Práctica 7: Búsqueda local

En la práctica búsqueda local se busca el valor máximo o mínimo local de una función dada. Analíticamente se encuentran dichos valores al derivar la función `f(x)` y encontrar los valores críticos dónde `f'(x)=0.` Sin embargo, existen métodos más versátiles los cuales se aproximan al valor obtenido analíticamente.

Meta

Maximizar la función bidimensional `f(x,y)` (Ecuación 1) tomando las coordenadas de la posición actual se generarán las posiciones de cuatro vecinos utilizando diferenciales, Δx y Δy, generados al azar y se actualizará la posición actual con la posición del vecino cuya evaluación de la función bidimensional `f(x,y)` sea mayor.

$$f(x,y) = \frac{(x+0.5)^4 - 30x^2 - 20x + (y+0.5)^4 - 30y^2 - 20y}{100} \qquad (1)$$

Desarrollo del código

Para el desarrollo de este código se utilizó el código ejemplo utilizado para minimizar una función unidimensional [1]. Se discutirá únicamente la función `paralelo(w)` la cual realiza las búsquedas locales, las líneas siguientes a la mencionada función activan la realización de cálculos de manera paralela con el paquete `do.Parallel` y crea la gráfica necesaria para mostrar los resultados de este experimento. El código completo puede dentro de este repositorio bajo el nombre "Cod_P7_T7".

Las modificaciones realizadas tienen por objetivo:

1. Genera la posición actual para el eje x y el eje y.
2. Generar un vector `best` con el objetivo de guardar la posición del valor máximo en los valores dentro del `for paso.`
3. Generar un diferencial para ambos ejes de manera aleatoria, los cuales toman el nombre de `delta.x` y `delta.y.`
4. Generar las nuevas posiciones para los cuatro vecinos `(left,right,up,down).`
5. Comparar y tomar las posiciones del vecino más alto
    a. Compara el valor de la `función f` para las posiciones vecinas en el eje horizontal.

b. Compara el valor de la función f para las posiciones vecinas en el eje vertical.
c. Compara el valor de la función f para las posiciones vecinas con el valor más alto en ambos ejes.

6. Compara la posición (curr.x,curr.y) contra las posición best al evaluarlas en la función f y guarda aquella posición con la cual se obtenga el valor más alto.
7. Guarda las posiciones para el for paso en el data.frame() resultados.

```
> paralelo <-function(w){
+
+   curr.x <-runif(1, low, high)
+   curr.y<-runif(1, low, high)
+   best <- c(curr.x,curr.y)
+   datos<-data.frame()
+   resultados<-data.frame()
+
+   for (paso in 1:tmax) {
+
+
+     delta.x <- runif(1,0,step)
+     delta.y <- runif(1,0,step)
+
+
+     left <- curr.x - delta.x
+     right <- curr.x + delta.x
+     up<-curr.y + delta.y
+     down<-curr.y-delta.y
+
+
+
+     if (f(left,curr.y) > f(right,curr.y)) {
+       best.x<- c(left,curr.y)
+     } else {
+       best.x<- c(right,curr.y)
+     }
+
+
+     if (f(curr.x,down) > f(curr.x,up)) {
+       best.y<-c(curr.x,down)
+     } else {
+       best.y<-c(curr.x,up)
+     }
+
+         if (f(best.x[1],best.x[2]) > f(best.y[1],best.y[2])) {
+       curr.x<- best.x[1]
+       curr.y<- best.x[2]
+     } else {
+       curr.x<- best.y[1]
+       curr.y<- best.y[2]
+     }
+
+
```

```
+       if (f(curr.x,curr.y) > f(best[1],best[2])) {
+       best[1]<-curr.x
+       best[2]<-curr.y
+
+     }
+
+     datos<-cbind(curr.x,curr.y,f(curr.x,curr.y),w,paso)
+     resultados<-rbind(resultados,datos)
+   }
+   return(resultados)
+ }
```

Resultados

En la figura 1 se observan los resultados obtenidos para diez repeticiones de este experimento. En esta figura se observa que todas las repeticiones llegan a un máximo en $f(x,y) \approx 0.0666822$ [2] en un máximo de 25 pasos.

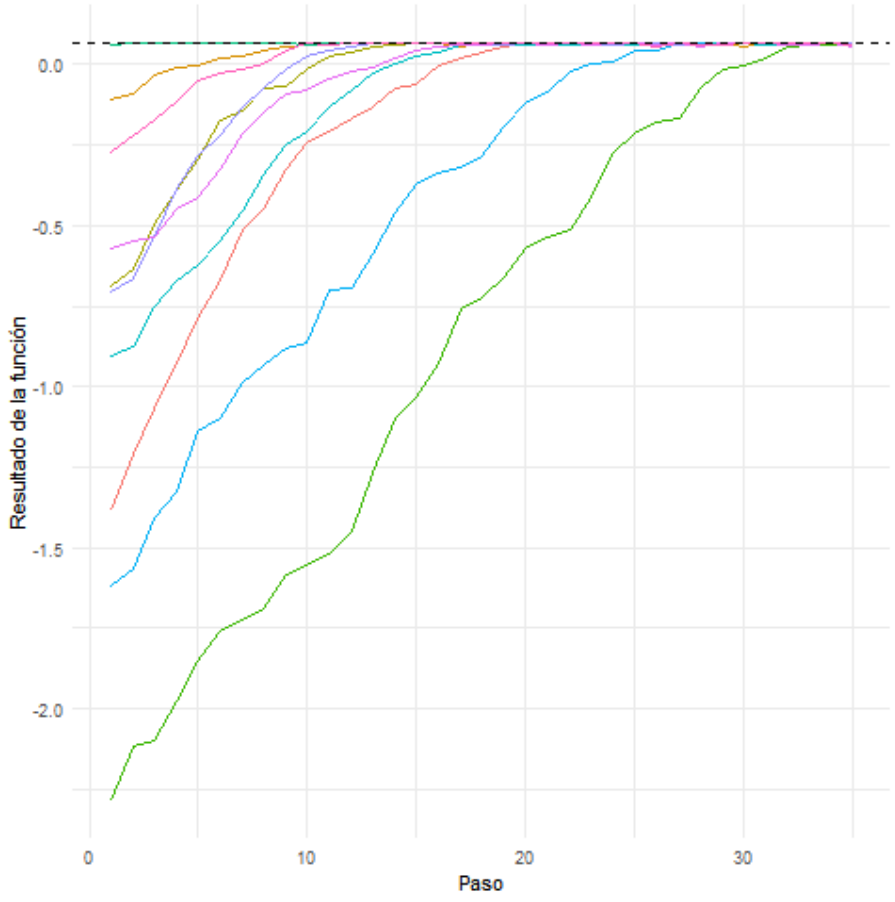


Figura 1 Valor de la ecuación 1 evaluada durante las diez réplicas del experimento

Práctica 6: Reto 1

Meta

Visualizar el recorrido de las posiciones durante la búsqueda de máximos para la ecuación 1 utilizando un mapa de calor para mostrar el valor de la función evaluada.

Desarrollo del código

Para el desarrollo de este código se utilizaron los paquetes `reshape2` y `ggplot2`. Basados en el código ejemplo que fue realizado con el paquete `lattice`, se creó el gráfico `base`, el cual muestra un mapa de calor para diferentes valores la `función f(x,y)` para un conjunto de valores `x` y `y`, para esta tarea se utilizó el comando `geom_raster` de `ggplot2`.

Con el objetivo de obtener y guardar los datos necesarios para crear el recorrido de la búsqueda local (coordenadas de `x` y `y` para diferentes réplicas) se utilizó la función `paralelo(w)` discutida en la sección de "Desarrollo de código" de la tarea, sólo que ahora se le denominó `replicas(r)`. Estos datos se guardaron dentro de la variable `trayecto`. La cual posteriormente fue filtrada en función a la columna `paso` con el objetivo de crear un nuevo `data.frame g` con las posiciones en `x` y `y` únicamente de la columna `paso` con valor a uno, dos, tres, hasta alcanzar el valor de `tmax`. Utilizando el `data frame g` se graficó una capa para el gráfico `base` utilizando el comando `geom_point` el cual ubicó quince puntos con las coordenadas obtenidas durante la paralelización.

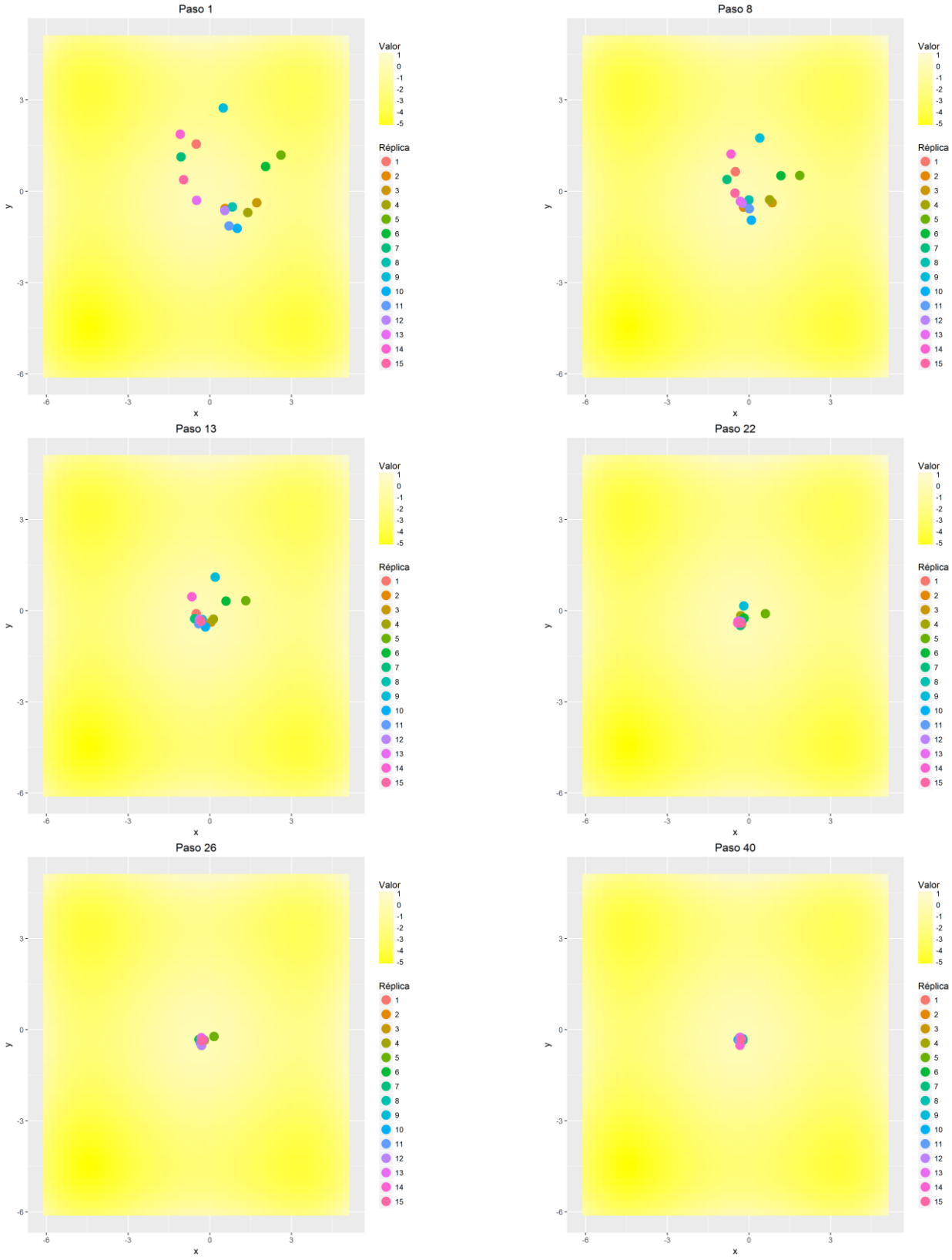
```
> library(reshape2)
> library(ggplot2)
>
> f <- function(x, y) {
+   return(((x + 0.5)^4 - 30 * x^2 - 20 * x + (y + 0.5)^4 - 30 * y^2 - 20
* y)/100)
+ }
> repeticiones<-15
>
> #Gráfico base
> x <- seq(-6, 5, 0.25)
> y <-  x
> z <- outer(x, y, f)
> colnames(z) <- y
> rownames(z) <- x
> d <- melt(z)
> names(d) <- c("x", "y", "Valor")
```

```
>
> base<-ggplot(d, aes(x, y)) +
+   geom_raster(aes(fill=Valor)) +
+   scale_fill_gradient(low="yellow", high="lightgoldenrodyellow")
>
>
> tmax<-40
> low <- -2
> high <- 3
> step <- 0.25
>
> replicas <-function(r){…}
>
>
> suppressMessages(library(doParallel))
> registerDoParallel(makeCluster(detectCores() - 1))
> trayecto<- foreach(r = 1:repeticiones, .combine=rbind) %dopar% replicas
(r)
> stopImplicitCluster()
>
> colnames(trayecto)=c("x","y","z","Replica","Paso")
> trayecto$Replica<- as.factor(trayecto$Replica)
>
> for (i in 1:tmax){
+
+ g<-trayecto[trayecto$Paso==i,]
+
+ base+ geom_point(data=g,aes(g$x,g$y,color=g$Replica),size=5)+
+   labs(color="Réplica")+
+   ggtitle(paste("Paso",i))+
+   theme(plot.title = element_text(hjust = 0.5))
+ ggsave(paste("P7_paso_",i,".png"))
+
+ }
```

Resultados

En la figura 2 se muestra una serie de imágenes para diferentes valores de `paso`, En ellos se puede observar cómo los puntos de diferentes colores inician en una posición al azar, y conforme el valor de `paso` aumenta todos los puntos se dirigen al centro de la imagen que corresponde al máximo local de la `función f.` Aquí se puede encontrar una animación paso a paso en formato .gif .

Figura 2 Proyección de la búsqueda local del valor máximo para la función f.



Práctica 6: Reto 2

Meta

Utilizar el recocido simulado, es decir, aceptar tanto resultados positivos, en los cuales, el valor de la posición evaluada es mayor a la posición actual o valores negativos, cuyos resultados son menores a la posición actual sólo si la probabilidad de aceptación es menor al cálculo de la exponencial de la diferencia entre ambas posiciones evaluadas divididas entre una temperatura T. (Ecuación 2)

$$p = e^{\frac{-(f(x')-f(x))}{T}} \qquad (2)$$

Desarrollo del código

Para el desarrollo de este código se reprodujo el experimento dónde se obtienen los mínimos para la `función f(x)` aceptando únicamente resultados menores y fue cambiado para buscar máximos en lugar de mínimos de la función f (Ecuación 3). Esto está señalado dentro del código en color verde y dentro de esta discusión se le denominará maximización original.

$$f(x) = 5\cos(14x - 3)\sin(2x^2 - 4x) + 2x^2 - 4x \qquad (3)$$

Las líneas de código las cuales realizan la función de recocido simulado se encuentran dentro de la `función replicas (r),` y fueron resaltadas en color azul. En las primeras líneas de la función se encuentran dos comandos `for`: el `for vt` y `for ve`, los cuales se utilizan para cambiar los diferentes valores de las variables `Temp` y `E`.

Dentro del recocido simulado se genera una posición actual x, totalmente al azar; para esta posición se calcula un delta aleatorio Δx, el cual puede tomar valores positivos como negativos, con el objetivo de generar la posición de un único vecino. Se evalúa la `función f` (ecuación 3) tanto para la posición actual y para el vecino único, aceptando la posición del vecino único como la nueva posición actual si la diferencia entre ambas funciones evaluadas es mayor a cero; si el resultado de esta diferencia es menor a cero entonces se calculará la probabilidad (ecuación 2) de aceptar valores menores al actual. Cada vez que se acepte un valor "peor" al actual, la temperatura Temp disminuirá en función a `E` con el objetivo de disminuir la probabilidad de aceptar este tipo de casos en los pasos posteriores. Para evitar que el valor de la posición actual se encuentre fuera de los límites establecidos (`high`), dentro del código se añadió un comando `if` dentro del `for pasos.`

En las gráficas generadas al final del código se comparan los valores de 4 réplicas (con una temperatura `Temp` y una variación `E` fijas) del procedimiento de recocido simulado contra el valor de la maximización original.

```
> library(ggplot2)
>
>
> f <- function(x) { # modificamos para que sea interesante
+   return(5 * cos(14*x - 3) * sin(2*x^2 - 4 * x) + 2 * x^2 - 4 * x)
+ }
>
> low <- -2
> high <- 4
> step <- 0.2
> varE<- seq(0.8,0.9,0.1)
> varT<-seq(1,3001,1000)
> tmax<-50
> ciclos<-4
>
> curr <- runif(1, low, high)
> rnormal<-data.frame()
> best<-curr
>
> for (tiempo in 1:tmax) {
+   delta <- runif(1, 0, step)
+
+   left <- curr - delta
+   right <- curr + delta
+   fl <- f(left)
+   fr <- f(right)
+
+   if (fl > fr) {
+     curr <- left
+   } else {
+     curr <- right
+   }
+
+   if (f(curr) > f(best)) {
+     best <- curr
+   }
+
+   dnormal<-cbind(curr,f(curr),best,f(best),tiempo)
+   rnormal<-rbind(rnormal,dnormal)
+
+ }
>
> colnames(rnormal)<-c("x","f(x)","best","f(best)","paso")
>
> replicas<-function(r){
+
+   resultados<-data.frame()
+   datos<-data.frame()
+   rpasos<-data.frame()
+
+ for (E in varE){
+
+ for (Temp in varT){
```

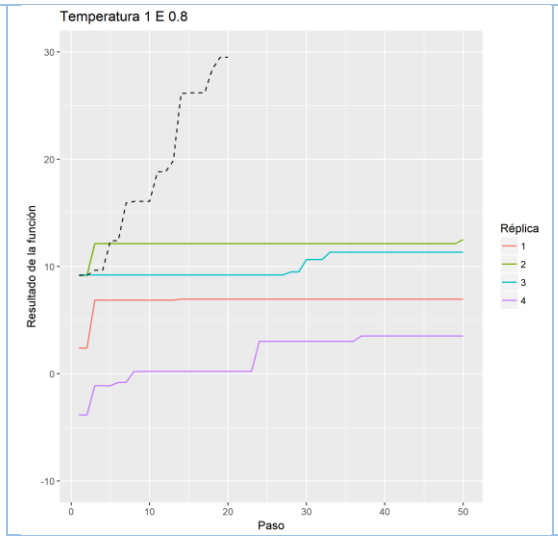
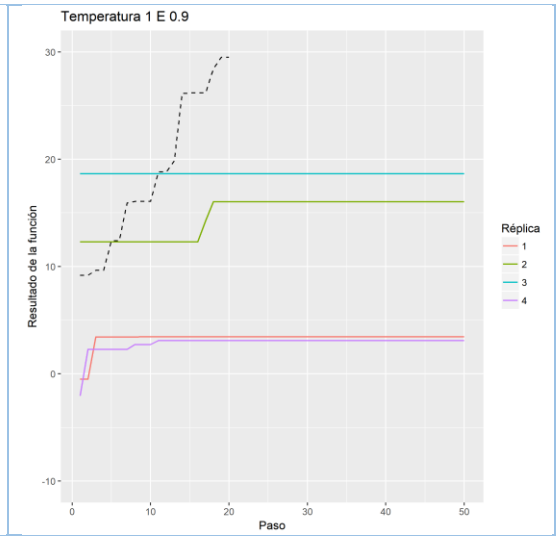
```
+   x<- runif(1, low, high)
+   T0<-Temp
+   cuenta<-c()
+   dpasos<-data.frame()
+   best<-x
+
+ for (pasos in 1:tmax){
+ if (x<high){xi<-x
+ dx=runif(1,-step,step)
+ xp<-x+dx
+ delta<-f(xp)-f(x)
+
+ if (delta>0){x<-xp}
+ else{if (runif(1)<exp(-delta/Temp)){
+   x<-xp
+   Temp=Temp*E
+   cuenta<-c(cuenta,1)
+ }}
+
+ if(f(x)>f(best)){best<-x}
+
+ dpasos<-cbind(r,E,T0,Temp,pasos,xi,xp,f(xi),f(xp),delta,sum(cuenta),bes
t,f(best))
+ rpasos<-rbind(rpasos,dpasos)
+ }
+   else
+     break;
+ }
+ }
+ }
+   return(rpasos)
+ }
>
> suppressMessages(library(doParallel))
> registerDoParallel(makeCluster(detectCores() - 1))
> datos<-foreach(r=1:ciclos,.combine=rbind) %dopar% replicas(r)
> stopImplicitCluster()
>
> colnames(datos)<-c("Replica","E","T0","Temperatura","Pasos","x","xp","f
(x)","f(xp)","Delta","Aceptacion","Mejor","f(mejor)")
>
>
> for (ve in 1:length(varE)){
+ g<-datos[datos$E==varE[ve],]
+
+
+ for (vt in 1:length(varT)){
+
+ g2<-g[g$T0==varT[vt],]
+ g2$Replica<- as.factor(g2$Replica)
+
+ ggplot() +
+   geom_line(data=g2, aes(x = g2$Pasos, y= g2$`f(mejor)`,color=g2$Replic
a),size=0.6)+
+   scale_y_continuous(name="Resultado de la función",limits = c(-10,30)
)+
+   scale_x_continuous(name="Paso")+
```

```
+   ggtitle(paste("Temperatura",varT[vt],"E",varE[ve]))+
+   geom_line(data=rnormal, aes(x = paso, y=rnormal$`f(best)`),linetype="
dashed",colour="black")+
+   labs(color="Réplica")
+
+ ggsave(paste("Variacion_E",varE[ve],"T",varT[vt],".png",sep="_"))
+
+ }
+ }
```

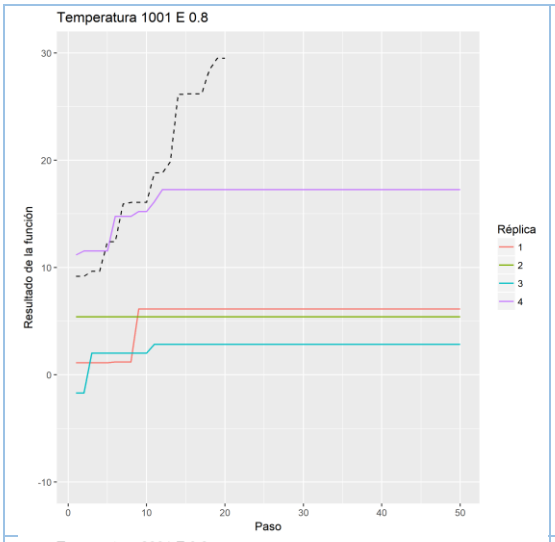
Resultados

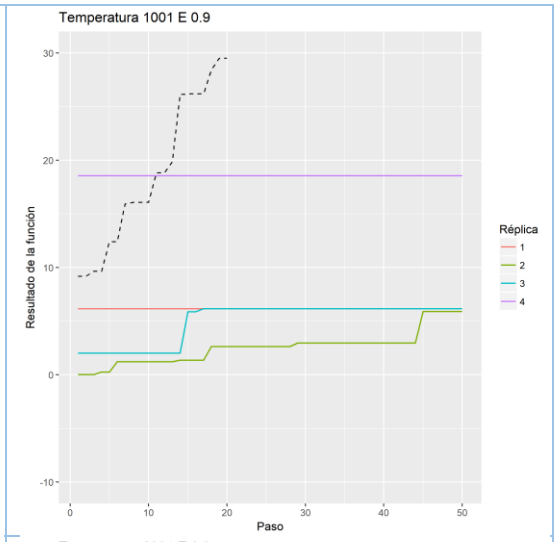
Si se observa la geometría de las líneas de la maximización original (línea negra punteada) contra la maximización con recocido simulado (líneas colores) se observa que ésta última tiende a caer en una meseta o un llano, es decir el valor considerado como el mejor se mantiene durante varios pasos. En la tabla 1 puede apreciarse que a mayor valor de temperatura `Temp` se encuentran valores más altos de la `función f` en pocos pasos.

Tabla 1 Comparación del efecto en el cambio del valor de la variación E (eje horizontal) contra la variación en el cambio de temperatura (eje vertical)

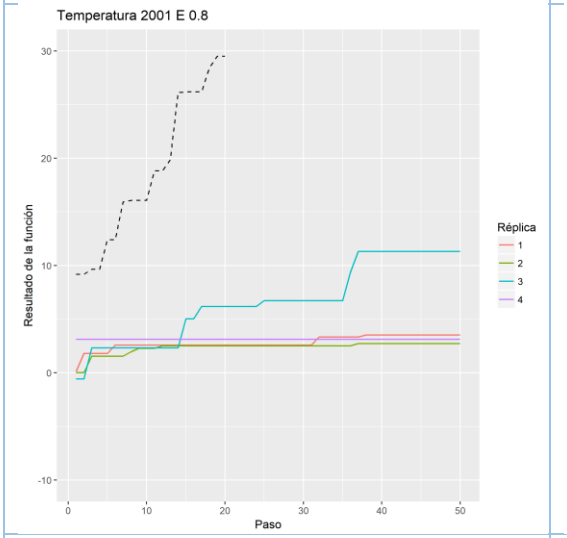
Temp	**E=0.8**	**E=0.9**
1		

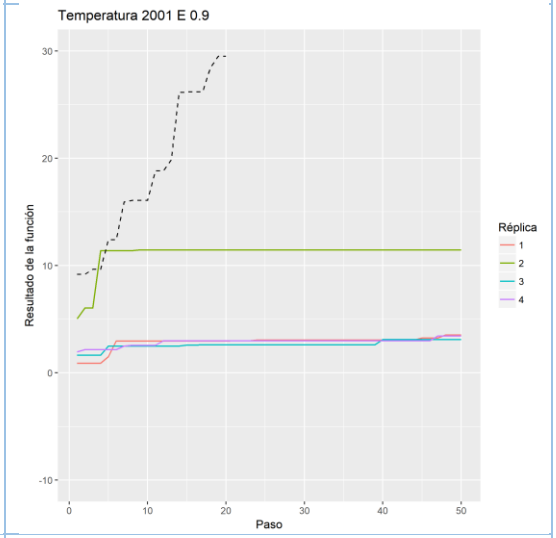
1001



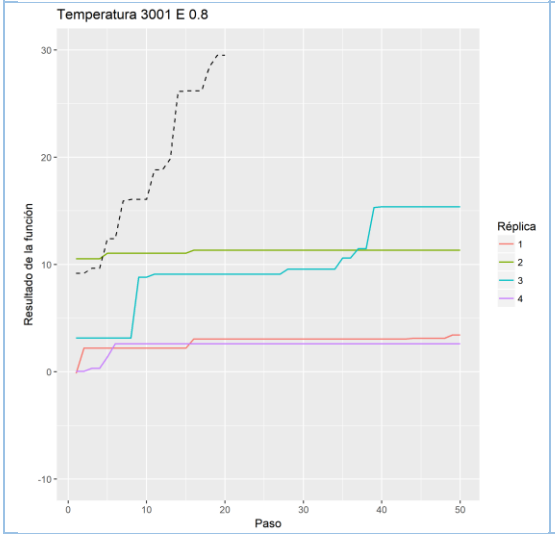


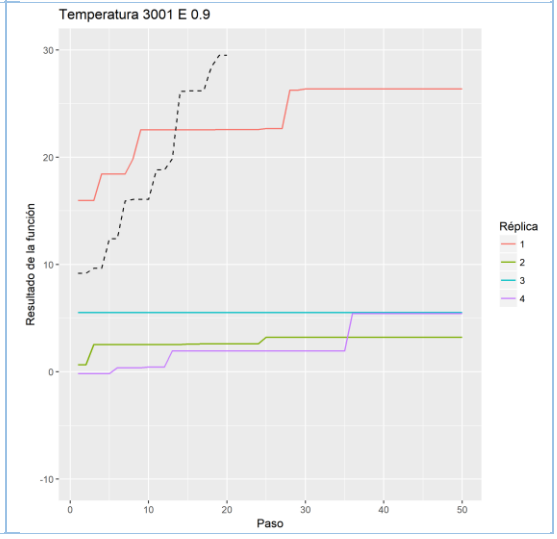
2001





3001





Referencias

[1] Elisa.dyndns-web.com. (2017). P7 — R paralelo — Schaeffer. [online] Available at: http://elisa.dyndns-web.com/teaching/comp/par/p7.html [Accessed 25 Sep. 2017].

[2] Wolframalpha.com. (2017). Wolfram|Alpha Widgets: "Max/Min Finder" - Free Mathematics Widget. [online] Available at: http://www.wolframalpha.com/widgets/view.jsp?id=372d3f309fef061977fb2f7ba36d74d2 [Accessed 26 Sep. 2017].