Algoritmo genético

Francisco Gerardo Meza Fierro

1. Introducción

En esta práctica se trabajará con el problema de la mochila, un problema conocido de optimización, pero se implementará bajo un algoritmo genético con objetivos de comparar qué tan cercana se encuentra la solución obtenida con dicho algoritmo respecto a la solución óptima obtenida mediante el algoritmo pseudo-polinomial.

Se paralelizará el código propuesto con intenciones de reducir los tiempos de ejecución.

2. Paralelización del código

Desde el inicio del análisis de esta tarea, resultaba evidente lo tardado que es el código, por lo que paralelizarlo resultó ser tarea obvia.

Recordemos que el objetivo principal de paralelizar un código es el de minimizar el tiempo en que el código esté corriendo. Después de analizar el código propuesto se determinó que, con la ayuda de la función `parSapply`, se paralelizarían las funciones `factible`, `objetivo`, `mutación` y `reproducción` debido a que estas funciones desempeñan operaciones independientes entre ellas.

Para ver si la implementación paralela resultó satisfactoria logrando reducir los tiempos que en un inicio el código secuencial arrojaba, bajo una población inicial de doscientos, se hicieron cinco réplicas en ambos códigos y se registraron sus tiempos de ejecución con ayuda de la función `Sys.time()`. La figura 1 reporta estos tiempos.

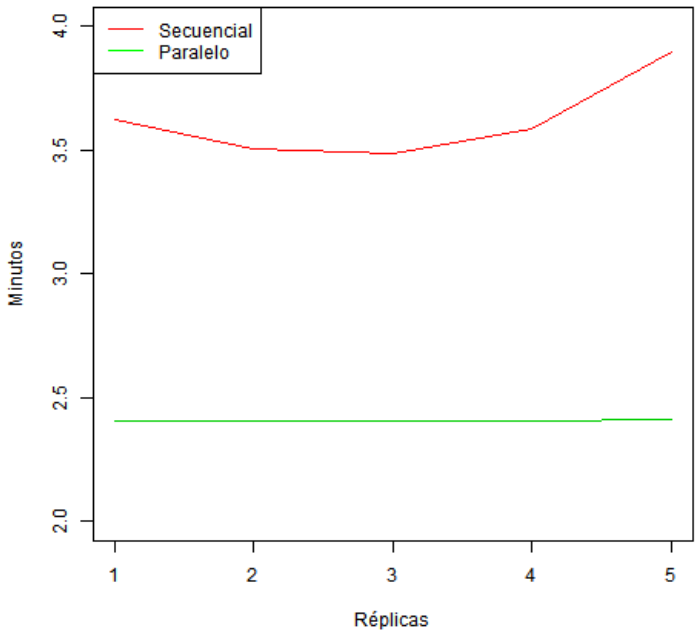


Figura 1: Tiempos efectuados por cada réplica

Es más que evidente que la paralelización resultó adecuada ya que se logró reducir los tiempos de cada réplica en poco más de un minuto.

En un acto independiente, se decidió reducir el número de la población inicial a un valor de cincuenta, con tal de trabajar con un número mucho menor de datos y así darle un poco de oportunidad al código secuencial de trabajar a la par contra el código secuencial. Para esto se crearon réplicas de manera análoga a las anteriores. Estos resultados se reportan en la figura 2

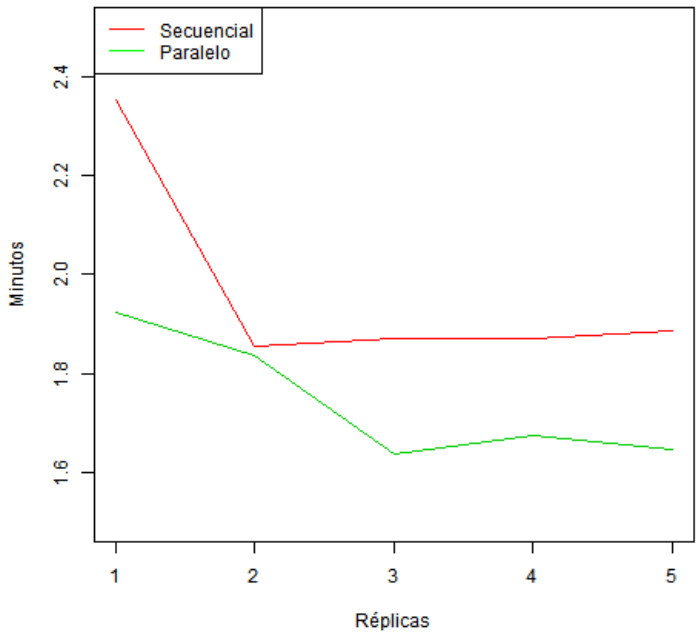


Figura 2: Tiempos efectuados por cada réplica

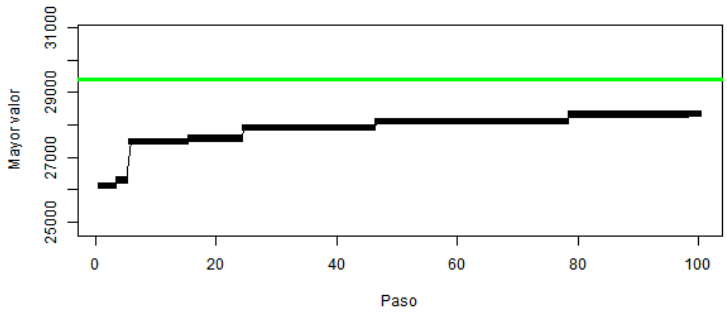
Aún con un valor muy chico de la población inicial, el paralelo sigue resultando mejor. Esto nos dice que la paralelización resultó ser adecuada y eficiente.

3. Selección de ruleta

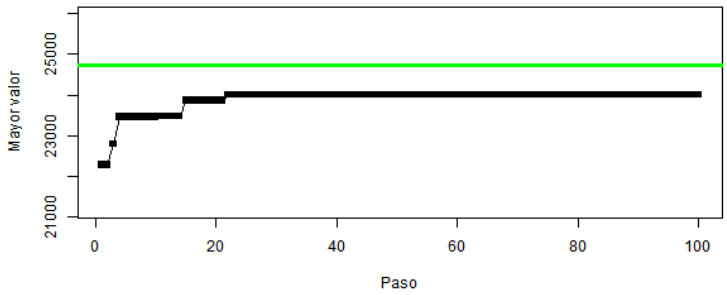
Con el código exitosamente paralelizado se efectuaron cambios en la selección de la solución, modificando las probabilidades que se toman para decidir si se es apto a ser padre. Se le dio una mayor probabilidad de ocurrencia a aquellos que fuesen más factibles y una probabilidad menor a los que no. Esto originó probabilidades de ocurrencia distintas a cada paso, las cuales se parametrizaron a valores entre 0 y 1 y estos valores son los que se usaron para la decisión de considerarlo como padre. Todo esto con la intención de que los padres que dan buenas soluciones sean capaces de crear hijos que también den buenas soluciones.

Al estar haciendo pruebas con esta nueva implementación para tratar de encontrar el óptimo, se registraron sus resultados y se compararon con los obtenidos en el código paralelizado. Todos estos resultados resultaron muy similares entre sí; la figura 3 muestra una de las múltiples comparaciones que se hicieron.

Para aclarar la duda de que los resultados fuesen iguales o diferentes, se hizo uso de la función `wilcox.test` que nos permite llevar a cabo la prueba estadística de Wilcox. Desafortunadamente esta prueba nos indicó que no existe alguna diferencia significativa entre estas dos implementaciones del código. Esto podría ser consecuencia de que los nuevos hijos creados por los mejores padres no necesariamente son factibles.



(a) Paralelo



(b) Paralelo y Selección de ruleta

Figura 3: Una comparación entre las dos implementaciones