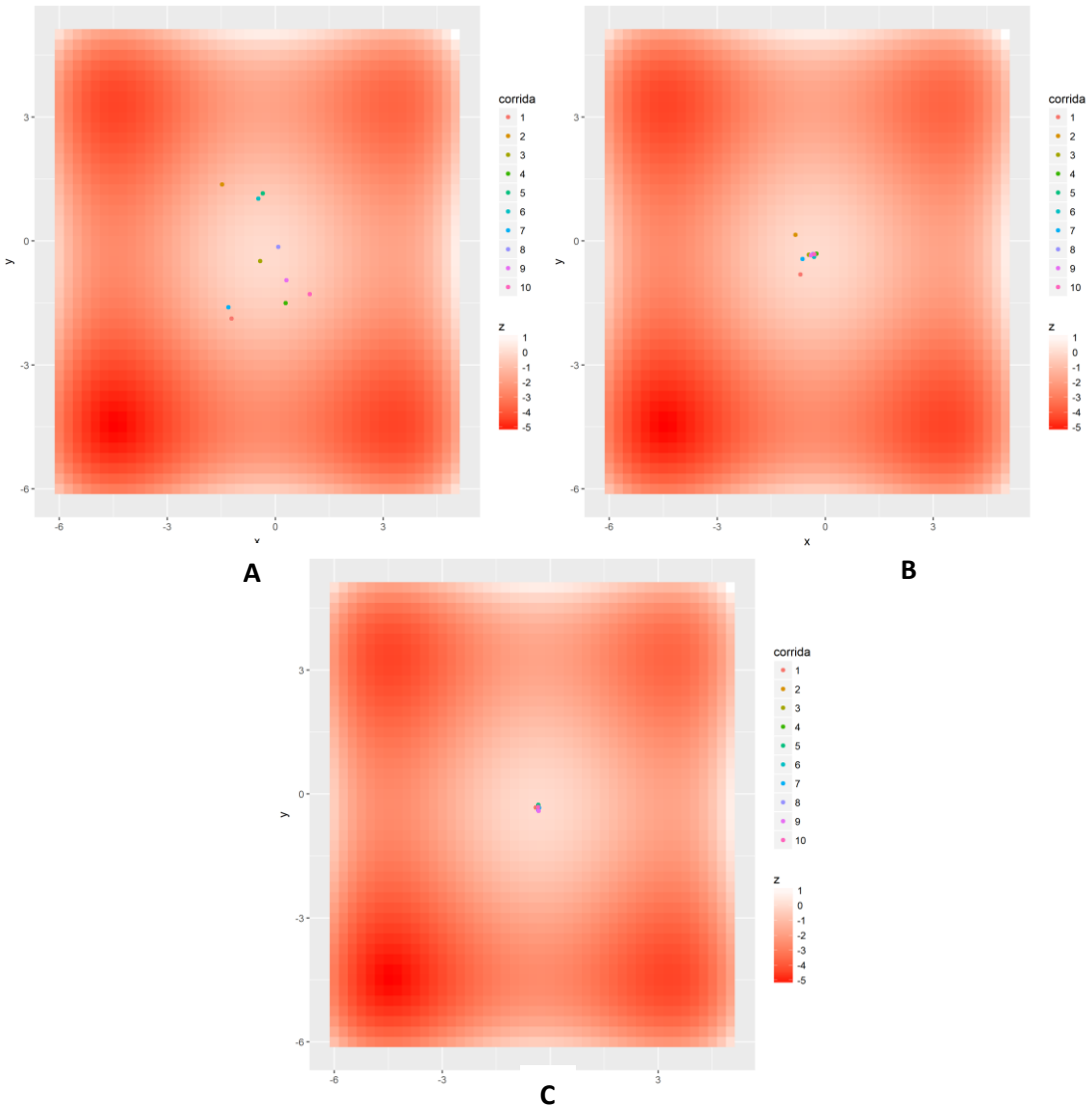


Figura 2. Mapa de calor de la función con las diez corridas (puntos de colores). Puntos máximos en blanco y mínimos en rojo. Imágen **A**, primer paso; Imágen **B,** paso quince; y **C**, paso 33.

En las gráficas planas o mapas de calor de la función (figura 2) se pueden observar que los puntos máximos se encuentran en el centro y en las aristas del margen de la imagen, encontrando el mayor valor (el proporcionado por wolframalpha) en el centro.

En la figura 2, también se observan diez puntos de colores correspondientes a cada corrida realizada. Los puntos inician en posiciones al azar (figura 2A) y conforme corren los pasos se van acercando al máximo central (figura 2C).

- Reto 2

En el reto dos se fija como objetivo modificar la regla de movimiento basándonos en el **recocido simulado** en dónde solamente se tendrá un vecino posible (*xpr*) que es igual a la posición actual (*x*) más delta, que en este caso puede tomar valores negativos.

Después de calcula la diferencia en función de *x* y *xpr* si es mayor que cero siempre se acepta al vecino ya que representa una mejora. Si no es mayor de cero entonces se calcula una probabilidad de aceptación en función de la temperatura, la cual va decreciendo en torno a un valor *j* cada que se acepta una empeora.

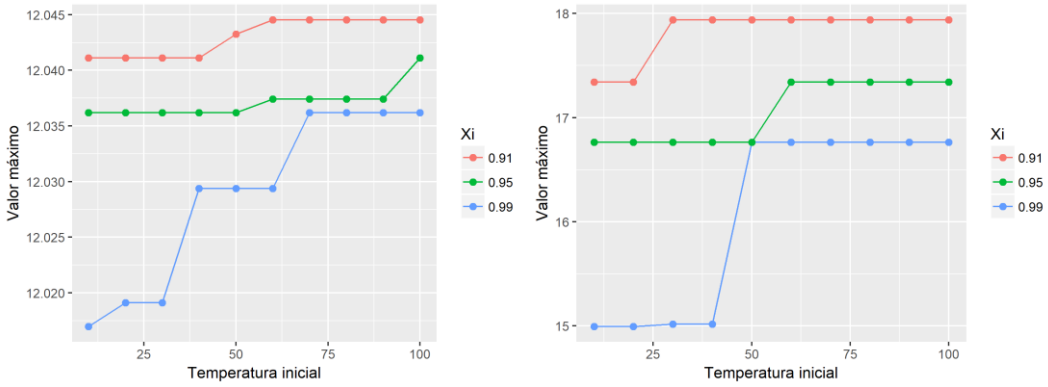
Para lograr lo anterior se utiliza la función unidimensional dada en la práctica, se agregan dos *for* para variar *j* y temperatura (*temp*), y se cambia la regla de movimiento con lo requerido (lo marcado en turquesa en el código).

Para la comparación del vecino se utiliza la variable *d*, se agrega la variable *p* para determinar la probabilidad de aceptar la empeora y por último *temp* se modifica con *j* al aceptar una empeora, todo esto se marca con verde en el código.

Se modifica el límite máximo que puede tomar delta en 0.25 y 0.1. Para finalizar se grafica el valor máximo obtenido contra la temperatura inicial de cada código con diferente delta con la herramienta *ggplot2* usando el código marcado en gris.

```
> for(j in c(0.99,0.95, 0.91)){
+ for(temp in seq(10,100,10)){
+   reg=cbind(temp)
+ for (tiempo in 1:t) {
+   x <- runif(1, low, high)
+   best=x
+   delta <- runif(1, 0, step)
+   xpr<- x + delta
+    d=f(xpr)-f(x)
+    p=exp(-d/temp)
+
+   datos=cbind(tiempo,x,xpr,temp)
+   if (d > 0) {
+     x=xpr
+   } else {
+       if (runif(1) < p) {
+        x=xpr
+       temp=temp*j
+     }
+
+   }
+
+   datos=cbind(datos,d,temp,p,x,f(x))
+   resultados=rbind(resultados,datos)
+ }
+   mejor=cbind(j,reg,temp,max(resultados[9]))
+  paso=rbind(paso,mejor)
+    }
+ }
> names(paso)=c("Xi","Tempi","Temperatura","Vmaximo")
> paso$Xi=as.factor(paso$Xi)
```

```
> library(ggplot2)
>
> g1<-ggplot(paso, aes(x=Tempi, y=Vmaximo,color=Xi)) +
+   geom_line() +
+   geom_point(size=2)+
+   xlab("Temperatura inicial") + ylab("Valor máximo")
> #ggsave("p7R2.png")
```



A) **B)**

Figura 3. Gráfica puntos de valor máximo de la función en relación con la temperatura inicial a tres valores de *j (Xi)* 0.99, 0.95 y 0.91. Se probaron dos valores máximos para delta, 0.25 (A) y uno (B).

Se observa que al aumentar el valor máximo de delta (figura 3B) los valores máximos que puede tomar la función son mayores y que los valores máximos varían menos en cuánto sube la temperatura (figura 3).
Al aumentar el rango en que disminuye la temperatura (*Xi*) existe mayor variabilidad en los datos del valor máximo que puede tomar la función.
Para compara se utilizó el código base de una función unidimensional, pero se modificó a que tuviera un solo vecino y sólo se aceptara en caso de ser una mejora, tal y cómo se visualiza en el código en morado.

```
> for (tiempo in 1: tmax) {
+   delta <- runif(1, 0, step)
+   xpr<- x + delta
+   fxpr<- f(xpr)
+   fx <- f(x)
+   if (fxpr > fx) {
+     x <- xpr
+   }
```

Para este caso también se varió el máximo de delta pero en 0.1, 0.25 y 1. Se corrió el código para cien pasos obteniendo en cada gráfica rangos diferentes para los valores de *f(x)*, de 7.5 a 8.2 para 0.1; 0 a 100 para 0.25; y 0 a 4000 para 1. Lo que nos dice que entre mayor pueda ser el cambio mayor será el valor para la función evaluada en *x*.
Además, se pueden observar diferencias en el comportamiento de búsqueda de cada corrida. Mientras que para deltas pequeños se observan un gran salto en los primeros pasos (figura 4A) que se mantiene sin gran variación hasta al final; para los deltas más grandes (figura 4A y 4B) se mantiene una línea constante hacia la mejora de los valores. Esto nos podría decir que en deltas pequeños es más difícil conseguir una mejora así que se estanca en un máximo local sin poder avanzar.

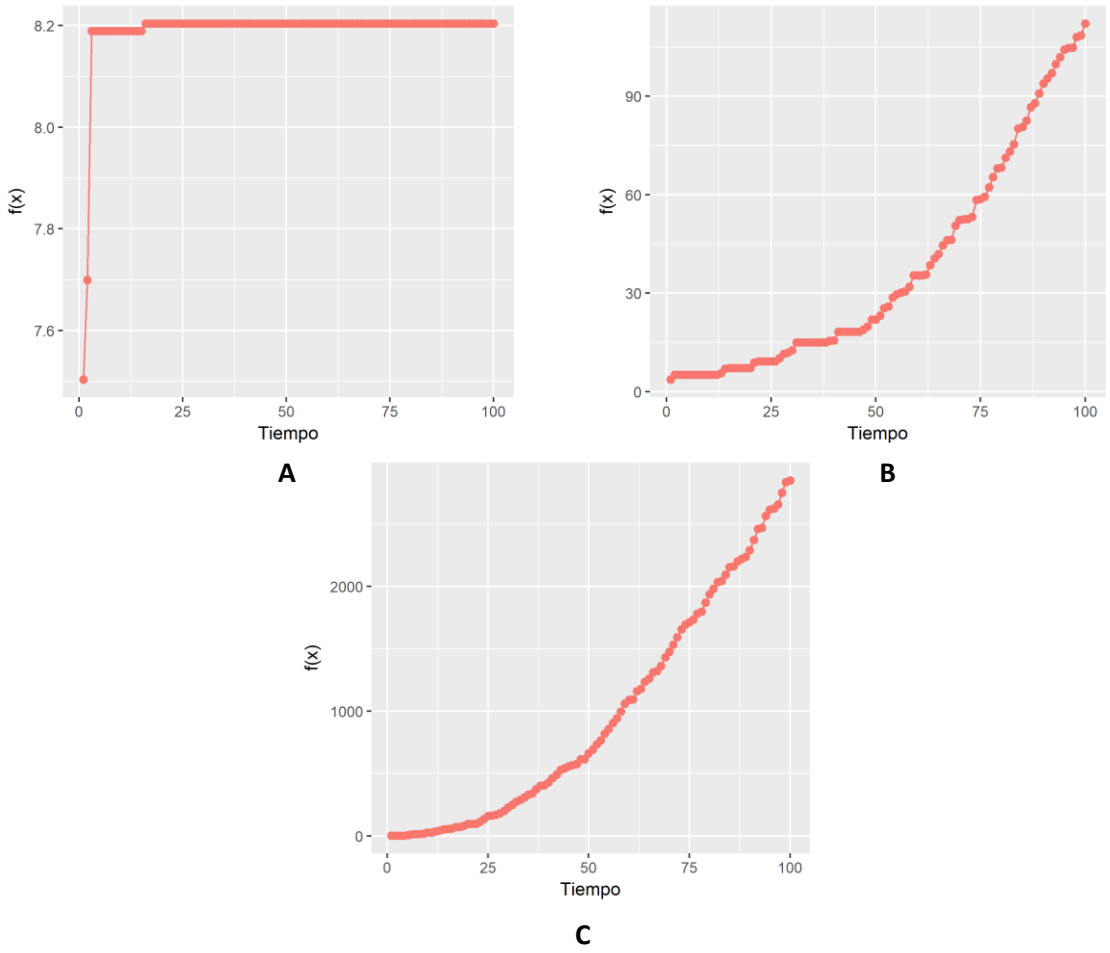


Figura 4. Gráfica de puntos de los valores de *f(x)* contra tiempo para cien pasos. Variando máximos para delta: 0.1 (**A**); 0.25, Imágen **B**; y 1 (**C**).

5. CONCLUSIONES

En base a lo observado en la figura 1 se puede concluir que el código se modificó exitosamente para determinar los valores máximos de la función bidimensional dada en la práctica.

Se obtuvo el mapa de calor o representación en plano de la función bidimensional.

Se esquematizó la búsqueda sobre el plano de la función con diferentes réplicas cayendo todas en el máximo valor al pasar 33 pasos.

Se modificaron los criterios de cambio por los del **recocido simulado** en dónde se puede aceptar una empeora con cierta probabilidad para poder mejorar posteriormente.

En ambos casos (búsqueda original y recocido simulado) se incrementan los valores que puede tomar la función al aumentar delta.

Si disminuimos la tasa en que baja la temperatura (*Xi*) y disminuimos delta existe menor variabilidad en los valores máximos para el recocido simulado.

Al minimizar el valor de delta existe menos probabilidad de cambio en la función y los valores máximos que toma son menores para el código original.