Los *Grouping problems (GP)* son un tipo especial de problemas de optimización combinatorios, que se han vuelto famosos debido a su aplicación a problemas reales, estos problemas se clasifican como NP-hard. Esto significa que no hay ningún algoritmo que encuentre en tiempo polinómico una solución óptima.

Un algortimo diseñado para resolver un GP, busca la mejor distribución de n elementos de un conjunto V en D diferentes grupos ($1 \leq n \leq D$), de tal forma que un elemento este en un solo grupo. Algunos ejemplos de GP son: el problema de la mochila (Bin Packing), balanceo de carga (Load Balancing), problema de enrutamiento de vehículos (Vehicle Routing Problem), el problema del viajero (Traveling Salesperson Problem), entre otros.

Bin Packing (BP)

El problema BP en una dimensión consiste en empacar una serie de elementos con cierto peso en contenedores con una capacidad específica. El objetivo es minimizar el número de contenedores utilizados. Dentro de la literatura, se encuentra que la efectividad de los algoritmos se reduce, conforme se aumenta la capacidad del contenedor, también que las instancias que tenían un peso promedio de 30 % la capacidad del contenedor, fueron más complicadas de resolver, en cambio, las instancias con un peso promedio mayor al 50 % de la capacidad del contenedor, son más fáciles [1]. Se dice que hay una solución óptima para el BP cuando los contenedores utilizados se llenan sin dejar espacio vacío.

Existe una librería con instancias para este problema (BPPLIB - A Bin Packing Problem Library) [2] que contiene más de 1600 con distintos pesos y distintos tamaños de contenedores, además tiene soluciones con algoritmos exactos. Para esta primera parte, se escogió las instancias de E. Falkenauer [3], de las cuales el peso de los objetos está entre [20, 100], y el rango óptimo de la solución es de [46, 52] elementos por contenedor, y la capacidad máxima de los contenedores es de 150.

Experimentos

Para realizar el evaluador, primero se leyó el peso de cada elemento y el tamaño máximo de cada contenedor. Después, se revolvieron los elementos con la función *random_shuffle*, y se fueron sumando los elementos hasta que llegara al tope del contenedor o lo sobrepasara, luego se añadía otro contenedor, y se llenaba con los elementos restantes. Este procedimiento se realizó N veces (N = 100,000). Se encontró el valor mínimo entre los N resultados y se repitió el experimento con otra semilla aleatoria 100 veces utilizando la librería **openMP**, y un clúster para que el resultado fuera más rápido.

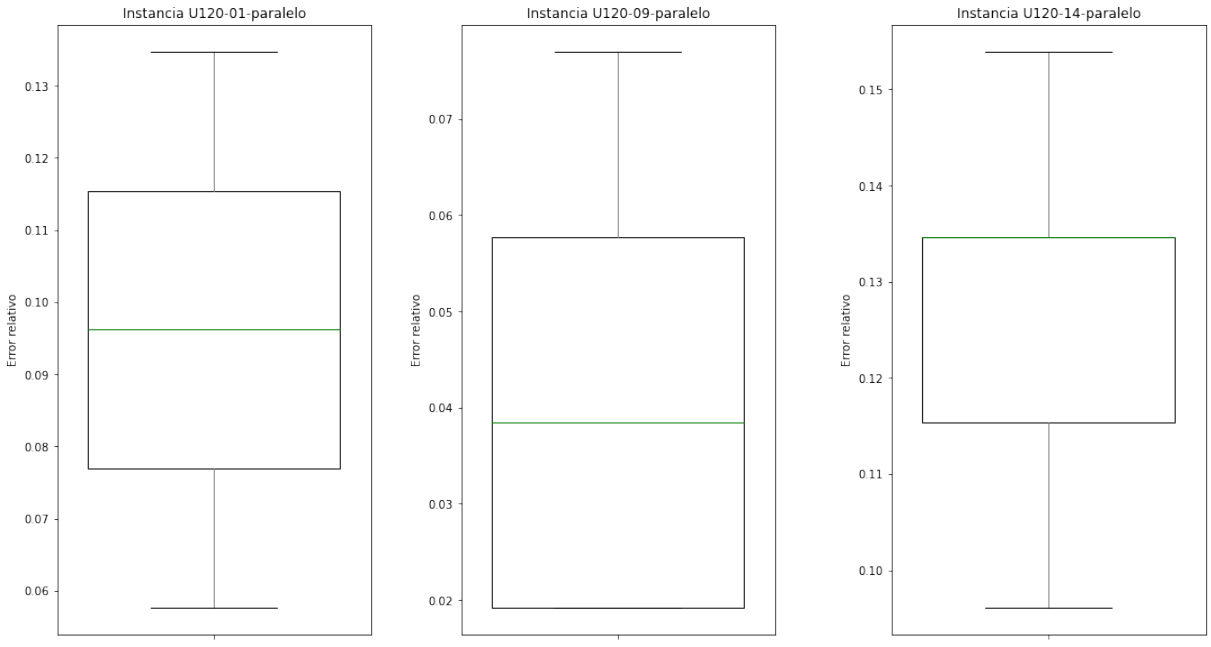

Figura 1: Boxplot primeros resultados

En la Fig.1 se muestra el error relativo y se calculó de la forma:

$$\text{Error relativo} = \frac{\text{Mínimo obtenido} - 52}{52},$$

utilizando el valor más grande, asimismo, se observa que la mayoría de las instancias hay una diferencia mínima, y que los valores máximos no son tan distintos. Además, dado que los cuantiles son equidistantes, asemejan una distribución uniforme, que coincide con lo conocido.

Conclusiones

El algoritmo utilizado fue de complejidad n, ya que utilizó la función *random_shuffle*, e hizo sumas en cada iteración. Sin embargo, ningún resultado alcanzó los encontrados en la literatura de [46, 52], por lo que no se encontraron soluciones óptimas.

Referencias

[1] Guadalupe Carmona-Arroyo, Jenny Betsabé Vázquez-Aguirre, and Marcela Quiroz-Castellanos. One-dimensional bin packing problem: An experimental study of instances difficulty and algorithms performance. *Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications*, pages 171–201, 2021.

[2] Maxence Delorme, Manuel Iori, and Silvano Martello. Bpplib: a library for bin packing and cutting stock problems. *Optimization Letters*, 12:235–250, 2018.

[3] Emanuel Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2:5–30, 1996.