Reporte práctica 11
Frentes de Pareto

1. INTRODUCCIÓN

Para la práctica once se utilizó la optimización de soluciones multicriterio en el que a un conjunto de agentes o variables se le asignan valores en dónde se optimicen dos o más funciones objetivo que se pueden contradecir una a otra.

Se le llama frente de Pareto a aquel punto en el que ninguno de los agentes afectados puede mejorar su situación sin reducir el valor de cualquier otro agente.

2. SIMULACIÓN

Primero se implementa un generador de polinomios aleatorios. Estos polinomios serán las funciones objetivo. Tendremos una variable por término y un término por grado por variable.

Se generan muchas soluciones al azar y se calculan los valores de las funciones objetivo para cada solución. En cada función objetivo de determina si se va a minimizar (*TRUE*) o maximizar (*FALSE*), y se marca con un color distinto las mejores soluciones.

Por lo general las soluciones resultan no coincidir e incluso ser opuestos para esto se determina la dominancia de una solución con respecto a otras mediante la llamada dominancia de Pareto, en dónde una solución domina a otra si no empeora ninguno de los objetivos y mejora a por lo menos uno.

A las soluciones no dominadas (*dominantes*) se les conoce como un frente de Pareto. Graficamos la distribución de las cantidades de soluciones dominantes sobre el conjunto de soluciones la subrutina *geom_violin* del paquete *ggplot2*.

Para la tarea se buscó paralelizar el cálculo donde se creyó conveniente y se graficó el porcentaje de soluciones de Pareto como función del número de funciones objetivo mediante diagramas de violín combinados con diagramas de caja-bigote.

Se decidió hacer en paralelo cuando se crean las funciones objetivo, cuando se asignan los valores y cuando se determinan las soluciones dominantes (*obj*, *val* y *dominadores*) del código; mediante el uso del *foreach*, así como se muestra a continuación:

```
#######################paralelizar objetivo
#for (i in 1: k)
  p1<-function (i) {
  return (poli (md,vc, tc))
```

```
  }
obj<- foreach (i = 1: k, .combine=) %dopar% p1(i)

##############################paralelizar valores
#for (i in 1: n) {# evaluamos las soluciones
 p2<-function(i) {
   sol <- matrix (runif (vc * n), nrow=n, ncol=vc)
   val <- matrix (rep (NA, k * n), nrow=n, ncol=k)
        for (j in 1: k) {# para todos los objetivos
        return(eval(obj[[j]], sol[i,], tc))
        }
 }
val<- matrix (foreach (i = 1: n, .combine=rbind) %dopar% p2(i), nrow =
n, ncol = k, byrow = TRUE)
###################paralelizar dominantes
#for (i in 1: n) {
p3<-function(i) {
 d <- logical ()
  for (j in 1: n) {
    d <- c (d, domin.by (sign * val[i,], sign * val[j,], k))
  }
  cuantos <- sum(d)
  return(cuantos)
  #no.dom <- c (no.dom, cuantos == 0) # nadie le domina
}
dominadores<-rbind (dom,foreach(i = 1:n, .combine=rbind)%dopar% p3(i))
```

Para determinar la diferencia en ejecución de cada código (original y paralelo) se utilizó la herramienta *source* y con ayuda de un *for* se determinaron cinco corridas y utilizando otros dos *for* se varió la cantidad de funciones objetivo y de soluciones secuencialmente (5, 10, 15, 20, 25 y 30 para los objetivos; y 100, 150 y 200 para las soluciones).

Utilizando la herramienta *ggplot2* se graficó la ejecución de los dos códigos:

```
library(ggplot2)
png("p11R1_plot1.png")
ggplot(data=Tiempos, aes(x=Objetivos, y=Tiempo, color=Tipo)) +
guides(color=guide_legend(title = NULL))+
  geom_boxplot()+ facet_grid(Soluciones~.)
  ggtitle("Tiempos comparados variando soluciones y k")
graphics.off()
```

Para visualizar la distribución de soluciones como función del número de funciones objetivo se graficó mediante gráficas de violín en combinación de diagramas caja-bigote el porcentaje de soluciones de Pareto. A continuación, se muestra el código utilizado:

```
library(ggplot2)
png("p11T_violinplot.png")
ggplot (data=Datos, aes (x=Objetivos, y=(Dominantes/n) *100)) +
  geom_violin (scale="width", fill="darkolivegreen1", color="black") +
  geom_boxplot (width=0.2, fill="dodgerblue3", color="darkblue") +
  xlab ("Funciones objetivo") +
  ylab ("Porcentaje de soluciones dominantes") +
  ggtitle ("Cantidad de soluciones dominantes") +
  theme (plot.title = element_text(hjust = 0.5))
graphics.off()
```

3. RESULTADOS

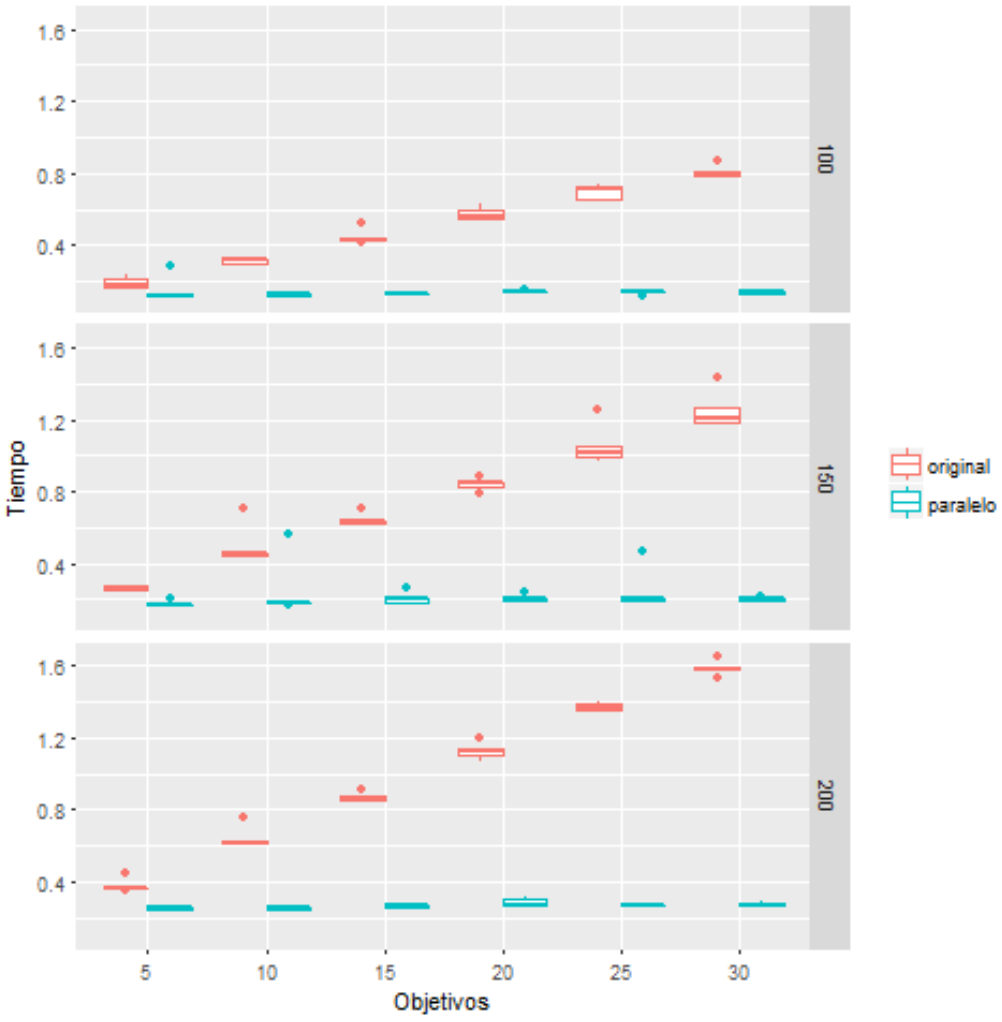

Figura 1. Gráfica caja-bigote de los tiempos de ejecución de los programas original (color salmón) y paralelizado (color celeste) variando funciones objetivo (eje horizontal) y soluciones (eje vertical izquierdo).

Se observó una diferencia entre el tiempo de ejecución del programa secuencial y el paralelo (figura 1) con tres diferentes cantidades de soluciones (100, 150 y 200) y seis de funciones objetivo (5, 10, 15, 20, 25 y 30). Resultando que el tiempo de ejecución es proporcional al número de soluciones y número de funciones objetivo, aumentando proporcionalmente conforme el aumento de los objetivos y las soluciones.

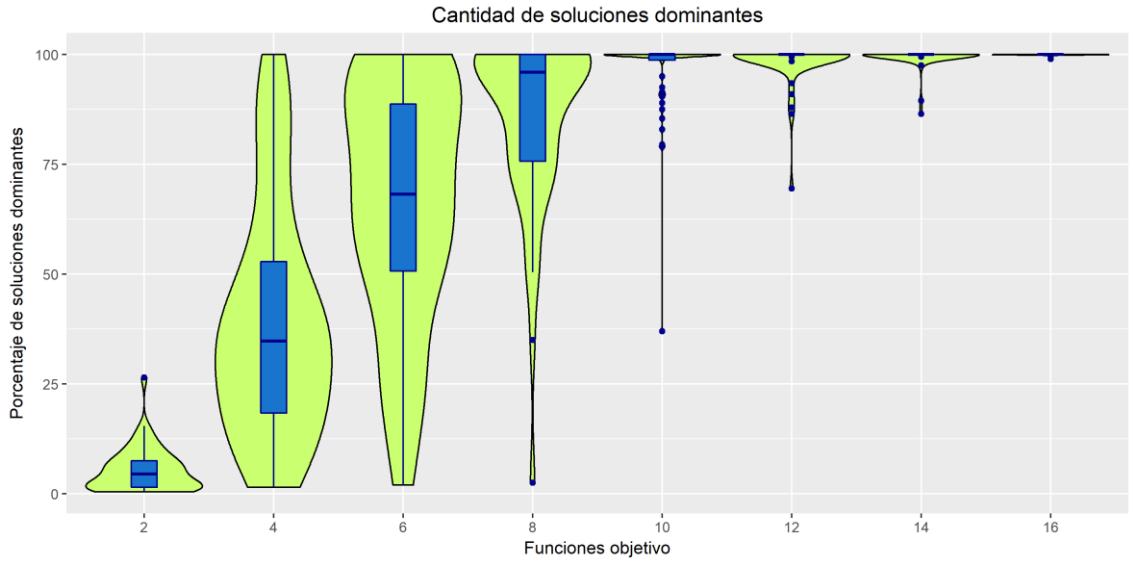

Figura 2. Gráficas de violín con gráficas caja-bigote sobrepuestas de la cantidad de soluciones dominantes (eje vertical) en función de diferentes cantidades de funciones objetivo (eje horizontal) dejando en 200 la cantidad de soluciones totales.

Después en las gráficas de violín sobrepuestas de diagramas caja-bigote (figura 2) se observa que entre mayor sea el número de funciones objetivo la **distribución** del porcentaje de las soluciones pertenecientes al frente es menor. Mientras que el porcentaje de soluciones que pertenecen al frente se apega cada vez más al 100%.

Para las cantidades de cuatro y seis funciones objetivo se encuentra la más amplia distribución del porcentaje de soluciones del frente. Y a partir de 8 funciones objetivo al rededor del 75% de soluciones pertenecen al frente. Ya para 16 funciones objetivo TODAS (100%) las soluciones están en el frente.

4. RETOS

- ## Reto 1

Para el reto uno se pide que se elija un subconjunto de las soluciones pertenecientes al frente de Pareto, esto para diversificar el frente y que no se concentren en zonas del frente.

Se utiliza el código original para graficar el frente, posteriormente se mide cuantas soluciones pertenecen al frente (*dimfrente*).

Se ordena el frente menor a mayor en base al valor de una función objetivo (*x*) para determinar los extremos del frente (*mejores*) y se guarda en la variable *ordenf*;

mediante un *for* que va de uno a *dimfrente-1* se guarda la distancia entre cada solución del frente. Se utiliza la distancia euclidiana entre cada solución.

Se fija una distancia *umbral* equivalente a la media de la distancia entre todas las soluciones del frente, en donde, las soluciones que se encuentren menor a esa distancia serán eliminadas. Haciendo el frente más diverso.

Se comparan las posiciones en dirección horizontal de cada solución del frente mediante un *for* que contiene una condición *if* que mantiene en el frente (*TRUE*) los extremos del mismo; pero si la solución comparada no es un extremo del frente y su distancia con el último valor guardado en el *nuevo* frente no es mayor al *umbral* entonces será eliminado del nuevo frente (*FALSE*).

Utilizando un vector lógico *nuevo* se eligen los que se quedarán (*TRUE*) para el frente diversificado.

Se presenta en seguida el código utilizado para la diversificación del frente de Pareto:

```
dimfrente<-dim(frente) [1]
val<-as.data.frame(val)
frente<-as.data.frame(frente)
ordenf<-frente [order (frente [,1]),]
    dis<-c ()
  for (i in 1: dimfrente-1) {
    d<- sqrt((ordenf[i,1]-ordenf[i+1,1]) **2+(ordenf[i,2]-
ordenf[i+1,2]) **2)
    dis<-c(dis,d)
  }
  umbral<-mean(dis)
  nuevo<-rep(FALSE,dimfrente)
  for (i in 1: dimfrente){
    if (ordenf[i,]==head(ordenf,n=1)||ordenf[i,]==tail(ordenf,n=1)){
      nuevo[i]=TRUE
      } else {
        j<-max(which(nuevo))
        d<-sqrt((ordenf[i,1]-ordenf[j,1]) **2+(ordenf[i,2]-
ordenf[j,2]) **2)
        if(d>=umbral) {
          nuevo[i]=TRUE
        } else{nuevo[i]=FALSE
          }
      }
  }
  mejor<-subset(ordenf,nuevo)
```

Se graficó mediante *ggplot2* los frentes (original, original sobreponiendo las soluciones diversificadas y el nuevo frente, ya diversificado) dejando la cantidad de funciones objetivo en dos.

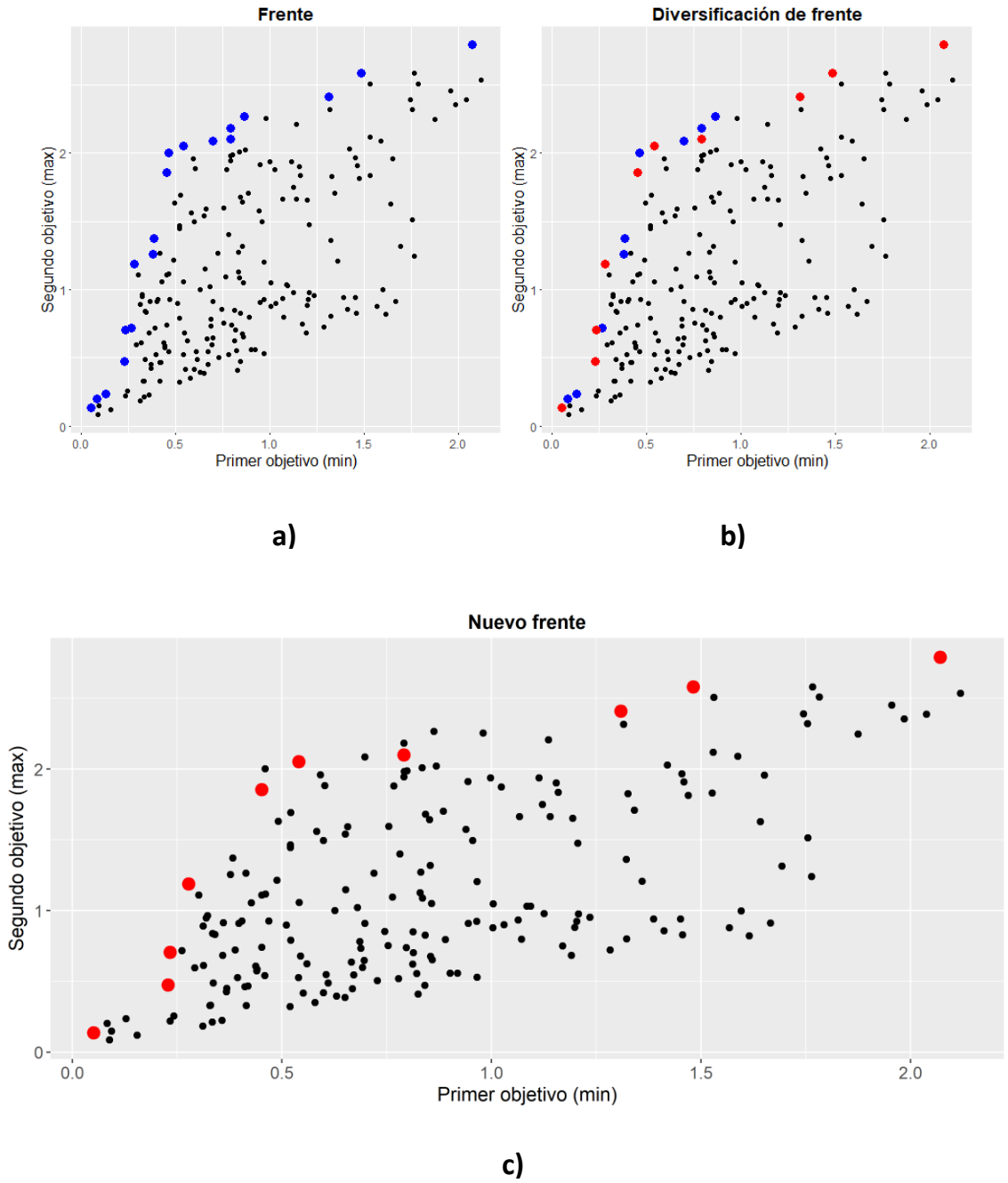

Figura 3. Gráfica de puntos de los frentes, original (a), original y diversificado (b), y el frente ya diversificado (c).

En las gráficas de los frentes (figura 3) se observa en puntos de color azul el frente original (a y b), en rojo el nuevo frente (b y c) y en negro las soluciones que no pertenecen al frente de Pareto.
Se observa que para el nuevo frente las soluciones del frente, están a una distancia equivalente, en la mayoría de los casos, además que no se observan alglomeraciones de soluciones en ciertas zonas del frente.

5. CONCLUSIONES

Se paralelizó el código dado mediante el uso de la herramienta *doparallel* y el comando *foreach*.

El código se ejecuta en menor tiempo al paralelizar las subrutinas *obj*, *val* y *dominadores*.

Existe una relación directamente proporcional entre la diferencia en tiempos de ejecución y la cantidad de soluciones.

Existe una relación directamente proporcional entre la diferencia en tiempos de ejecución y la cantidad de funciones objetivo.

Se logró graficar los porcentajes de soluciones de Pareto en función del número defunciones objetivo.

A mayor cantidad de criterios que se tomen en cuenta para determinar las mejores soluciones, mayor cantidad de soluciones se apegarán a ellos.

Se eligió un subconjunto del frente de Pareto en base a un *umbral* mínimo de distancia euclidiana entre soluciones.

Se eliminaron las soluciones del frente de Pareto que no cumplieran con la distancia *umbral* entre soluciones.

Mediante la librería *ggplot* se graficó un nuevo frente de Pareto, más diversificado, con una separación mínima igual al promedio de las distancias euclidianas entre las soluciones.