

Algoritmo genético

Alberto Benavides

16 de octubre de 2017

Objetivos

1. Paralelizar el algoritmo genético de la práctica número diez del curso **R paralelo: simulación y análisis de datos** publicada por Schaeffer (n.d.) y estudiar los efectos en su tiempo de ejecución.

Simulación y resultados

El equipo usado en esta simulación contaba con sistema operativo Windows 10 Home Single Language, procesador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 2904MHz de dos núcleos principales y cuatro lógicos. El lenguaje de programación utilizado fue R.

A lo largo del código, se paralelizaron los ciclos que iteraban entre número de individuos y número de reproducciones, correspondientes a las funciones `poblacion.inicial`, `mutacion`, `reproduccion`, `objetivo` y `factible`, de las cuales, sólo la primera no pertenece propiamente al algoritmo genético. Las variables iniciales son 200 individuos (`init`) con 50 cromosomas (`n`) cada uno, una probabilidad de mutación de 0.05 (`pm`) y 50 reproducciones (`rep`) entre individuos por cada generación de un total de 50 generaciones (`tmax`). Las funciones se paralelizaron con dos núcleos físicos por `parSapply` de la librería `parallel`, como sigue:

```
poblacion.inicial <- function(n, tam) {  
  pobl <- matrix(rep(FALSE, tam * n), nrow = tam, ncol = n)  
  if(parallel){  
    a <- parSapply(cluster, 1:tam, function(i){  
      round(runif(n))  
    })  
    pobl <- t(a)  
  } else{  
    for (i in 1:tam) {  
      pobl[i,] <- round(runif(n))  
    }  
  }  
}
```

```

    }
  }
  return(as.data.frame(pobl))
}
[...]
```

```

clusterExport(cluster, "pm")
clusterExport(cluster, "mutacion")
clusterExport(cluster, "n")
clusterExport(cluster, "p")
a <- parSapply(cluster, 1:tam, function(i){
  if (runif(1) < pm) {
    mutacion(p[i,], n)
  }
})
a <- Filter(Negate(is.null), a)
a <- data.frame(matrix(unlist(a), nrow=50, byrow=T))
a <- t(a)
p <- rbind(p, a)
[...]
```

```

clusterExport(cluster, "p")
clusterExport(cluster, "tam")
clusterExport(cluster, "reproduccion")
a <- parSapply(cluster, 1:rep, function(i){
  padres <- sample(1:tam, 2, replace=FALSE)
  hijos <- reproduccion(p[padres[1],], p[padres[2],], n)
  hijo1 <- hijos[1:n] # primer hijo
  hijo2 <- hijos[(n+1):(2*n)] # segundo hijo
  return(rbind(hijo1, hijo2))
})
a <- data.frame(matrix(unlist(a), nrow=50, byrow=T))
a <- t(a)
p <- rbind(p, a)
rownames(p) <- 1:nrow(p)
[...]
```

```

clusterExport(cluster, "p")
clusterExport(cluster, "obj")
clusterExport(cluster, "fact")
clusterExport(cluster, "objetivo")
clusterExport(cluster, "valores")
clusterExport(cluster, "factible")
clusterExport(cluster, "pesos")
clusterExport(cluster, "capacidad")
a <- parSapply(cluster, 1:tam, function(i){

```

```

obj <- c(obj, objetivo(unlist(p[i,]), valores))
fact <- c(fact, factible(unlist(p[i,]), pesos, capacidad))
return(cbind(fact, obj))
})
a <- t(a)
colnames(a) <- c("fact", "obj")
p <- cbind(p, a)

```

Una muestra de los valores generados por el algoritmo se muestran en la figura 1 (p. 3) donde, aunque se aproximan, no logran dar con el valor óptimo en las generaciones definidas.

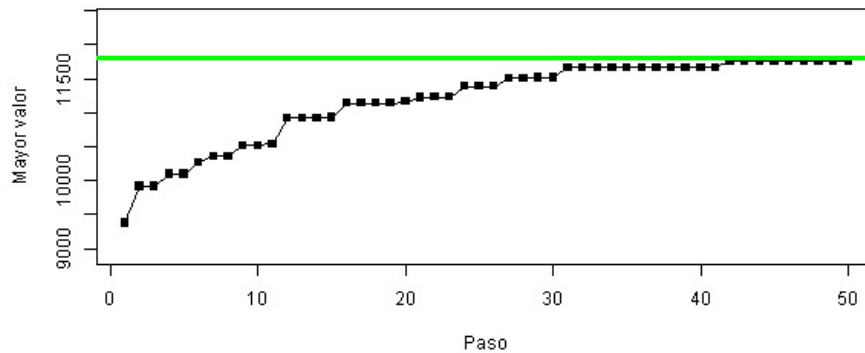


Figura 1: Valor óptimo en verde y valores generados por el algoritmo genético como cuadros en negro.

El estudio del tiempo que toma correr este algoritmo con y sin paralelización se hizo repitiendo el experimento en un ciclo de diez iteraciones paralelas y diez iteraciones no paralelas, cuyos diagramas de caja y bigotes pueden consultarse en la figura 2 (p. 4) y comprobar que el tiempo que toma en ejecutarse el algoritmo en paralelo es menor que el requerido por el mismo código sin paralelizar el experimento.

A estos tiempos se les hizo una regresión lineal con la función `lm(tiempo ~ tipo, data = times)` para conocer la relación que existe entre ambos. El tipo de corrida $p = [1, 2]$ y el tiempo de ejecución t están relacionados mediante la ecuación $t = -6.86p + 16.30$, de donde se tiene, por la pendiente $m = -6.86$, que al paralelizar se disminuye 6.86 veces el tiempo de ejecución, datos que pueden visualizarse en la figura 3 (p. 5).

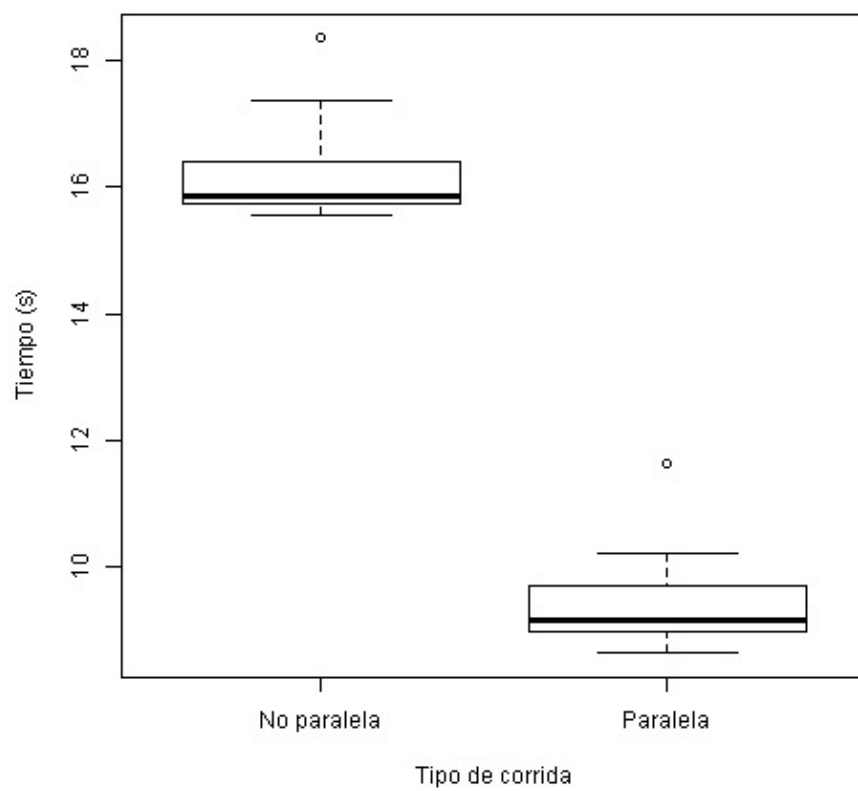


Figura 2: Diagramas de caja y bigotes de los tiempos de ejecución del experimento en paralelo y sin paralelizar.

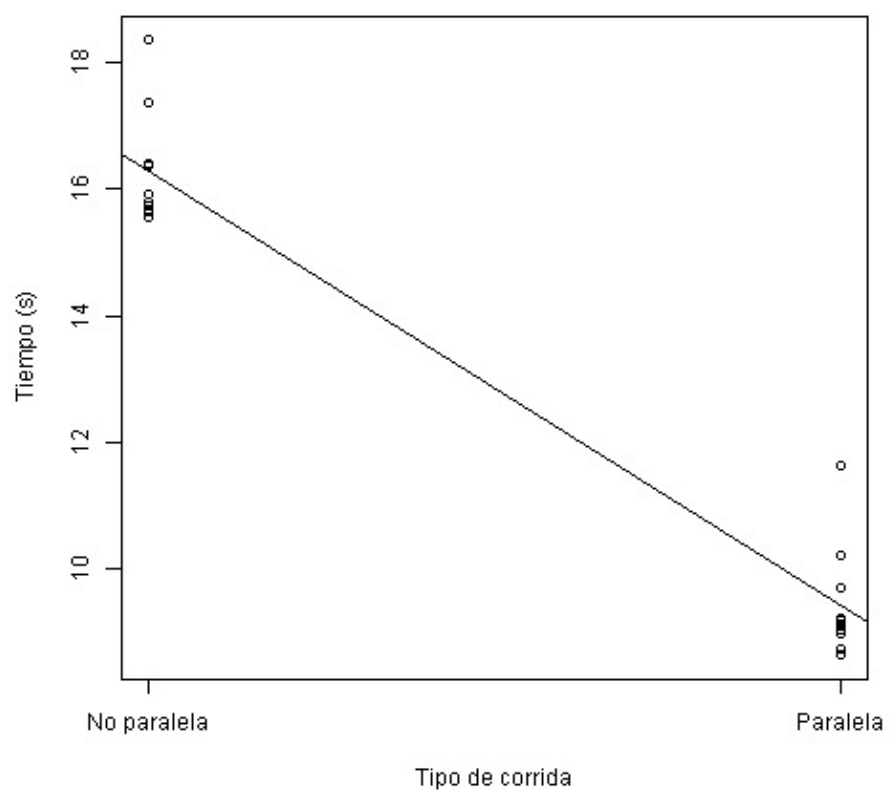


Figura 3: Relación lineal entre el tipo de corrida y el tiempo de ejecución.

Referencias

Schaeffer, Elisa. n.d. *Práctica 10: Algoritmo Genético*. <http://elisa.dyndns-web.com/teaching/comp/par/p10.html>.