Tarea 2-Optimización Estocástica

1st Daniel Vallejo Aldana
Maestría en Ciencias de la Computación
Centro de Investigación en Matemáticas
daniel.vallejo@cimat.mx

***Resumen*—En el presente trabajo se implementan heurísticas de trayectoria el problema de las p-medianas, un problema de optimización combinatoria.Para esto se implementó el algoritmo de Stochastic Hill Climbing en donde se busca encontrar los mejores vecinos de una solución dada a partir de una solución inicial aleatoria y de una solución generada con alguna heurística constructiva que se describirá en este trabajo.**

***Index Terms*—p-medianas, grafos, búsqueda local, heurística constructiva.**

I. Introducción

De acuerdo a [1], el problema de las p-medianas consiste en localizar p-instalaciones (medianas) dentro de una red (grafo) con $|V|$ nodos y $|E|$ aristas de tal forma que minimicemos la suma de todas las distancias de cada vértice v del grafo $G(V, E)$ a su instalación más cercana. Varios métodos se han utilizado para la resolución de este problema, en [1] se propone un método basado en relajación Lagrangiana, en [2] se analiza resolver este problema con algoritmos de métodos de partición como lo son la Variable Neighbor Search (VNS) o la búsqueda tabú. Así mismo el trabajo de Loranca et.al. [2], es usado como comparación de los resultados obtenidos en el presente trabajo.

En el presente trabajo se implementa una heurística constructiva para iniciar de una solución prometedora en lugar de una solución completamente aleatoria. Así mismo se implementa el algoritmo de búsqueda local de Stochastic Hill climbing para encontrar mejores soluciones a una ya existente. Se realiza una comparación de los resultados obtenidos con el método de búsqueda local contra el generar soluciones aleatorias usado en la Tarea 1.

II. Metodología

II-A. Heurística Constructiva

Para la construcción de la heurística constructiva consideremos la matriz de costo $C \in Mat_{|v|\times|v|}(\mathbb{R})$. Para cada nodo $n \in V$ consideremos el costo promedio de ir de ese nodo a cualquier otro, para este caso sea c dicho costo entonces definimos c de la forma

$$c(n) = \frac{1}{|V|}\sum_{i=1}^{|V|} C[n][i]$$

Consideramos los p primeros nodos tales que su costo c sea mínimo y hacemos un shuffle aleatorio de dichos vectores. De esa forma construimos la solución inicial basada en las distancias promedio de los nodos a sus respectivos vecinos.

II-B. Algoritmo de búsqueda local

Para el algoritmo de búsqueda local necesitamos definir los elementos utilizados. Por la definición de la solución en la tarea anterior, consideramos una solución como un vector de tamaño p que contiene números diferentes que corresponden a los nodos que serán usados como las medianas. Así mismo debemos de guardar un vector con los nodos que no están siendo utilizados como posibles medianas.

II-B1. Evaluación incremental: Para la parte de evaluación incremental se creó un vector de heaps actualizables en donde cada elemento del vector corresonde a un heap actualizable para cada nodo. Ese heap es de tamaño p y la ventaja de esta estructura de datos es que permite el cálculo del valor de la función objetivo de forma más eficiente. Así mismo nuestro descriptor corresponde a una estructura del tipo

```
struct {
int u;
int nu;
} descriptor;
```

Donde `u` corresponde a la posición dentro del vector solución que se va a modificar y que será intercambiada por la posición `nu`. Con esto creamos los vecinos y hacemos uso de la evaluación incremental.

El algorimtmo de búsqueda local se muestra a continuación

Algorithm 1 FindBetterSol

Require: Solución inicial s, C, G Donde C es el contenedor de colas de prioridad y G el grafo
Ensure: Mínimo local s^* y $flag$ que indica si se encontró una mejor solución
`flag=False`
Tomar un aleatorio entre k 0 y p
Crear un vector de descriptores `des`
Costo total $cc = currentCost(C)$
random shuffle de los descriptores `des`
for d in descriptor **do**
  Intercambia posición k con los nodos no usado
  if nuevo costo ¡cc **then**
    cc=nuevo costo
    $modificaSolucion(s)$
    flag=True
    break
  end if
end for
return s,flag

III. Resultados

III-A. Instancia 1

Probamos los algoritmos diseñados en la presente tarea para la primera instancia la cual consta de 100 nodos, 200 aristas y cuyo valor de $p = 5$. El boxplot obtenido para cada uno de

Busqueda local
Require: s,C,G
Ensure: s^*
  flag=True
  while flag **do**
    flag=FindBetterSol(s,C,G)
  end while

los métodos de encontrar las mejores soluciones candidatas se muestra a continuación en la figura 1

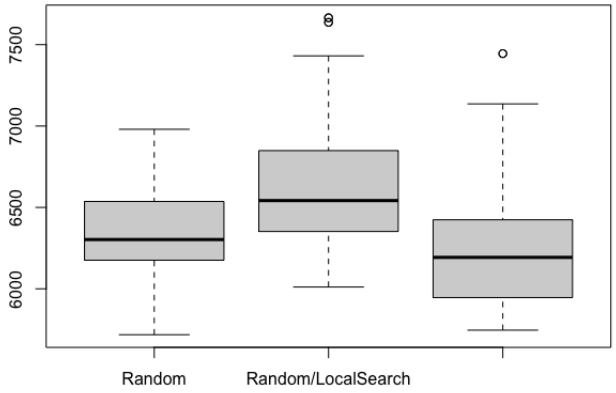

Figura 1. Boxplot de las soluciones aleatorias (izquierda), boxplot de los valores de las soluciones aleatoriamente inicializadas pero con búsqueda local (centro) y soluciones con heurística constructiva (derecha)

Podemos observar que en este caso el uso de heurísticas constructivas ayudó a mover la mediana de las mejores soluciones candidatas. En este caso usar soluciones aleatorias y posteriormente búsqueda local no ayuda demasiado a encontrar una mejor solución sin embargo, tampoco empeora los resultados de forma considerable.

La tabla de comparación se muestra a continuación

Método	**Media**	**Min**	**Max**
Aleatorio	6356	5718	6980
Aleatorio + búsqueda local	6613	6011	7665
Heurística constructiva + búsqueda local	6249	5746	7445
Mejor conocido		5891	

Cuadro I
TABLA COMPARATIVA DE RESULTADOS OBTENIDOS

En este caso vemos que dos de nuestras heurísticas lograron vencer el mejor resultado conocido para el caso de la primera instancia. Para el caso de la primera instancia vemos que la heurística constructiva dio buenos resultados.

Al comparar la tabla de tiempos de ejecución en cuanto a tiempo de búsqueda local en segundos podemos notamos lo siguiente.

Método	**Media**	**Min**	**Max**
Aleatorio + búsqueda local	0.04506	0.01757	0.11246
Heurística constructiva + búsqueda local	0.03492	0.01773	0.08106

Cuadro II
TABLA COMPARATIVA DE TIEMPOS DE BÚSQUEDA

Los boxplot correspondientes a los tiempos de búsqueda se muestran a continuación.

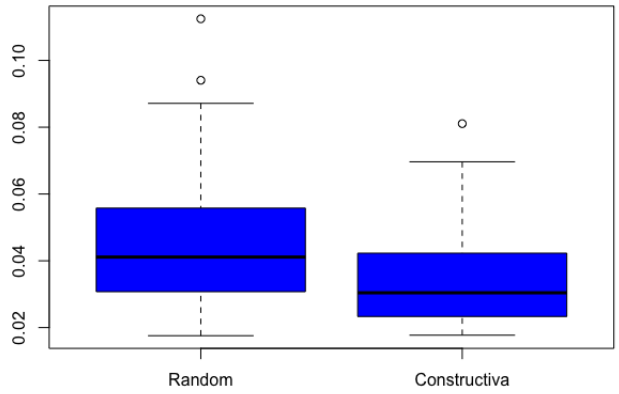

Figura 2. Comparación de tiempos que dura la búsqueda local en la instancia

III-A1. Instancia 2: La instancia 2 descrita en [1] corresponde a una colección de 200 nodos, 800 aristas y $p = 40$. Los resultados obtenidos por el algoritmo en cuanto a valor de la función objetivo se muestran a continuación.

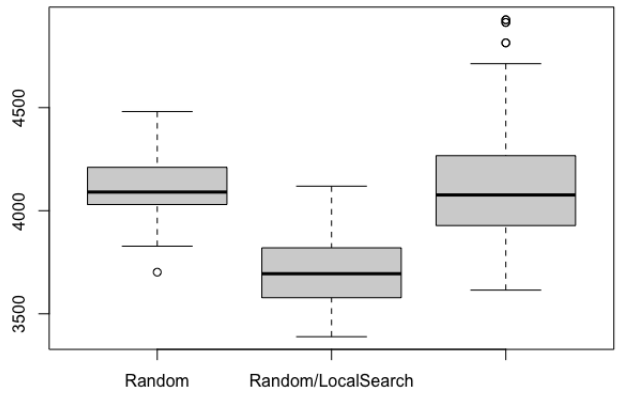

Figura 3. Boxplot de las soluciones aleatorias (izquierda), boxplot de los valores de las soluciones aleatoriamente inicializadas pero con búsqueda local (centro) y soluciones con heurística constructiva (derecha)

La tabla comparativa en cuanto a valor de la función objetivo se muestra en el cuadro III

Método	**Media**	**Min**	**Max**
Aleatorio	4090	3702	4481
Aleatorio + búsqueda local	3709	3389	4119
Heurística constructiva + búsqueda local	4126	3615	4926
Mejor conocido		2837	

Cuadro III
TABLA COMPARATIVA DE RESULTADOS OBTENIDOS

En este caso podemos ver que la heurística constructiva ayuda a encontrar mejores soluciones sin embargo inicializar de forma

aleatoria ciertas soluciones y después proceder con búsqueda local da los mejores resultados obtenidos hasta el momento. Si bien no se ha logrado encontrar una solución similar a la mejor solución obtenida, se está más cercano que solamente evaluando soluciones aleatorias.

Al comparar la tabla de tiempos de ejecución en cuanto a tiempo de búsqueda local en segundos podemos notamos lo siguiente.

Método	**Media**	**Min**	**Max**
Aleatorio + búsqueda local	0.4124	0.1452	1.9832
Heurística constructiva + búsqueda local	0.8740	0.1663	2.3605

Cuadro IV
TABLA COMPARATIVA DE TIEMPOS DE BÚSQUEDA

Los boxplot correspondientes a los tiempos de búsqueda se muestran a continuación.

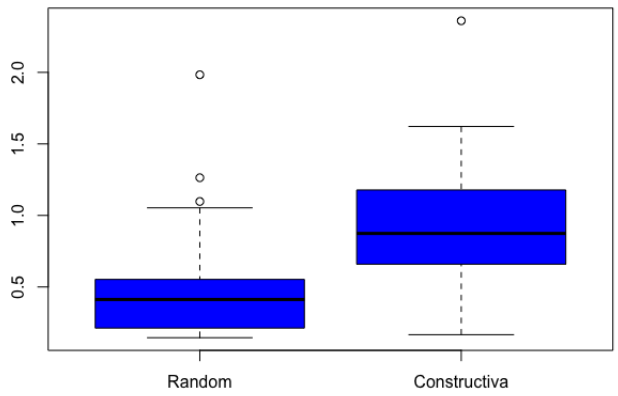

Figura 4. Comparación de tiempos que dura la búsqueda local en la instancia

III-A2. Instancia 3: La instancia 3 es la instancia más grande evaluada en el presente problema, consta de 800 nodos con 12800 aristas y valor de $p = 80$. Los resultados de la función objetivo se muestran a continuación.

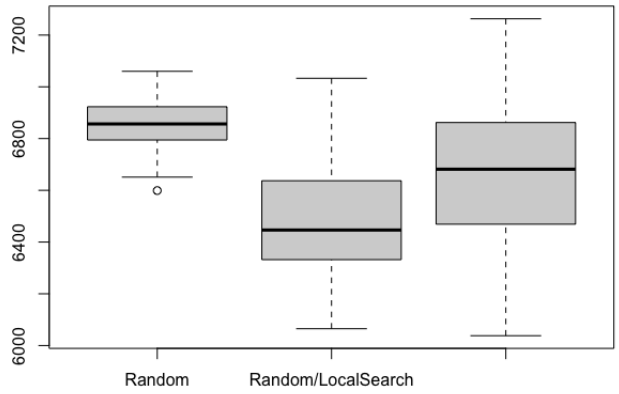

Figura 5. Boxplot de las soluciones aleatorias (izquierda), boxplot de los valores de las soluciones aleatoriamente inicializadas pero con búsqueda local (centro) y soluciones con heurística constructiva (derecha)

La tabla comparativa en cuanto a valor de la función objetivo se muestra en el cuadro III

Método	**Media**	**Min**	**Max**
Aleatorio	6856	6599	7060
Aleatorio + búsqueda local	6446	6065	7033
Heurística constructiva + búsqueda local	6682	6038	7263
Mejor conocido		5057	

Cuadro V
TABLA COMPARATIVA DE RESULTADOS OBTENIDOS

En este caso vemos que tanto las soluciones inicializadas con la heurística constructiva como con las soluciones aleatorias donde después se les aplica búsqueda local dieron mejores resultados que aquellas soluciones donde solo se generan soluciones aleatorias.

Podemos notar que entre más grande es la instancia que estamos tratando mayor es la diferencia entre utilizar algún tipo de heurística para poder encontrar mejores soluciones. En este caso el hecho de utilizar búsqueda local marca una gran diferencia en las instancias más grandes.

Al comparar la tabla de tiempos de ejecución en cuanto a tiempo de búsqueda local en segundos podemos notamos lo siguiente.

Método	**Media**	**Min**	**Max**
Aleatorio + búsqueda local	31.67	8.26	75.70
Heurística constructiva + búsqueda local	46.60	8.54	111.53

Cuadro VI
TABLA COMPARATIVA DE TIEMPOS DE BÚSQUEDA

Los boxplot correspondientes a los tiempos de búsqueda se muestran a continuación.

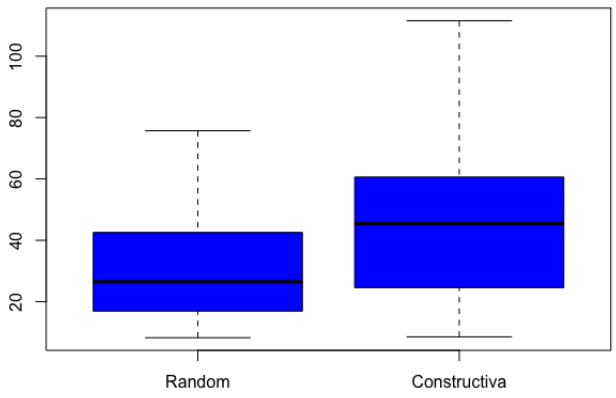

Figura 6. Comparación de tiempos que dura la búsqueda local en la instancia

IV. Conclusión

Con base en los experimentos realizados en el presente trabajo podemos notar que el uso del algoritmo de búsqueda local ayudó a encontrar mejores soluciones de las que ya se tenían consideradas en la tarea 1. Así mismo vemos que con base en los resultados experimentales, el uso de metaheurísticas constructivas no ayuda mucho a encontrar una mejor solución que iniciando con soluciones aleatorias. Así mismo vemos que los tiempos de ejecucuión usando metaheurísticas constructivas son en promedio mayores a los que usan una solución aleatoria como solución inicial.

Referencias

[1] John E Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21(2):270–273, 1985.

[2] María Beatríz Bernábe Loranca, Jorge A Ruiz-Vanoye, Rogelio González Velázquez, Marco Antonio Rodríguez Flores, and Martín Estrada Analco. P-median: A performance analysis. *Res. Comput. Sci.*, 88:19–30, 2014.