Práctica 10: algoritmo genético

Ricardo Rosas Macías

21 de mayo de 2019

1. Introducción

En la práctica se aborda el problema de la mochila; mejor conocido como Knaspack, en el cual se busca la optimización de la combinación posible para la mezcla de artículos máximo que puede estar dentro de esta, en función al peso de cada uno.

2. Objetivo

Se realizó cambios en el código proporcionado en la página web [2], de tal forma que la programación dinámica con ayuda de los algoritmos permite crear de una manera más sencilla la mejor combinación posible.

2.1. Descripción

La finalidad del experimento es [2]:

> "Paralelizar el algoritmo genético y estudia los efectos en su tiempo de ejecución con pruebas estadísticas y visualizaciones, variando el número de objetos en la instancia. Genere instancias con tres distintas reglas: el peso y el valor de cada objeto se generan independientemente con una distribución normal, el peso de cada objeto se generan independientemente con una distribución normal y su valor es correlacionado con el peso, con un ruido normalmente distribuido de baja magnitud, el peso de cada objeto se generan independientemente con una distribución normal y su valor es inversamente correlacionado con el peso, con un ruido normalmente distribuido de baja magnitud.
> Asimismo, determinar para cada uno de los tres casos a partir de qué tamaño de instancia el algoritmo genético es mejor que el algoritmo exacto en términos de valor total obtenido por segundo de ejecución."

3. Resultados y conclusiones

Se realizó el código en base a los antecedentes anteriormente reportados [3]. En las primeras líneas del código se definió los parámetros de experimentación con las cuales se trabajó; se estableció una serie de instancias como se describen en el objetivo. El código permite exhibir la eficiencia de selección de muestra en el experimento; siendo repetido 5 veces para obtener un valor estadístico más exacto. Además se paralelizó y configuró para que proporcione el tiempo que toma en ejecutar el código, de manera que logre mostrar las diferencias en tiempos de ejecución normal y paralelizada.

```
startP=Sys.time()
n <- 50
generaciones <- c(200, 250, 300)
pmutaciones <- c(0.05, 0.1, 0.2)
reproducciones <- c(50, 75, 100)
iteraciones <- c(10, 15, 20)
replicas <- 5

datos <- data.frame(Replica = integer(), Optimo = integer(),
                    Generaciones = numeric(), ProbMut = numeric(),
                    Reproducciones = numeric(), Iteraciones = numeric(),
                    Objetivo = double())
cluster <- makeCluster(detectCores())
opt <- knapsack(capacidad, pesos, valores)[2]
clusterExport(cluster, "n")
clusterExport(cluster, "pesos")
clusterExport(cluster, "valores")
clusterExport(cluster, "capacidad")

for (tmax in iteraciones) {
  for (rep in reproducciones) {
    for (pm in pmutaciones) {
      for (init in generaciones) {
        for (r in 1:replicas) {

          #Inicio del genetico parsapply
          pobl <- t(parSapply(cluster, 1:init, function(i) {
            return(round(runif(n)))
          }))

          p <- as.data.frame(pobl)
          tam <- dim(p)[1]

            clusterExport(cluster, "p")
            clusterExport(cluster, "tam")

            probabilidades <- runif(tam) < pm
            mutados <- which(probabilidades %in% TRUE)

            mutaciones <- t(parSapply(cluster, mutados, function(i) {
              pos <- sample(1:n, 1)
              mut <- p[i,]
              mut[pos] <- (!p[i,][pos]) * 1
              return(as.numeric(mut))
            }))
```

```
46
47              hijos <- matrix(parSapply(cluster, 1:rep, function(i) {
48                padres <- sample(1:tam, 2, replace = FALSE)
49                pos <- sample(2:(n-1), 1)
50                x <- p[padres[1],]
51                y <- p[padres[2],]
52                xy <- c(x[1:pos], y[(pos+1):n])
53                yx <- c(y[1:pos], x[(pos+1):n])
54                return(as.numeric(c(xy, yx)))
55              }), ncol = n, byrow = TRUE)
56
57              p <- rbind(p, mutaciones, hijos)
58              tam <- dim(p)[1]
59
60              clusterExport(cluster, "p")
61              clusterExport(cluster, "tam")
62
63              p$obj <- parSapply(cluster, 1:tam, function(i) {
64                return(sum(p[i,] * valores))
65              })
66
67              p$fact <- parSapply(cluster, 1:tam, function(i) {
68                return(sum(p[i,] * pesos) <= capacidad)
69              })
70
71              mantener <- order(-p[, (n + 2)], -p[, (n + 1)])[1:init]
72
73              p <- p[mantener,]
74              tam <- dim(p)[1]
75
76              assert(tam == init)
77
78              factibles <- p[p$fact == TRUE,]
79              mejor <- max(factibles$obj)
80              mejores <- c(mejores, mejor)
81
82            }
83
84            datos <- rbind(datos, data.frame(Replica = r, Optimo = optimo,
85                                             Generaciones = init, ProbMut = pm,
86                                             Reproducciones = rep, Iteraciones = tmax,
87                                             Objetivo = max(mejores)))
88          }
89        }
90      }
91    }
92  }
93  stopCluster(cluster)
94  endP=Sys.time()
```

En la figura 1 se puede observar en como la paralelización del experimento puede permitir minimizar los tiempos de ejecución. Asimismo, al incrementar la instancia se puede ver un comportamiento más abrupto en la ejecución normal, en comparación del paralelizado; que muestra ligeros cambios.

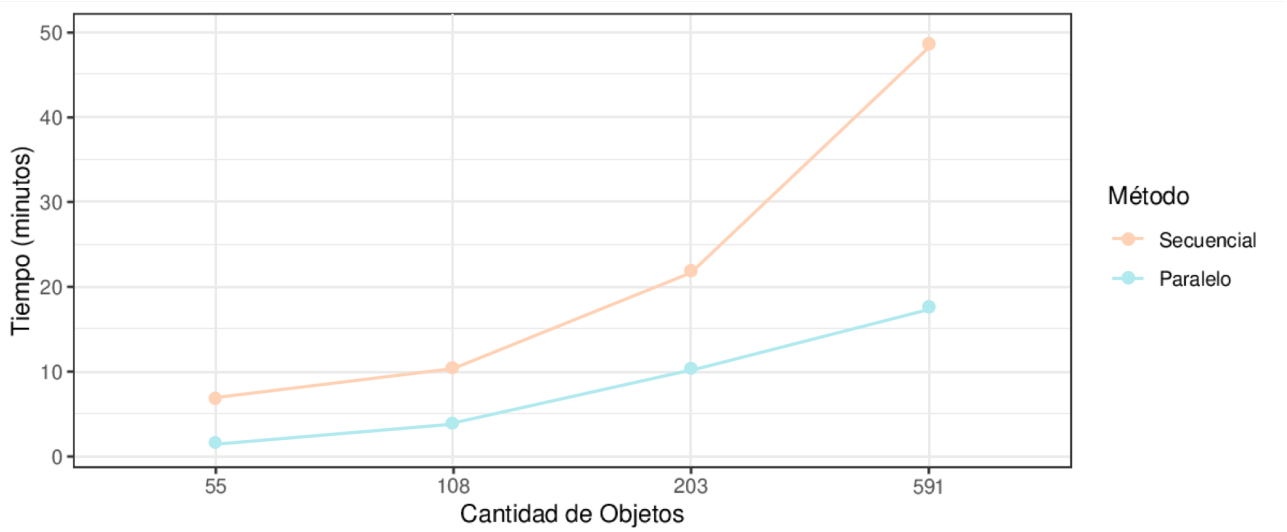

Figura 1: Comparación de ejecución de código

Con ayuda de la paquetería *ggplot2* [1] se obtuvo la visualización del error de la instancia creada, de modo que permite demostrar de manera cuantitativa el ruido del experimento. En la figura 2 se logra denotar que al incrementar las variables este tiene un ruido que se mantiene constante en comparación con el comportamiento de las otras cantidades de generaciones.

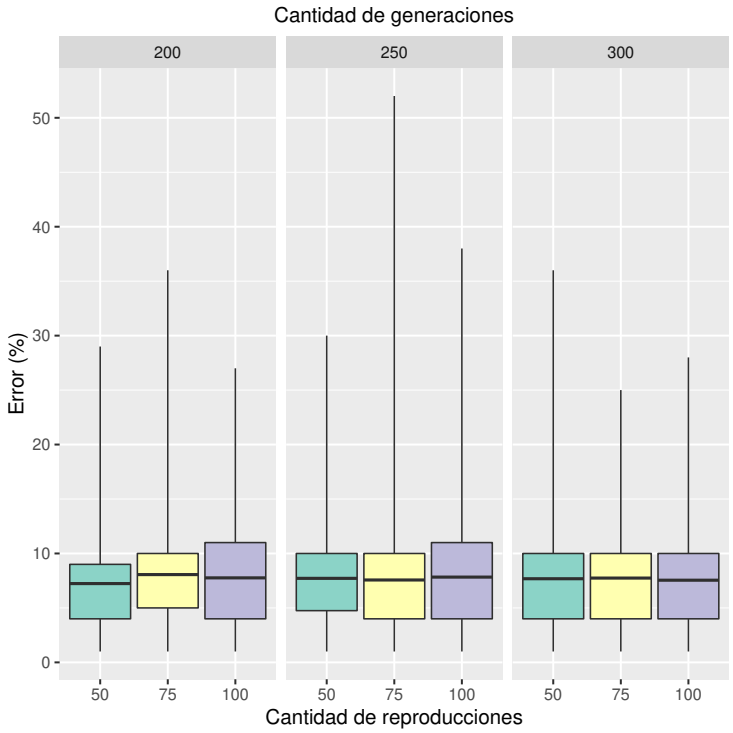

Figura 2: Porcentaje de error respecto a las instancias generadas

Referencias

[1] Hadley Wickham et al. Package 'ggplot2', 2019. URL https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf.

[2] Elisa Schaeffer. Práctica 10: algoritmo genético, 2019. URL https://elisa.dyndns-web.com/teaching/comp/par/p10.html.

[3] Ángel Moreno. Homework 10, 2018. URL https://github.com/angisabel44/Simulation/tree/master/Homework10.