Práctica 11

Frente de Pareto

Introducción

En esta práctica se implementó un generador de polinomios aleatorios (funciones objetivo) donde se busca que las soluciones que puedan corresponder a una mejora en uno, no ocasionen una empeora en otro. Para resolver este tipo de problemas se utiliza la optimización multi-objetivos, dentro de este tipo de optimización se encuentra un método conocido como dominancia de Pareto el cual consiste en que una solución domina a otra si no empeora ninguno de los objetivos y mejora por lo menos a uno [1]. Para esta práctica se calculó la dominancia de Pareto y se identificaron las soluciones dominadas y no dominadas de un conjunto de funciones objetivo (polinomios).

Objetivos

1. Paralelizar el código original y estudiar los efectos en el tiempo de ejecución. Y graficar el porcentaje de Pareto como función del número de funciones objetivos y analizar el comportamiento obtenido.
2. Diversificar el frente de Pareto.

Simulación y Resultados

Para cumplir con el primer objetivo primero se identificaron las etapas del código que se repetían muchas veces y que son independientes entre sí, para poder paralelizar. Las etapas elegidas fueron la generación de las funciones objetivo representadas por polinomios, la evaluación cada una de las soluciones en cada una de las funciones objetivos y por último, el cálculo de las soluciones dominadoras y no dominadoras. Las modificaciones que se realizaron fueron, la declaración de tres funciones para cada una de las etapas y se utilizó la instrucción *foreach* del paquete *doparallel* para realizar la paralelización. El código se muestra a continuación:

```
1. Fobjetivo<- function(i){
2.   return(poli(md,vc, tc))
3. }
4. obj<-foreach(i=1:k) %dopar% Fobjetivo(i)
5.
6. Fsoluciones<- function(i){ # evaluamos las soluciones
7.   resul<-double()
8.   for (j in 1:k) { # para todos los objetivos
9.     datos<-eval(obj[[j]], sol[i,], tc)
10.          resul<-cbind(resul,datos)
11.       }
12.       return(resul)
13.     }
14.     sol <- matrix(runif(vc * n), nrow=n, ncol=vc)
15.     val <- matrix(rep(NA, k * n), nrow=n, ncol=k)
16.     val<-foreach(i=1:n, .combine = rbind) %dopar% Fsoluciones(i)
```

```
17.
18.      Fpareto<- function(i){
19.        no.dom <- logical()
20.        dominadores <- integer()
21.        d <- logical()
22.        for (j in 1:n) {
23.          d <- c(d, domin.by(sign * val[i,], sign * val[j,], k))
24.        }
25.        cuantos <- sum(d)
26.        return(cuantos==0)
27.      }
28.
29.      no.dom<-foreach(i=1:n, .combine = c) %dopar% Fpareto(i)
30.      frente<- subset(val, no.dom) # solamente las no dominadas
31.      tam<-dim(frente)[1]
```

Para comprobar si la paralelización fue efectiva se implementó un nuevo programa por el cual por medio de la instrucción *source* se corrieron ambos programas, el paralelizado y el original, en cada uno de ellos se utilizó un valor de *k* igual a 10 y se varió el número de soluciones *n* (200, 400, 600, 800, 1000), para cada una de las combinaciones se corrieron 10 réplicas. Los resultados obtenidos se puede observar en la figura 1. El código para realizar esta parte se muestra a continuación:

```
1. for (k in c(10)){
2.   for (n in seq(200,1000,200)){
3.     for(r in 1:10){
4.
5.     source('~/GitHub/SimulacionComputacional/P11/Original.R',
   encoding = 'UTF-8')
6.     Toriginal=cbind(k,n,"o",Tiempo)
7.
8.
9.     source('~/GitHub/SimulacionComputacional/P11/P11.R',
   encoding = 'UTF-8')
10.          Tparalelo=cbind(k,n,"p",Tiempo)
11.          Resultados=rbind(Resultados,Toriginal,Tparalelo)
12.          }
13.        }
14.      }
```

En la figura 1 se puede observar que para un número de soluciones *n* mayor a 200 el programa paralelizado siempre obtiene un tiempo de ejecución menor que el programa original. La diferencia entre las cajas-bigote se nota bastante para cada una de las corridas a distintos valores de *n* soluciones, por lo cual de manera visual se puede apreciar que la diferencia en los tiempos de ejecución entre ambos programas es significativa y no es necesario realizar una prueba estadística.

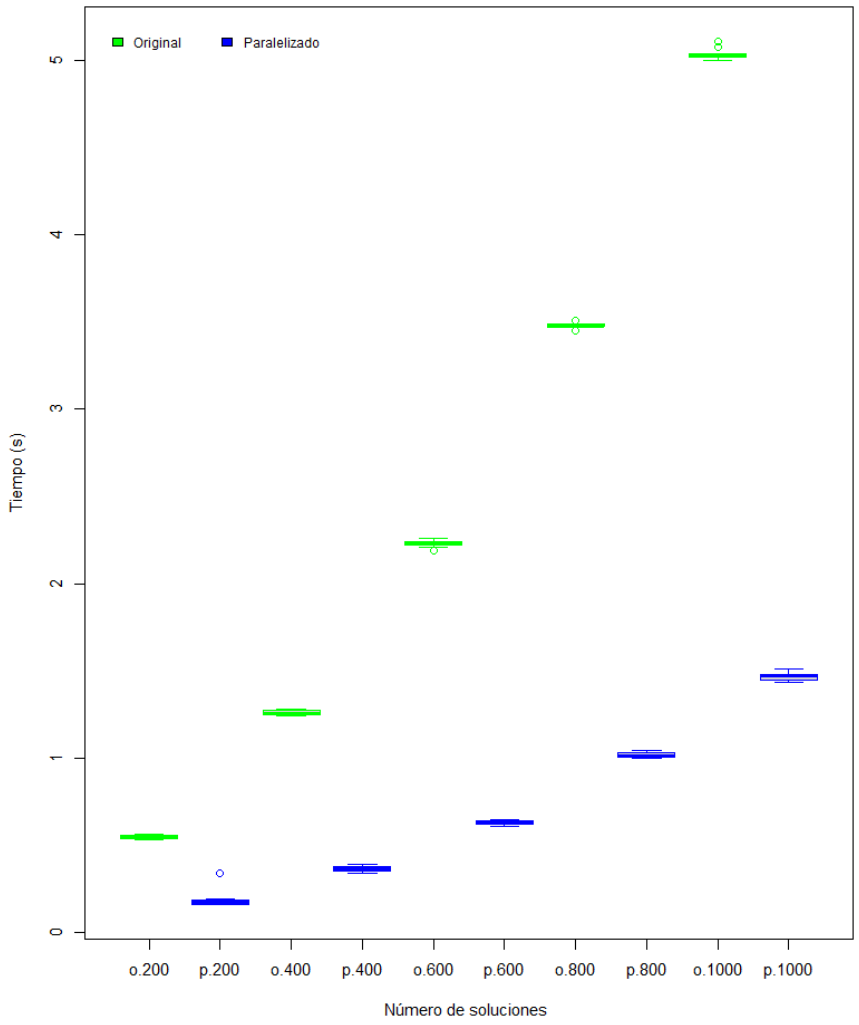


Figura 1. Gráfico de caja-bigote comparación de tiempos entre el programa original y paralelizado para diferentes valores de n y *k* igual a 10.

Para cumplir con la segunda parte del primer objetivo se graficó el comportamiento del porcentaje del frente de Pareto como función de número de funciones objetivo. Para realizar este grafico se utilizó el código a continuación:

```
1. n=200
2.   for (k in seq(2,10,2)){
3.     for (r in 1:50){
4.
5.       source('~/GitHub/SimulacionComputacional/P11/P11.R',
   encoding = 'UTF-8')
6.       Toriginal=cbind(tam,k,r)
7.       Resultados=rbind(Resultados,Toriginal)
8.   }
9. }
10.      stopImplicitCluster()
11.      names(Resultados)=c("Frente","nObjetivos","Replicas")
12.      Resultados$nObjetivos<- as.factor(Resultados$nObjetivos)
13.      library(ggplot2)
```

```
14.       ggplot(data=Resultados,
   aes(Resultados$nObjetivos, (Resultados$Frente/n)*100)) +
15.        geom_violin(scale="width",fill="dodgerblue4",
   color="black")+
16.         geom_boxplot(width=0.2, fill="dodgerblue2",
   color="aliceblue")+
17.       xlab("Número de funciones objetivo k") +
18.       ylab("Porcentaje de funciones no dominantes (%)")+
19.         theme_grey()
20.      ggsave(file=paste("p11_violin.png", sep='')) #Nombre del jpeg
```

Los resultados se muestran en la figura 2 donde se corrió el código para un valor de soluciones *n* igual a 200 y variando el número de funciones objetivo *k* (2, 4, 6, 8, 10) se realizaron 50 réplicas para cada uno de los valores de *k*.

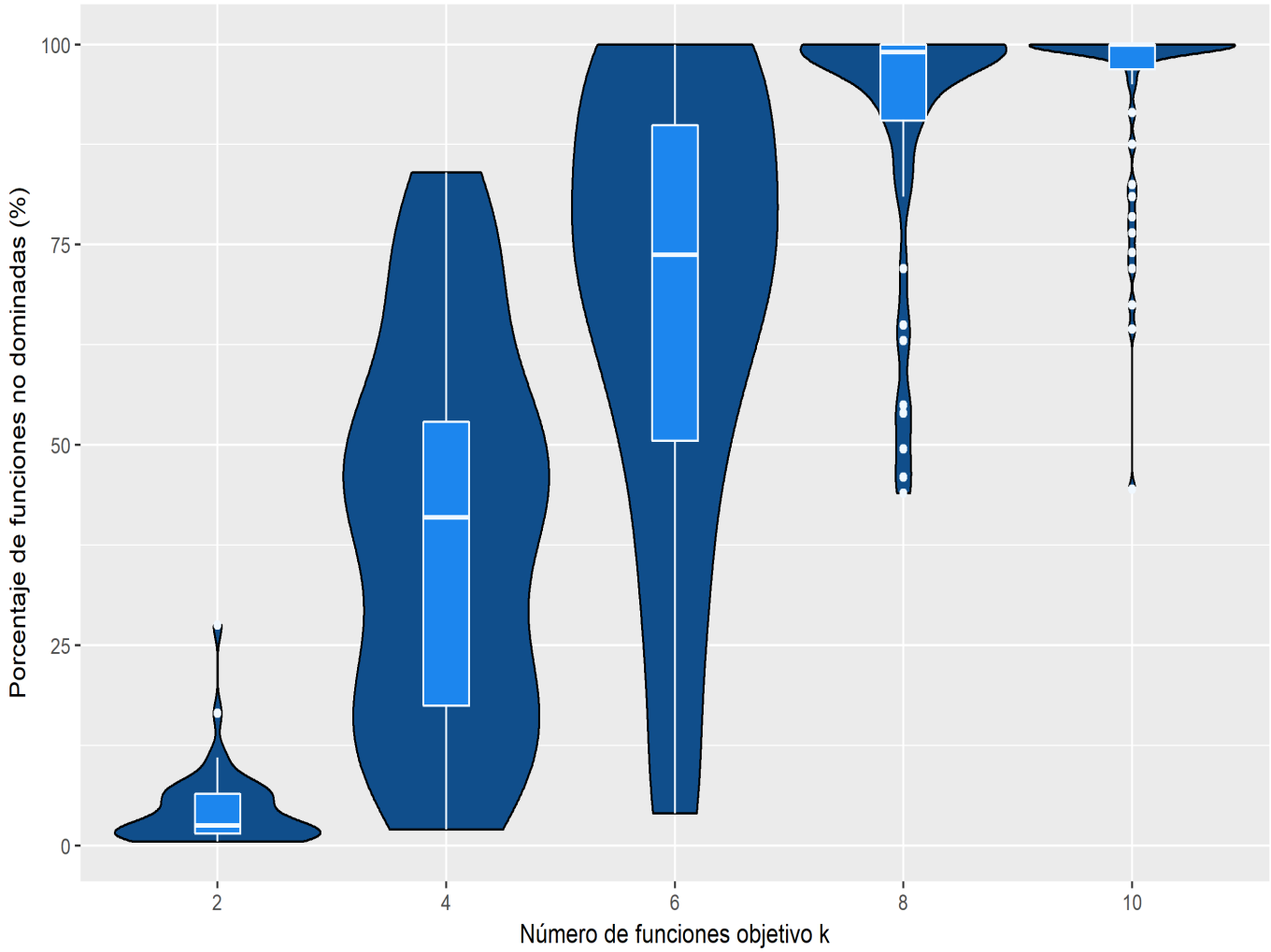


Figura 2. Gráfico de violín para diferentes valores de *k* con *n* igual a 200.

En la figura 2 se observa el porcentaje de soluciones no dominadas que corresponden a nuestro frente de Pareto, se aprecia que existe una tendencia donde a mayor número de funciones objetivos el porcentaje de soluciones no dominadas comienza a aumentar, donde para un número de funciones objetivos *k* igual a dos, el porcentaje de soluciones no dominadas se mantiene por debajo del 30% y en su mayoría por debajo de un 15%. Sin embargo cuando el número de funciones objetivos aumenta se puede observar como la distribución de estas soluciones no dominadas se acerca a porcentajes mayores donde para un valor de *k* igual a diez, la mayor parte de las soluciones pertenecen al frente de Pareto caso contrario a lo que ocurre a valores de *k* menores. La razón de este comportamiento puede deberse a que al tener una cantidad mayor de rubros (funciones objetivo) con los que pueden cumplir las soluciones hay una mayor probabilidad de que las soluciones sean mejores en al menos algún rubro y no empeoren a ningún otro.

Reto 1

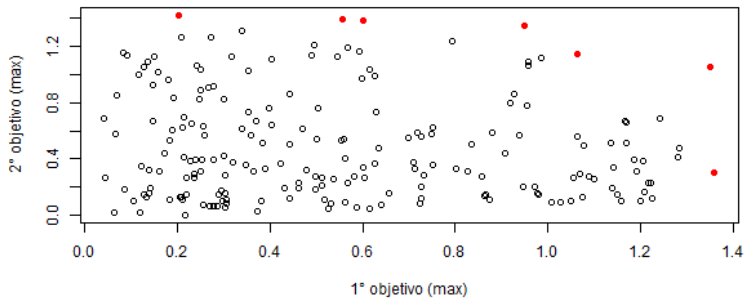
El primer reto tiene como objetivo diversificar el frente de Pareto obtenido hasta ahora, para este reto se consideró un sistema bidimensional, esto quiere decir, un número de funciones objetivo *k* igual a dos y un valor de soluciones *n* igual a 200 esto con el fin de ver el resultado de manera más sencilla. Diversificar sirve para evitar que las soluciones sean parecidas entre sí, ya que de esta manera es más fácil elegir una de las soluciones si cada una de ellas es diferente, en lugar de tener varias soluciones que sean parecidas entre ellas.

En este método de selección primero se condicionó si el largo del vector *frente* es mayor a dos ya que no tendría caso diversificar un frente de solo dos soluciones, luego se ordenaron las soluciones del frente de mayor a menor con respecto a una función objetivo (en este caso para la función objetivo dada en el eje de las abscisas), una vez ordenadas se calculó la separación entre cada una de las soluciones y se calculó una distancia de umbral, en este caso la distancia de umbral será igual a la media de las distancias de separación. Posteriormente se declaró un vector lógico *conservar* del largo de nuestro frente y es el encargado de elegir a las soluciones que conformarán el nuevo frente, estas serán las que se encuentren a una distancia mayor o igual que la distancia de umbral, manteniendo siempre la primera y última solución de nuestro frente ordenado. Finalmente se utilizó la instrucción *split.screen* para graficar la figura 3. El código para realizar dicha selección de muestra a continuación:

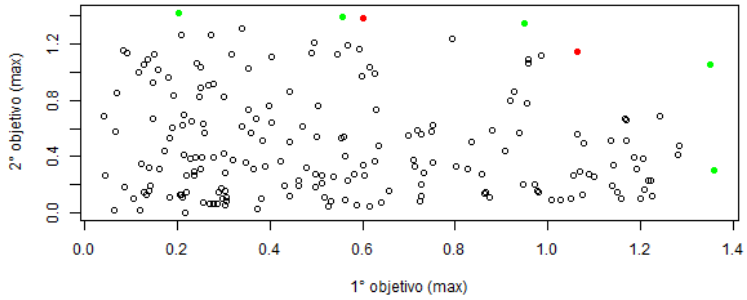
```
#RETO 1

png("Reto1.png",width = 600,height = 900,pointsize=12)
split.screen(c(3,1))
split.screen(c(1,2), screen = 1)
screen(1)
plot(val[,1], val[,2], xlab=xl,
     ylab=yl)
points(frente[,1], frente[,2], col="red", pch=19, cex=1)

     if (tam>2){ #Determinar si el frente tiene más de dos puntos
       frente<-as.data.frame(frente)
```

```
13.         colnames(frente)<-c("x","y")
14.         nFrente<-frente[order(frente$x),] #Ordenar de mayor a menor
    en función de una función objetivo x
15.
16.       # Calcular las distancias entre cada uno de los puntos del
    frente
17.         separacion<-c()
18.         for (i in 1:tam-1){
19.           s<- sqrt((nFrente[i,]$x-
    nFrente[i+1,]$x)^2+(nFrente[i,]$y-nFrente[i+1,]$y)^2)
20.           separacion<-c(separacion,s)
21.         }
22.         umbral<-mean(separacion) #distancia umbral
23.
24.         conservar<-rep(FALSE,tam)
25.         for (i in 1:tam){
26.           if (nFrente[i,]==head(nFrente,n=1)||nFrente[i,]==tail(nFr
    ente,n=1)){
27.             conservar[i]=TRUE}
28.           else{
29.               j<-max(which(conservar))
30.               s<-sqrt((nFrente[i,]$x-
    nFrente[j,]$x)**2+(nFrente[i,]$y-nFrente[j,]$y)**2)
31.               if(s>=umbral){
32.                 conservar[i]=TRUE
33.                }
34.               else{
35.                 conservar[i]=FALSE
36.                 }
37.             }
38.         }
39.
40.       Fdiversificado<-subset(nFrente,conservar)
41.       screen(2)
42.       plot(val[,1], val[,2], xlab=xl,
43.            ylab=yl)
44.       points(frente[,1], frente[,2], col="red", pch=19, cex=1)
45.       points(Fdiversificado[,1], Fdiversificado[,2], col="green",
    pch=19, cex=1)
46.       screen(3)
47.       plot(val[,1], val[,2], xlab=xl,
48.            ylab=yl)
49.       points(Fdiversificado[,1], Fdiversificado[,2], col="green",
    pch=19, cex=1)
50.       }
51.       dev.off()
```

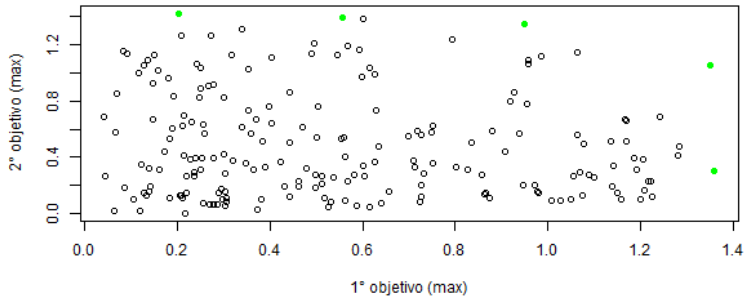
Los resultados de este método de selección se pueden observar en la figura 3. Donde el inciso a) corresponde al frente de Pareto original, b) el frente de Pareto donde se marcan cuales soluciones se van a remover y el inciso c) el frente de Pareto diversificado. Se observa como se eliminan algunas soluciones de nuestro frente y como resultado final obtenemos un frente diversificado. Hay que tomar en cuenta que la distancia de umbral es un parámetro y puede ser modificado, así como podrían existir otros métodos de selección.



a)



b)



c)

Figura 3. Gráfico de dispersión de puntos a) Frente de Pareto original, b) Paso intermedio, c) Frente de Pareto diversificado

Referencias

[1] P11 — R paralelo — Schaeffer, Elisa.dyndns-web.com, http://elisa.dyndns-web.com/teaching/comp/par/p11.html.