Tarea 1-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 implementa una forma de evaluar el problema de las p-medianas, un problema de optimización combinatoria.Para esto se implementó el algoritmo de Dijkstra para encontrar el costo real más pequeño para ir de un nodo $u \in G$ a un nodo $v \in G$. La evaluación se realizó sobre soluciones aleatoriamente generadas.**

***Index Terms*—p-medianas, grafos, Dijkstra, cómputo paralelo**

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 preente trabajo.

En el presente trabajo se crea una forma de representación del grafo de entrada $G(V, E)$ obtenido de la base de instancias OR-Lib para el problema de uncapacited p-medians, basada en listas de listas. Así mismo se implementa el método de Dijkstra para calcular el mínimo costo de un nodo $u \in G$ a un nodo $v \in G$. Finalmente se define una forma de generar soluciones aleatorias basadas en permutaciones y se evalúan dichas soluciones con la función de costo definida en [1]. Dichos resultados se comparan con los resultados obtenidos en [2].

II. Metodología

II-A. Representación de la Instancia

Para el problema de las p-medianas consideramos como instancia a un grafo G que cumple con las siguientes propiedades.

$$C(i, j) = C(j, i) \text{ es decir } G \text{ es un grafo bidirigido}$$

Donde $C \in Mat_{|v|\times|v|}(\mathbb{R})$ es la matriz inicial de costo antes de ser procesada por el algoritmo de Dijkstra para obtener la matriz de costo total. Si no existe conexión directa de i a j entonces $C(i, j) = \infty$.

En nuestro caso representamos el grafo G como una instancia `list(pair (int,int) )` de la STL. La matriz de costo parcial C y la matriz de costo total T se representan con tipo de datos `vector ((vector (int)))` .

II-A1. Obtención de la matriz de costo total T: De acuerdo a [1], para obtener la matriz de costo total T donde $T(i, j)$ representa el mínimo costo de ir del nodo i al nodo j en G, se utiliza el algoritmo de Floyd. En nuestro caso utilizaremos el algoritmo de Dijkstra para encontrar la matriz de costo total T. Para esto necesitamos correr el algoritmo desde $|V|$ fuentes diferentes para poder calcular dicha matriz. Para el algoritmo de Dijkstra se utilizaron colas de prioridad por lo que la complejidad del algoritmo es $O(|E| + |V| \log |V|)$ por lo que la construcción de la matriz de costo total es de orden $O(|V|(|E| + |V| \log |V|))$

El algoritmo de Dijkstra con colas de prioridad se muestra a continuación.

```
function Dijkstra(Graph, source):
    dist[source] ← 0

    create vertex priority queue Q

    for each vertex v in Graph.Vertices:
        if v ≠ source
            dist[v] ← INFINITY
            prev[v] ← UNDEFINED

        Q.add_with_priority(v, dist[v])


    while Q is not empty:
        u ← Q.extract_min()
        for each neighbor v of u:
            alt ← dist[u] + Graph.Edges(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u
                Q.decrease_priority(v, alt)

    return dist, prev
```

Figura 1. Algoritmo de Dijkstra con colas de prioridad

Así vemos que $T[i] = Dijkstra(G, i)$.

II-B. Creación y Evaluación de soluciones candidatas

Para el presente trabajo creamos una clase `Solucion`, la cual consta de un vector de tamaño p donde cada entrada corresponde a un número i tal que $1 \leq i \leq |V|$ que corresponde a una posible mediana que pudiera elegirse. Por construcción, se tiene que el conjunto de medianas no puede contener números repetidos. Para construir el conjunto de p medianas consideramos un vector de dimensión $|V|$ tal que $a[i] = i$. Realizamos una permutación aleatoria $P(a)$ y consideramos los primeros p elementos de dicha permutación. Para la construcción de la solución se ocupa una permutacion aleatoria lo cual tiene complejidad $O(|V|)$.

Para la evaluación se crea una matriz de $E_{p\times|V|}(\mathbb{R})$ donde cada entrada $E[k][j]$ representa el costo de $P(a)[k]$ a j. Posteriormente consideramos el mínimo por columnas de dicha matriz y sumamos los elementos del vector resultante. La creación de la matirz y la obtención del mínimo es de $O(p|V|)$ y la suma es de $O(|V|)$. Dicha función de costo es la utilizada en [1].

III. Resultados

III-A. Primera Instancia

La primera instancia evaluada consta de un conjunto de 100 nodos con 200 aristas y $p = 5$. Generamos 100 ejecuciones en

donde en cada una se generaban 100000 soluciones candidatas válidas y las evaluamos de acuerdo a la función de costo considerada en el presente proyecto para posteriormente elegir la mejor solución de las 100000 consideradas, la generación y evaluación de soluciones candidatas se realizaron con la librería `openmp` cuya función es realizar ejecuciones en paralelo. La computadora utilizada para el presente trabajo es una `Alienware Aurora R9 ©` con procesador `Intel Core i9 ©`.

Los resultados obtenidos sobre la primera instancia se muestran en la tabla I.

MinCost [2]	MinCost(ours)	MaxCost (ours)
5891	**5718**	6980

Cuadro I
MEJOR SOLUCIÓN OBTENIDA POR NUESTRO MÉTODO Y PEOR SOLUCIÓN OBTENIDA COMPARADA CON LA MEJOR SOLUCIÓN CONOCIDA

El boxplot correspondiente a estos resultados se muestra en la figura 2.

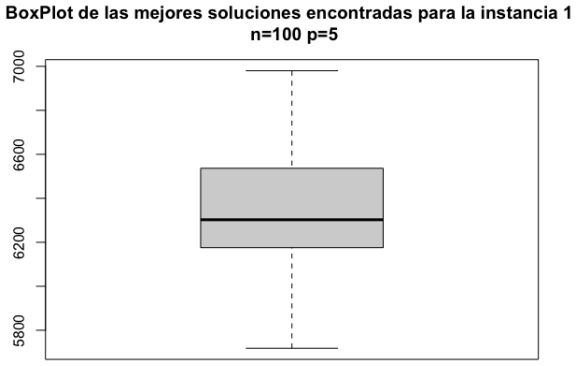

Figura 2. Boxplot de la primera instancia

III-B. Segunda Instancia

La segunda instancia evaluada consta de un grafo G con 200 nodos, 800 aristas y $p = 40$. Los resultados obtenidos sobre esta instancia se muestran a continuación.

MinCost [2]	MinCost(ours)	MaxCost (ours)
2734	3702	4481

Cuadro II
MEJOR SOLUCIÓN OBTENIDA POR NUESTRO MÉTODO Y PEOR SOLUCIÓN OBTENIDA COMPARADA CON LA MEJOR SOLUCIÓN CONOCIDA

Podemos notar que en este caso al ser p más grande no fue tan sencillo para el algoritmo escoger de forma aleatoria las medianas para obtener una buena solución.

El boxplot que representa los costos de las mejores soluciones encontradas se muestran en la figura 3.

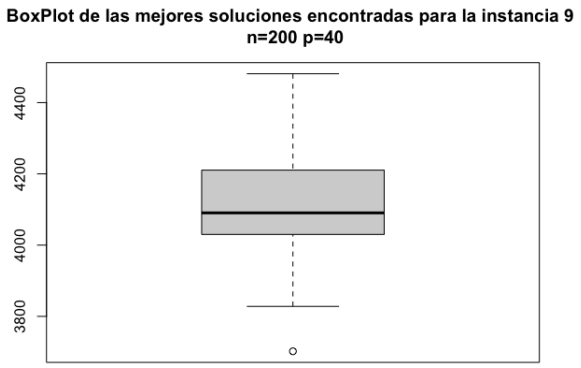

Figura 3. Boxplot de la segunda instancia

III-C. Tercera Instancia

La segunda instancia evaluada consta de un grafo G con 800 nodos, 12800 aristas y $p = 80$. Los resultados obtenidos sobre esta instancia se muestran en el cuadro III

MinCost [2]	MinCost(ours)	MaxCost (ours)
5057	6599	7060

Cuadro III
MEJOR SOLUCIÓN OBTENIDA POR NUESTRO MÉTODO Y PEOR SOLUCIÓN OBTENIDA COMPARADA CON LA MEJOR SOLUCIÓN CONOCIDA

El boxplot que representa los costos de las mejores soluciones encontradas se muestran en la figura 4

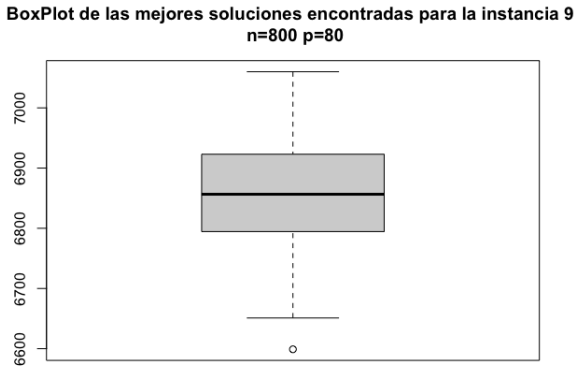

Figura 4. Boxplot de la tercer instancia

Podemos ver que cuando el número de nodos crecen así como el número p, es más complicado encontrar una solución aleatoria que sea buena en comparación con la mejor solución conocida.

IV. CONCLUSIÓN

Con base en los resultados obtenidos en el presente proyecto vemos que el problema de las p-medianas es un problema cuya complejidad crece a medida que el número de nodos y el valor de p crecen. Por tal razón no es factible tratar de resolverlos usando solamente generadores aleatorios sino métodos más eficientes para encontrar buenas soluciones candidatas.

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.