Algoritmo Genético

Edson Raúl Cepeda Márquez

9 de abril de 2019

1. Objetivo

El objetivo principal de esta práctica es analizar y comparar los tiempos de ejecución de un algoritmo genético paralelizado empleado en la resolución del problema de la mochila cuando se cambian las reglas de los valores introducidos. Se hace uso del código escrito en el lenguaje de programación R [2] asi como del material de apoyo disponible en la página [3] de la Dra. Elsa Schaeffer.

2. Desarrollo

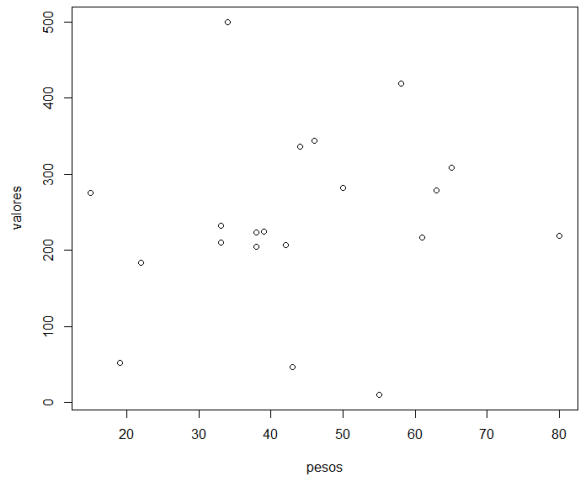
Se empieza por crear las nuevas reglas en los valores y los pesos. De esta manera se tienen tres instancias. La instancia no correlacionada conserva los valores tal y como se calculan en el código original. Para la instancia correlacionada se implementa la siguiente función.

```
generador.valores.correlacionados <- function(pesos,min,max) {
  n <- length(pesos)
  valores <- double()
  for (i in 1:n) {
    media <- pesos[n]
    desv <- runif(1)
    valores <- c(valores, pesos[i] + rnorm(1))
  }
  valores <- normalizar(valores) * (max - min) + min
  return(valores)
}
```

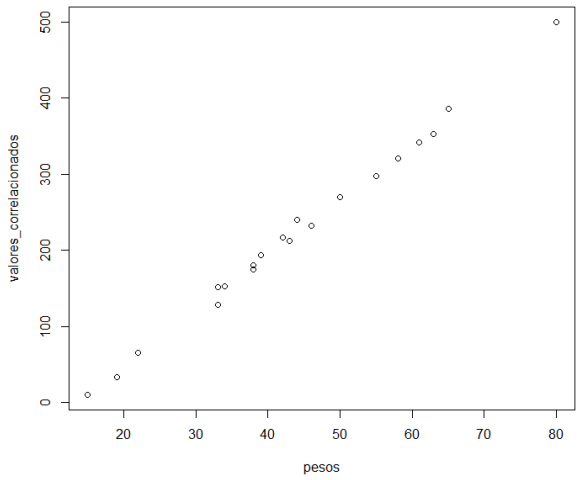
Ahora se guardan los valores para cada una de las instancias.

```
valores <- generador.valores(pesos, 10, 500)
valores_correlacionados <- generador.valores.correlacionados(pesos,10,500)
valores_inversos <- generador.valores.correlacionados(rev(pesos),10,500)
```

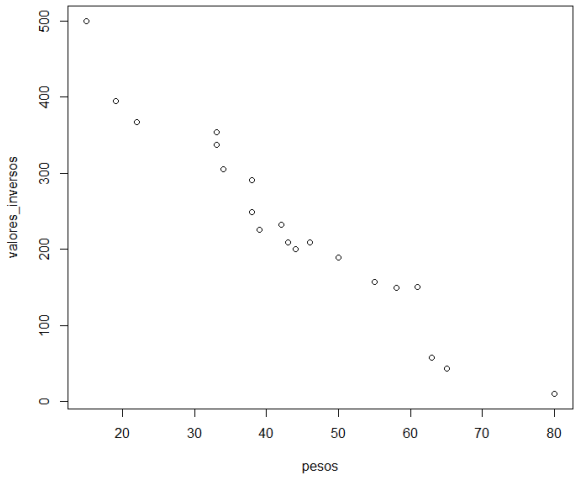
El resultado de crear estas nuevas instancias se puede observar en la figura 1.



(a) No correlacionados



(b) Correlacionados



(c) Inversamente correlacionados

Figura 1: Valores y pesos

Se procede a paralelizar las funciones más importantes para ahorrar tiempo de ejecución. Estas son las funciones en las que los genomas mutan, se reproducen, comprueban su objetivo y comprueban si son factibles. Se excluyen como funciones.

```
p.mutar <- function() {
  if (runif(1) < pm) {
    return(mutacion(p[i,], n))
  }
}
```

```r
repro <- function() {
  padres <- sample(1:tam, 2, replace=FALSE)
  hijos_t <- reproduccion(p[padres[1],], p[padres[2],], n)
  return(hijos_t)
}
```

```r
obj.f <- function() {
  obj_t <- double()
  obj_t <- c(obj_t, objetivo(p[i,], valores))
  return(obj_t)
}
```

```r
fact.f <- function() {
  fact_f <- integer()
  fact_f <- c(fact_f, factible(p[i,], pesos, capacidad))
  return(fact_f)

}
```

Las funciones previas se ejecutan paralelamente y e toman los tiempos que le toma a cada instancia con el comando `system.time`.

3. Resultados y conclusiones

Los tiempos de ejecución que le toma a cada instancia la no correlacionada, la correlacionada y la inversamente correlacionada, no tiene mucha diferencia. Cada una de las instancias se acerca considerablemente al resultado esperado dado por el algoritmo exacto. Existe un beneficio observable a la hora de paralelizar pero el agloritmo exacto sigue siendo más rápido. En la figura 2 se observa los tiempos que toma a cada instancia acabar sus iteraciones.

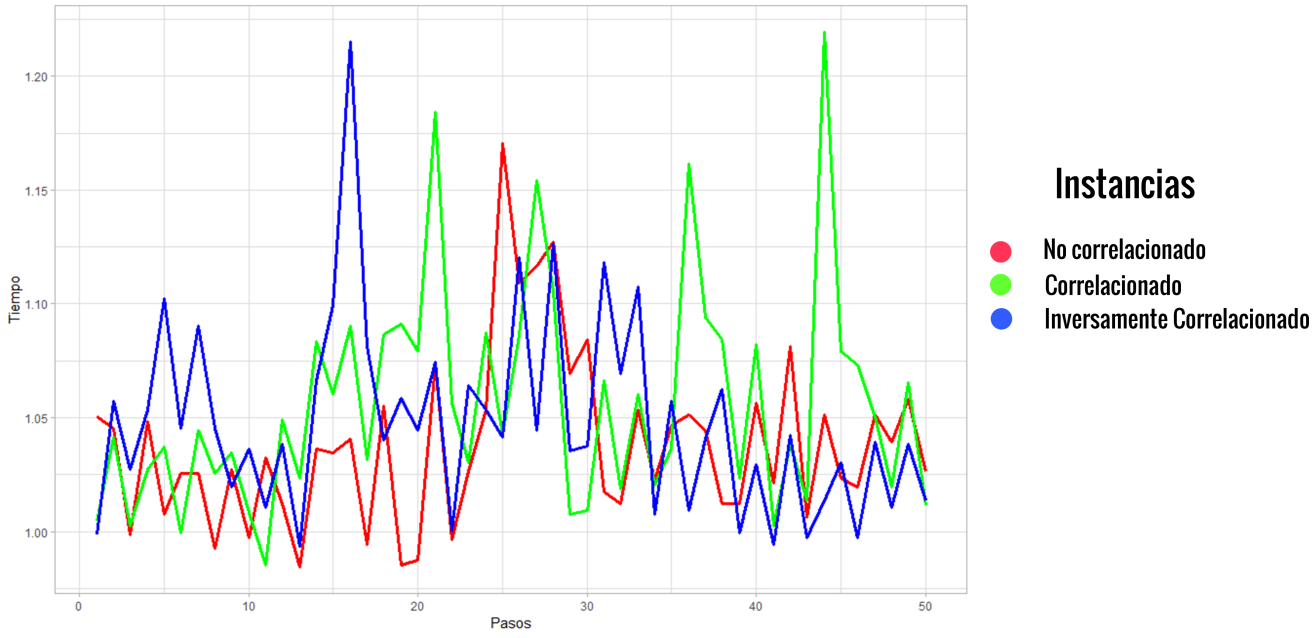


Figura 2: Tiempos medidos por cada instancia

Referencias

[1] Ángel Moreno: Algoritmo genético
https://github.com/angisabel44/Simulation/blob/master/Homework10/p10parsapply.R

[2] R: R Project, 2019
https://www.r-project.org/

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