Práctica 11: Frentes de Pareto

En esta práctica se simulan escenarios en los cuales se toman decisiones en base a algunos objetivos o criterios deseados, los cuales pueden ser contradictorios entre sí, puesto que la mejora en uno de estos criterios puede repercutir como una empeora en otro. En esta clase de escenarios, existen soluciones que no son dominadas por otras soluciones, es decir éstas implican una mejora en algún objetivo o criterio sin agravar los otros objetivos; esta clase de soluciones forman un frente, conocido como frente de Pareto.

Meta

Paralelizar el código secuencial [1] utilizado como ejemplo donde sea crea conveniente; además graficar el porcentaje de soluciones que pertenecen al Frente de Pareto en función al número de objetivos `k` en un gráfico tipo violín.

Desarrollo del código

En esta sección sólo serán discutidas las modificaciones realizadas al código secuencial, el código completo puede encontrarse aquí, si se desea utilizar este código puede modificarse la instrucción `source` con los archivos de el código secuencial y paralelo. Cabe mencionar que la librería utilizada en esta paralelización fue el paquete `doParallel` y para graficar el paquete `ggplot2`.

Se optó por paralelizar las líneas de código que crean de manera aleatoria `k` polinomios o funciones objetivo, éstas son alojadas en la lista `obj`.

```
>   obj <- list()
>   obj<-foreach(i=1:k,combine=rbind) %dopar% poli(md, vc, tc)
```

La siguiente sección paralelizada es aquella en dónde se evalúan las funciones objetivo `obj`, para ello se creó la función `evaluacion(j)`. Los resultados de la evaluación son guardados en la matriz `val`.

```
> evaluacion<-function(j){
+   datos<-double()
+   for (i in 1:n) {# para todos los objetivos
+     res<- eval(obj[[j]], sol[i,], tc)
+     datos<-rbind(datos,res)
+   }
+   return(datos)
+ }
>   val<-foreach(j=1:k,.combine=cbind)%dopar%evaluacion(j)
```

La función `dominios(i)` fue creada con el objetivo de evaluar si una solución domina a otras soluciones y contabilizar el número total de soluciones que dominan a una solución dada.

```
> dominios<- function(i){
+   d <- logical()
+   for (j in 1:n) {
+     d <- c(d, domin.by(sign * val[i,], sign * val[j,], k))
+   }
+   return(sum(d))
+ }
>   dominadores<-foreach(i=1:n,.combine=c)%dopar% dominios(i)
```

Como ya se mencionó las soluciones que no son dominadas por otras soluciones, son las que forman el frente de Pareto, éstas fueron calculadas de forma secuencial ya que no involucra una gran cantidad operaciones, es decir sólo es una verificación a una condición.

```
>   no.dom <- logical()
>   for (i in 1:n){
+     no.dom<-c(no.dom,dominadores[i]==0)
+   }
```

Para verificar que la paralelización fue realizada de manera exitosa, se calculó el tiempo de ejecución del código secuencial y el paralelizado. Además, se facilitó la interpretación de datos al ser visualizados en un gráfico tipo caja-bigote y se realizó una prueba estadística para confirmar si la diferencia entre tiempos de ejecución es significativa. Las pruebas estadísticas se realizaron con la instrucción `t.test` y su código y los resultados de todas las combinaciones posibles se encuentran aquí.

```
> colnames(resultados)<-
c("Tipo","Replica","Objetivos","Soluciones","Tiempo")
> resultados$Tipo <- as.factor(resultados$Tipo)
> resultados$Soluciones <- as.factor(resultados$Soluciones)
> resultados$Objetivos <- as.factor(resultados$Objetivos)
> resultados$Tiempo<-
as.numeric(levels(resultados$Tiempo))[resultados$Tiempo]
>
> library(ggplot2)
> ggplot(data=resultados, aes(x = Soluciones, y=
Tiempo,color=Tipo)) +
+   geom_boxplot()+facet_grid(Objetivos~.,switch = "both")+
+   theme(legend.position = "bottom")+
+   theme_bw()
> ggsave("Paralelizacion_k5.png")
```

Con el fin de observar el porcentaje de soluciones que pertenecen al frente de Pareto en función al número de funciones objetivo `k`, fue necesario agregar la variable `tam` que indica el número de filas del cuadro de datos `frente`, es decir cuántas soluciones pertenecen a

dicho frente. Para obtener un conjunto de datos fue necesario hacer ciclos de este procedimiento variando el número de funciones objetivos `k`. Los resultados son representados en un gráfico tipo violín combinado con un gráfico caja-bigote con el objetivo de conocer la distribución del porcentaje de soluciones que pertenecen al frente de Pareto.

```
>tam<-dim(frente)[1]
+ ggplot(data=datos,aes(datos$Objetivos,(datos$Soluciones*100/n)))
+
+   geom_violin(scale="width",fill="dodgerblue4")+
+   geom_boxplot(width=0.25,fill="gainsboro",
color="black",outlier.size = 0.1) +
+   xlab("Objetivos") +
+   ylab("Frecuencia (%)")
+ ggsave("Frecuncia_objetivos_no.png")
```

Resultados

En la figura 1 se muestra los resultados de la medición de tiempos de ejecución del programa original o secuencial y el programa paralelizado en función al número de funciones objetivo `k` y al número de soluciones `n`. Es posible apreciar una leve mejoría en los tiempos de ejecución con funciones objetivo tan pequeñas como tres y un número de soluciones relativamente bajas como 200. Esta diferencia es más grande conforme las variables de objetivo y soluciones aumentan.

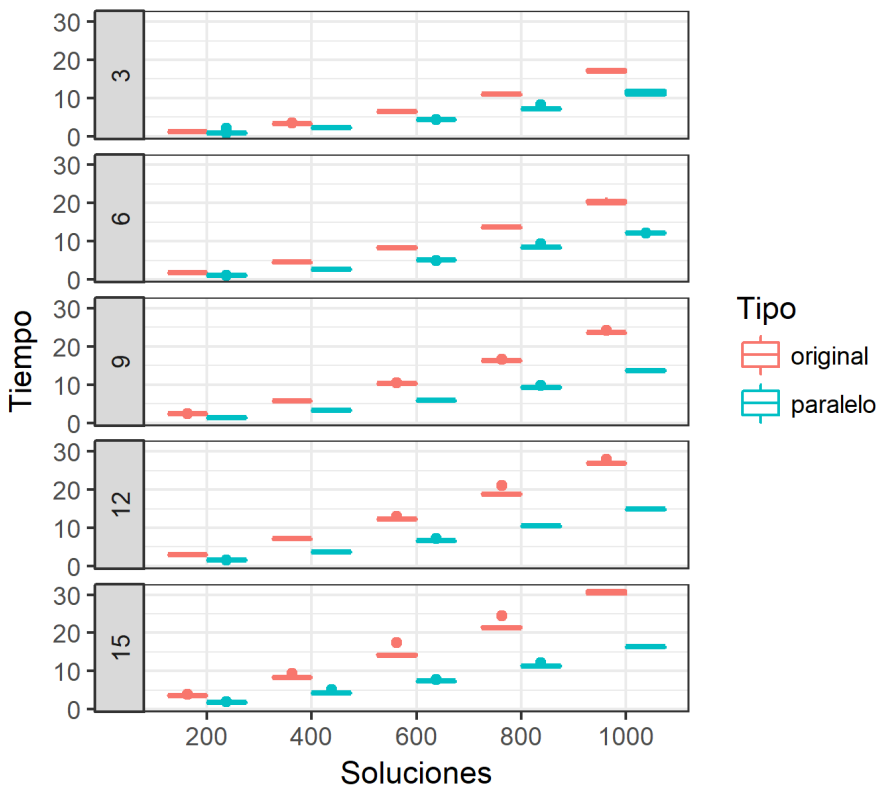

Figura 1 Tiempo de ejecución para diferente número de funciones objetivo (k) y número de soluciones (n)

Si bien existe una diferencia entre ambos tiempos de ejecución, no es posible observar si esta diferencia es verdaderamente significativa. Para lo cual se procedió a formular dos hipótesis nulas:

1. El número de soluciones `n` no interfiere en el aumento del tiempo de ejecución del programa secuencial y paralelizado.
2. La cantidad de funciones objetivo no tiene un impacto en el aumento del tiempo de ejecución entre ambos programas.

En el caso de la hipótesis nula 1 (Cuadro 1) se observa que sólo para un número relativamente pequeño de soluciones, tal como 200, el `p-valor` es mayor a 0.05 por lo tanto esta diferencia no es significativa, es decir no implica una ventaja el utilizar el código paralelizado en lugar del código secuencial. En cambio, para número de soluciones mayores el utilizar el código paralelizado ahorra tiempo en su ejecución.

Cuadro 1 Influencia del número de soluciones n en el tiempo de ejecución para tres funciones objetivo

Soluciones n	p-valor	Hipótesis nula 1	Diferencia
200	0.06154	Aceptada	No significativa
400	8.031×10^{-8}	Rechazada	Significativa
600	7.306×10^{-10}	Rechazada	Significativa
800	5.528×10^{-5}	Rechazada	Significativa
1000	1.108×10^{-6}	Rechazada	Significativa

Para la hipótesis nula 2 (Cuadro 2) la situación no es distinta a la hipótesis nula 1, puesto que, sólo para cantidades pequeñas la hipótesis nula es aceptada. Mientras que a mayores valores de funciones objetivo la diferencia entre ambos tiempos de ejecución es significativa, rechazando esta hipótesis.

Cuadro 2 Influencia del número de criterios k en el tiempo de ejecución para 200 soluciones

Objetivos *k*	p-valor	Hipótesis nula 2	Diferencia
3	0.06154	Aceptada	No significativa
6	3.926×10^{-9}	Rechazada	Significativa
9	4.095×10^{-8}	Rechazada	Significativa
12	1.453×10^{-13}	Rechazada	Significativa
15	2.168×10^{-8}	Rechazada	Significativa

Por lo tanto, se puede decir que para una cantidad de funciones mayores a tres y un número de soluciones mayores a 200 sería benéfico utilizar el código paralelizado.

En la figura 2 es posible observar que el porcentaje de soluciones que pertenecen al frente de Pareto en función a la cantidad de objetivos `k`, de ella se concluye que a mayor cantidad de objetivos existe un mayor porcentaje de soluciones que pertenecen al frente de Pareto, hasta llegar el momento en que todas las soluciones forman parte de este frente, volviendo infructuoso utilizar la optimización multicriterio. Esto se debe a que la probabilidad de que

las soluciones de frente de Pareto en el primer objetivo sean igual de buenos en todos los objetivos establecidos es muy disminuye conforme el número de objetivos aumenta, logrando así que, una solución dada sea la mejor en al menos un objetivo, lo cual engrandece el frente de Pareto.

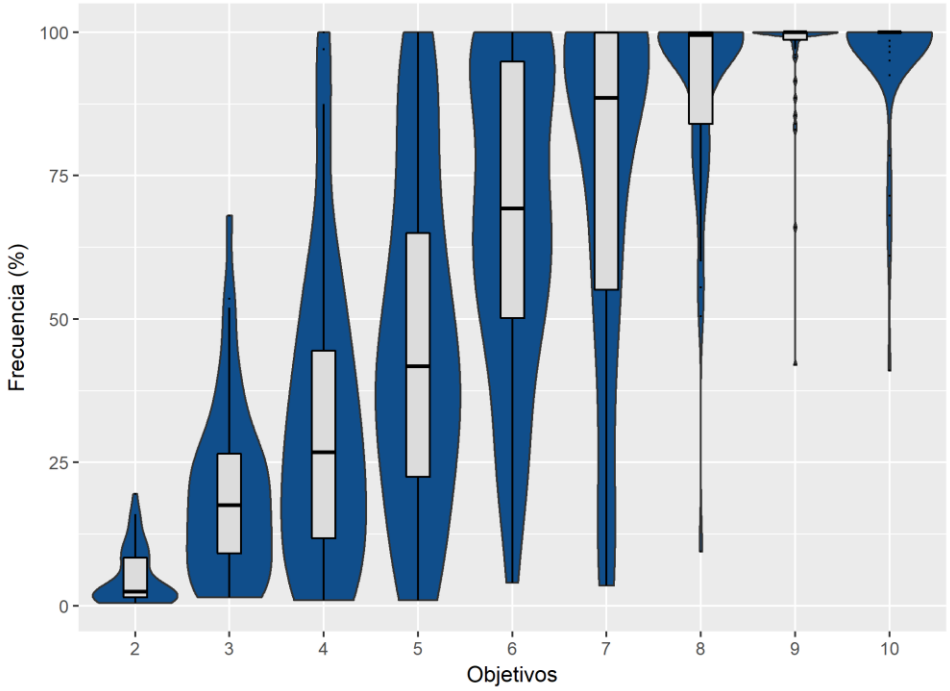

Figura 2 Gráficos de violín que muestra la distribución del porcentaje de soluciones que pertenecen al frente de Pareto

Reto 1

Meta

Diversificar un subconjunto de soluciones que pertenecen al frente de Pareto, es decir, evitar el agrupamiento estas soluciones. Graficar los resultados de la selección indicando con un color aquellas que se encuentran dentro del subconjunto diverso.

Desarrollo del código

Siguiendo el objetivo planteado, se desarrolló el siguiente código el cual sólo será útil si el número de soluciones que pertenecen al frente de Pareto es mayor a dos. Esta condición fue establecida con el objetivo de no utilizar recursos en un frente que no puede ser diversificado debido a su baja población. Para iniciar la diversificación se procedió a ordenar de manera ascendente el cuadro de datos `frente` en función a un objetivo dado, esto con el objetivo de calcular distancias entre soluciones vecinas. El cálculo de estas distancias es útil para establecer el radio mínimo de separación `rd` entre soluciones

pertenecientes al frente de Pareto, en esta ocasión el radio mínimo de separación es el promedio de la distancia entre soluciones vecinas. Después se declaró el vector lógico `mantener` el cual será necesario para elegir a las soluciones que son aceptadas en el frente diversificado. La iteración `i` recorre las posiciones del vector lógico `mantener` y evalúa si éstas deben conservarse en base 2 condiciones:

1. Es la primera o la última solución del frente
2. La distancia euclidiana entre solución en la posición `i` y la posición de última solución que se conservará `j` es mayor o igual al radio mínimo de separación.

En caso de que las respuestas sean afirmativas, esta solución será conservada en el frente diversificado.

```
>   if (nf>2){
+   n.frente<-frente[order(frente$x),]
+   distancia<-c()
+   for (i in 1:nf-1){
+     d<- sqrt((n.frente[i,]$x-
n.frente[i+1,]$x)**2+(n.frente[i,]$y-n.frente[i+1,]$y)**2)
+     distancia<-c(distancia,d)
+   }
+   rd<-mean(distancia)
+
+   mantener<-rep(FALSE,nf)
+
+   for (i in 1:nf){
+     if
(n.frente[i,]==head(n.frente,n=1)||n.frente[i,]==tail(n.frente,n=1
)){
+       mantener[i]=TRUE}else{
+         j<-max(which(mantener))
+         d<-sqrt((n.frente[i,]$x-
n.frente[j,]$x)**2+(n.frente[i,]$y-n.frente[j,]$y)**2)
+         if(d>=rd){mantener[i]=TRUE}else{mantener[i]=FALSE}
+       }
+   }
+   diverso<-subset(n.frente,mantener)
```

Con el fin de visualizar la diversificación del frente de Pareto se procedió a graficar las soluciones originales de dicho frente (color rojo) y las soluciones diversificadas del mismo (color verde).

```
+   ggplot()+
+     geom_point(data=val,aes(x,y))+
+     xlab(xl)+ylab(yl)+
+     geom_point(data=frente,aes(x,y),color="red",size=2)+
+     geom_point(data=diverso,aes(x,y),color="green",size=3)+
+     ggtitle("Ambos frentes")
+   ggsave("Ambos_frentes.png")
```

Resultados

En la figura 3 se observa la evolución del frente de Pareto original (Figura 3.a) hasta llegar al frente de Pareto diversificado (Figura 3.c). En la Figura 3.b se observa las soluciones originales (puntos rojos) que no cumplieron la condicional de distancia de separación por lo cual estas soluciones saldrán del frente diversificado (puntos verdes). Logrando así soluciones diversas en el frente, lo cual facilitará la elección de una solución.

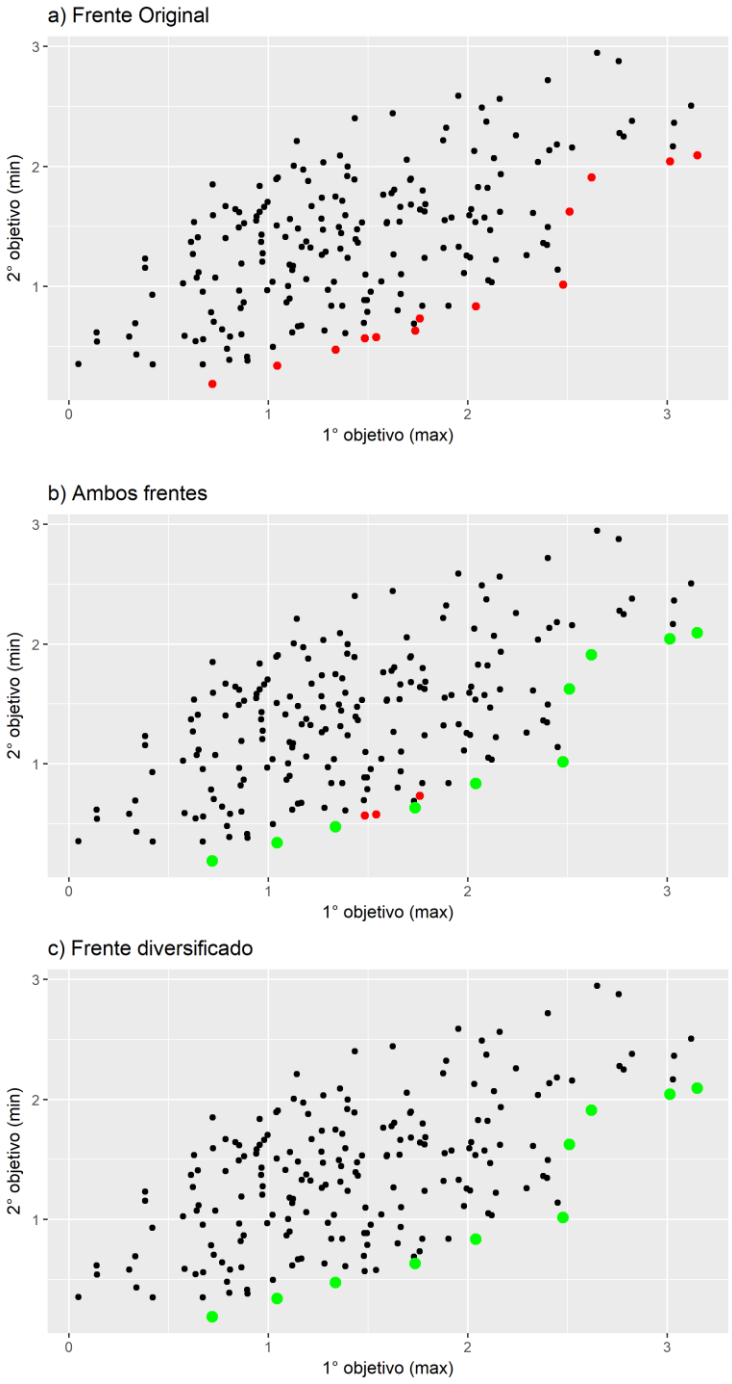

Figura 3Comparación entre el frente de pareto dado (puntos rojos) y el frente de pareto diversificado (puntos verdes)

Referencias

[1] Elisa.dyndns-web.com. (2017). P11 — R paralelo — Schaeffer. [online] Available at: http://elisa.dyndns-web.com/teaching/comp/par/p11.html [Accessed 24 Oct. 2017].