Práctica 9

Algoritmo genético

Objetivos

1. Paralelizar el código original y estudiar los efectos en el tiempo de ejecución.
2. Cambiar el método de selección de padres por el método ruleta y analizar si la calidad en la solución es mejor.

Simulación y Resultados

En esta práctica se implementó un algoritmo genético para una población de individuos de tamaño *init* donde cada individuo de esta población tiene *n* cantidad de genes que puede transmitir a la siguiente generación, por medio del problema de la mochila se establecen los valores y los pesos de cada uno de estos genes y cada individuo se crea con una combinación de los mismos al azar. Obteniéndose para esta práctica 200 individuos, cada uno de ellos con una combinación distinta de estos 50 genes. En cada generación los individuos pueden mutar con una probabilidad *pm* cambiando uno de sus genes de manera aleatoria, posteriormente, se eligen padres al azar, que darán cien hijos en cada generación, donde al final se elegirán los 200 mejores individuos (que la suma de su función objetivo se adecue al valor óptimo) para dar lugar a la generación siguiente y así sucesivamente.

Para lograr nuestro primer objetivo se analizó el código original y se determinaron las partes del código que eran posibles paralelizar. Estas partes del código fueron las etapas de mutar, reproducción y factibilidad, estas etapas son completamente independientes y se repiten muchas veces a lo largo de la simulación es por ello que se decidió paralelizar. Para cada una de estas etapas se creó una función como se muestra a continuación:

```
#Paralelizar mutaciones
mutar<-function(i){
  if (runif(1) < pm) {
    return( mutacion (unlist(p[i,]), n))
  }
}
#Paralelizar cruces
cruces<-function(i){
  padres <- sample(1:tam, 2,replace=FALSE)
  hijos <- reproduccion(p[padres[1],], p[padres[2],], n)
  h1 <-hijos[1:n] # primer hijo
  h2 <-hijos[(n+1):(2*n)] # segundo hijo
  hijo<- rbind(h1,h2)
  return(hijo)
}
#Paralelizar factbilidad
factibilidad <- function(i){
  obj <- c(objetivo(p[i,], valores))
  fact <- c(factible(p[i,], pesos, capacidad))
  resul<-(cbind(obj,fact))
  return(resul)
}
```

Luego de implementar estas funciones se utilizó el paquete *doparallel* y la instrucción *foreach* para cada una de las funciones.

```
for (iter in 1:tmax) {
  p$obj <- NULL
  p$fact <- NULL

  p<-rbind(p, foreach(i=1:tam, .combine = rbind) %dopar% mutar(i)) #MUTAR
  p<-rbind(p,foreach(i=1:rep, .combine=rbind) %dopar% cruces(i)) #CRUCES
  tam <- dim(p)[1]
  obj <- double()
  fact <- integer()
  rownames(p)<-c(1:dim(p)[1])
  p<-data.frame(sapply(p,function(x)as.numeric(as.character(x))))
  p<-cbind(p,foreach(i=1:tam,
.combine=rbind) %dopar% factibilidad(i)) #FACTBILIDAD
  mantener <- order(-p[, (n + 2)], -p[, (n + 1)])[1:init]
  p <- p[mantener,]
  tam <- dim(p)[1]
  assert(tam == init)
  factibles <- p[p$fact == TRUE,]
  mejor <- max(factibles$obj)
  mejores <- c(mejores, mejor)
}
stopImplicitCluster()
```

Para determinar si la simulación fue efectiva, se procedió a calcular el tiempo de ejecución por medio de un nuevo programa donde se utilizó la instrucción *source* para llamar el código original y el paralelizado. Para esta prueba se utilizó una población inicial *init* de (50, 150, 200, 250) individuos con un número de generaciones *tmax* de 50, una probabilidad de mutación *pm* 0.05. Los resultados se muestran en la figura 1.

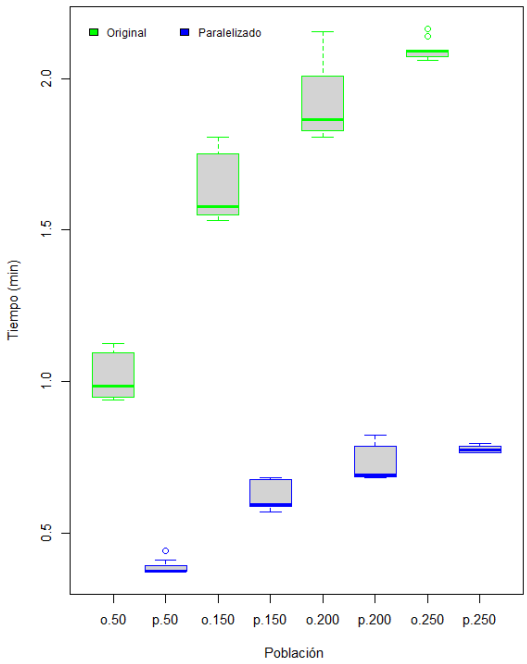

Figura 1. Gráfico de caja-bigotes de comparación de tiempos entre el programa paralelizado y el original.

En la figura 1 se observa que la brecha entre los tiempos entre el programa original y el paralelizado es amplia por lo cual no es necesario complementar el estudio con un análisis estadístico. Además, se observa que entre mayor sea la población inicial el programa paralelizado brilla más.

Se decidió hacer un estudio con números de población inicial *init* más pequeños y determinar a partir de que valores el programa paralelizado era factible, los resultados se muestran en la figura 2.

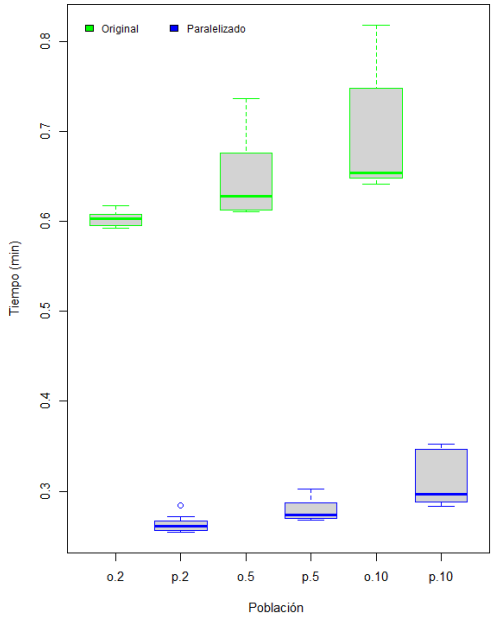

Figura 2. Gráfico de caja-bigote comparación de tiempos para *init* menores.

En la figura 2 se observa que incluso con poblaciones iniciales *init* pequeñas (2, 5, 10) el programa paralelizado tiene un tiempo de ejecución mucho más pequeño que el programa secuencial.

Reto 1

El primer reto consiste en modificar el método de selección de padres a la hora de reproducirse. En lugar de elegir a los individuos que serán padres de manera al azar, esta vez se elegirán con una probabilidad que será directamente proporcional a su valor en la función objetivo. Para implementar este cambio primero se generó una nueva función llamada *crucesruleta* donde se estableció con ayuda del parámetro *prob* de la instrucción *sample* la nueva probabilidad con un vector llamado *bestpadres*, este vector lo que guarda es la probabilidad de ser elegidos obtenida calculando los valores objetivo de cada uno de los individuos iniciales y posteriormente dividiendo cada uno entre el valor total de valores objetivo esto con el fin de que la probabilidad de ser elegidos sea directamente proporcional a su valor en la función objetivo. El código para implementar estos cambios se muestran a continuación:

```
#Paralelizar cruces ruleta
crucesruleta<-function(i){
  padres <- sample(1:tam, 2, prob=bestpadres, replace=FALSE)
  hijos <- reproduccion(p[padres[1],], p[padres[2],], n)
  h1 <-hijos[1:n] # primer hijo
  h2 <-hijos[(n+1):(2*n)] # segundo hijo
  hijo<- rbind(h1,h2)
  return(hijo)
}
```

```
bestpadres<-foreach(i=1:tam, .combine=c) %dopar% objetivo(p[i,],valores)#Calcular lo valores de cada individuo
bestpadres<-bestpadres/sum(bestpadres)
```

Estos cambios se realizaron modificando el código de la tarea base, donde se incluyó un *for* para realizar el número de réplicas para la simulación y los *for* de generaciones, uno donde la forma de selección es completamente al azar y el otro donde implementamos esta nueva forma de selección como se muestra a continuación:

```
for (replica in 1:2){

  #Tarea base paralelizado con ruleta
  p <- poblacion.inicial(n, init)
  tam <- dim(p)[1]
  assert(tam == init)
  pm <- 0.05
  rep <- 50
  tmax <- 50
  Rmejores <- double()

  for (iter in 1:tmax) {

    p$obj <- NULL
    p$fact <- NULL
    p<-rbind(p, foreach(i=1:tam, .combine = rbind) %dopar% mutar(i)) #MUTAR
    bestpadres<-foreach(i=1:tam, .combine=c) %dopar% objetivo(p[i,],valores)#Calcular lo valores de cada individuo
    bestpadres<-bestpadres/sum(bestpadres)
    p<-rbind(p,foreach(i=1:rep, .combine=rbind) %dopar% crucesruleta(i)) #CRUCES
    tam <- dim(p)[1]
    obj <- double()
    fact <- integer()
    rownames(p)<-c(1:dim(p)[1])
    p<-data.frame(sapply(p,function(x)as.numeric(as.character(x))))#Convertir de caracter a numerico
    p<-cbind(p,foreach(i=1:tam, .combine=rbind) %dopar% factibilidad(i)) #FACTBILIDAD
    mantener <- order(-p[, (n + 2)], -p[, (n + 1)])[1:init]
    p <- p[mantener,]
    tam <- dim(p)[1]
    assert(tam == init)
    factibles <- p[p$fact == TRUE,]
    Rmejor <- max(factibles$obj)
    Rmejores <- c(Rmejores, Rmejor)
  }

#Tarea base paralelizado sin ruleta
  p <- poblacion.inicial(n, init)
  tam <- dim(p)[1]
```

```
38.   assert(tam == init)
39.   pm <- 0.05
40.   rep <- 50
41.   tmax <- 50
42.   mejores <- double()
43.
44.   for (iter in 1:tmax) {
45.     p$obj <- NULL
46.     p$fact <- NULL
47.
48.     p<-rbind(p, foreach(i=1:tam, .combine = rbind) %dopar% mutar(i)) #MUTAR
49.
50.     p<-rbind(p,foreach(i=1:rep, .combine=rbind) %dopar% cruces(i)) #CRUCES
51.     tam <- dim(p)[1]
52.     obj <- double()
53.     fact <- integer()
54.     rownames(p)<-c(1:dim(p)[1])
55.     p<-data.frame(sapply(p,function(x)as.numeric(as.character(x))))
56.     p<-cbind(p,foreach(i=1:tam,
    .combine=rbind) %dopar% factibilidad(i)) #FACTBILIDAD
57.     mantener <- order(-p[, (n + 2)], -p[, (n + 1)])[1:init]
58.     p <- p[mantener,]
59.     tam <- dim(p)[1]
60.     assert(tam == init)
61.     factibles <- p[p$fact == TRUE,]
62.     mejor <- max(factibles$obj)
63.     mejores <- c(mejores, mejor)
64.   }
65.
66. stopImplicitCluster()
```

Al final se graficó por medio del paquete *lattice* ambos métodos de selección, los resultados se muestran en la figura 3. Se puede observar que cuando se implementa un sistema de selección por ruleta en las primeras generaciones se tiene un aumento marcado a comparación a cuando se realiza sin ruleta. Sin embargo, al final este cambio es poco significativo como se muestra de manera visual. Esto puede ser debido a que al inicio te aseguras de tener a los mejores padres, pero mientras avanzan las generaciones y se tienen hijos de los hijos tal vez esta mejora se va degradando dando un resultado muy similar para el método de selección sin ruleta.

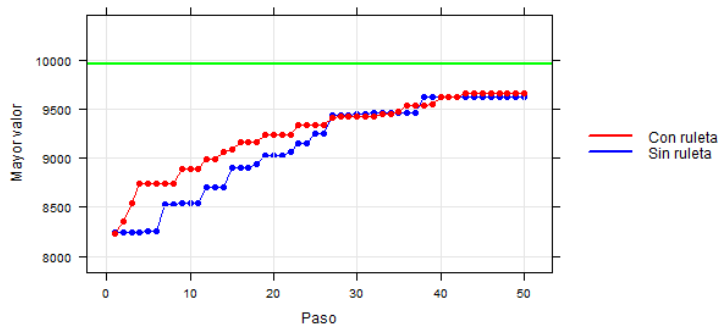

Figura 2. Comparación de ambos métodos de selección.

Conclusiones

Se puedo comprobar que el programa paralelizado se realizaba en un tiempo menor que el programa secuencial, sin importar la cantidad de población inicial, aunque tal vez sería recomendable variar algún otro parámetro como la cantidad de generaciones o la cantidad de genes *n* y corroborar si el programa paralelizado sigue siendo eficiente. Para los distintos tipos de selección de padres se observó durante la simulación que el método con ruleta no tenía un efecto significativo en la calidad de la solución a comparación del método sin ruleta.