Heurística constructiva para el problema de recogida de basuras

Oscar Mauricio Cepeda Valero[1]

Universidad de los Andes
Cra 1 Nº 18A - 12, 28922, Bogotá, Colombia
om.cepeda86@uniandes.edu.co

Abstract

El problema de ruteo sobre arcos es uno de los problemas más trabajados en el área de optimización, dado que cuenta con unas características de complejidad que lo hacen difícil de solucionar. Adicionalmente, es un problema con grandes aplicaciones en la vida diaria, como lo es en la entrega de mensajería masiva, recolección de basuras, censado, recolección de basuras, entre otros. De esta manera, este documento presenta el diseño y aplicación de una heurística constructiva para el problema de recogida de basuras. Para el diseño de la misma se tomo de base el método del vecino cercano, de esta manera se obtuvo una heurística que arroja resultados aceptables con un algoritmo sencillo. Los resultados obtenidos se validaron contra la solución óptima. Toda la construcción del algoritmo se desarrollo en Python. La heurística se evaluó en 7 instancias reportadas en literatura. En general la heurística arrojo buenos resultados en las 7 instancias, obteniendo valores variaciones desde el 2.2% al 18.0% respecto al óptimo.

1 Introducción

Un problema común de optimización combinatoria que tiene una diversidad de aplicaciones para las situaciones de recogida y distribución es el Problema de ruteo sobre arcos capacitado (CARP). Por ejemplo, en el problema de la recogida de residuos [1], los camiones capacitados salen de un depósito para recoger los residuos que están dispersos por las calles. Otra aplicación consiste en el problema de la limpieza de las carreteras en invierno, los vehículos cargados de sal salen para esparcirla por las distintas carreteras [6]. Adicionalmente, este problema sigue siendo de gran interés, en una revisión reciente se encontraron más de 230 investigaciones que tratan de métodos y variaciones del problema de ruteo en arcos [3].

Estas son algunas de varias situaciones en las que se ha aplicado CARP, sin embargo como se observa aún existen muchas opciones para abordar el problema. Una de las diversas opciones que se han utilizado para resolver el CARP se tratan de los métodos metaheurísticos. Por ello, en instancias grandes se han abordado heurísticas como Path Scanning, el método de división de Ulusoy, metaheurísticas como busqueda Tabu, recocido simulado y al algoritmos genéticos, entre otros [2].

Este ensayo se centra principalmente en la construcción de una heurística constructiva, la cual se evaluó sobre las 23 instancias de Golden y Wong [5]. Está heurística se basó en el vecino cercano, el cual es uno de los métodos más utilizados para la construcción de soluciones a problemas combinatorios, y se ha aplicado comúnmente en problemas tanto de ruteo en arcos como de ruteo en nodos.

La estructura de este artículo es la siguiente. La formulación del problema y la metodología para abarcarlo se presenta en la Sección 2. Los resultados de la aplicación del modelo en 7 instancias se muestran en la Sección 3. La discusión de los resultados se presentan en la sección 4, y la sección 5 presenta las principales conclusiones del estudio.

1.1 Formulación CARP

El problema de ruteo sobre arcos capacitado (CARP) es una versión del problema de enrutamiento de arco en el que cada vehículo está sujeto a una restricción de capacidad. La investigación reciente se ha centrado en crear y probar algoritmos heurísticos que resuelvan aproximadamente el CARP, esto debido a la dificultad computacional del problema.

De igual manera diferentes formulaciones matemáticas han sido propuestas, pero aún así es un problema difícil de solucionar en instancias grandes. La función objetivo del problema corresponde a la minimización de la suma de los costos de los arcos recorridos prestando servicio más los costos de los arcos recorridos sin prestar servicio [4]. La función objetivo se presenta a continuación:

$$\min Z : \sum_{p\in\mathcal{I}}\sum_{e\in\mathcal{E}} c_e x_e^p + \sum_{p\in\mathcal{I}}\sum_{e\in\mathcal{E}} c_e y_e^p \tag{1}$$

En la Ecuación 1 1 x_e^p corresponde a una variable binaria que tomara un valor de 1 si el arco es recorrido prestando el servicio por el camión p, mientras y_e^p corresponde a una variable binaria que tomara un valor de 1 si el arco es recorrido sin prestar el servicio por el camión p

Adicionalmente para la construcción de la heurística se tomaron en cuenta las condiciones de visitar todos los arcos y respetar la capacidad de los vehículos, las cuales se representan en las ecuaciones 2 y 3 respectivamente.

$$\sum_{p\in\mathcal{I}} x_e^p = 1 \qquad \forall e \in \mathcal{E} \tag{2}$$

$$\sum_{e\in\mathcal{E}} d_e x_e^p \leq Q \qquad \forall p \in \mathcal{I} \tag{3}$$

La condición de mantener la conectividad no se presenta, dado que es se aseguro a partir de la construcción de la matriz de distancias y rutas más cortas entre todos los nodos de la red.

2 Metodología

2.1 Algoritmo

El algoritmo aplicado para solucionar el CARP se construyó en Python, apoyado por las librerias ***pandas*** y ***networkx***. El algoritmo construido se baso en el método del vecino más cercano, los pasos del heurístico son los siguientes:

1. Lectura de parámetros. En este paso se leen y cargan los parámetros de demanda en el arco, costo de atravesar el arco, capacidad del vehículo y número de vehículos.
2. Construcción de la matriz de distancias mínimas entre todos los nodos de la red.
3. Para cada uno de los arcos a visitar se crea el arco inverso con la misma demanda y el mismo costo. Por ejemplo. Si se debe visitar el arco (1,2) se crea el arco (2,1) con la misma demanda y el mismo costo.
4. Construcción de una matriz de distancias entre todo el conjunto de arcos. La matriz de distancias corresponde a la distancia desde el nodo final de un arco y el nodo de inicio del siguiente.
5. Creación de rutas:
   - 5.1 Identificar el arco más cercano al origen y agregarlo al camión disponible. Disminuir la capacidad del camión.
   - 5.2 Identificar el arco más cercano al último arco agregado y rutearlo. Si el camión dispone de capacidad.
   - 5.3 Eliminar el arco y su inverso de la lista de candidatos a visitar.
   - 5.4 Si el camión tiene capacidad y el último nodo de destino no es el origen, volver al paso 5.2, si no terminar la ruta en el origen.
   - 5.5 Si la lista de arcos candidatos es vacía terminar el ruteo, si no volver al paso 5.1, reiniciando la capacidad
6. Determinar la función objetivo del conjunto de rutas obtenidas en el paso 5.

Instancia	id	nodes	edges	Capacity	FO heurística	FO óptimo	Tiempo (s)	GAP
gdb1	1	12	22	5	329	316	0.255135	4.1%
gdb2	2	12	26	6	365	339	0.332587	7.7%
gdb3	3	12	22	5	324	275	0.326966	17.8%
gdb4	4	11	19	4	337	287	0.330172	17.4%
gdb5	5	13	26	6	445	377	0.415884	18.0%
gdb6	6	12	22	5	344	298	0.333694	15.4%
gdb7	7	12	22	5	332	325	0.405814	2.2%

Table 1: Resultados para siete instancias del CARP

3 Resultados

El heurístico construido fue evaluado sobre siete (7) instancias de literatura [5]. Las instancias fueron obtenidas de ***https://www.uv.es/belengue/carp.html***, de las cuales Longo han reportado los valores óptimos [7]. Estos valores fueron obtenidos a partir de transformaciones al modelo lineal para solucionarse como un problema de ruteo de vehículos. Así, con estos valores es posible desarrollar la evaluación de la eficiencia del método. Los experimentos computacionales para la aplicación de la heurística se realizaron en una CPU Intel(R) Core(TM) i7-10610U a 1,80GHz 2,30 GHz (con 16GB de RAM) utilizando Python 3.9.7. Un ejemplo de respuesta arrojada por el algoritmo se presenta en 1, en el cual se encontraron 5 rutas con costos de 98, 73, 64, 78 y 47. El ejemplo corresponde a una solución para el problema $gdb1$

```
[[[(1, 2), (2, 3), (3, 4), (4, 1), (1, 7)], 98],
[[(1, 10), (10, 11), (11, 5), (5, 6), (6, 7)], 73],
[[(1, 12), (12, 5), (5, 3), (6, 12), (12, 7)], 64],
[[(2, 4), (2, 9), (9, 10), (10, 8), (8, 11)], 78],
[[(7, 8), (11, 9)], 47]]
```

Figure 1: Conjunto de rutas obtenidas por el heurístico para la instancia gdb1

En la Tabla 1 se presentan los valores obtenidos para la heurística constructiva, los valores óptimos y el GAP de la solución y tiempos de computo.

Para contrastar el desempeño de la heurística se aplicó el algoritmo sobre 20 instancias. Se excluyeron las instancias 15, 17 y 21, dado que se han reportado con errores. Así, se analizó su desempeño del algoritmo para encontrar buenas soluciones y el tiempo de computo en función del tamaño de la red del problema . Este comportamiento se puede observar en la Figura 2

4 Discusión

El enfoque del vecino cercano es uno de los métodos que son utilizados con frecuencia en un contexto de enrutamiento de nodos. Sin embargo en la heurística propuesta en este paper se utilizó para el enrutamiento sobre arcos. El método construido es bastante sencillo de aplicar y genera buenos resultados en las instancias presentadas. La aplicación del modelo se pudo contrastar contra los óptimos obtenidos, de esta manera es evaluó la eficiencia del mismo, encontrando una media de desviación respecto al óptimo de 11.8% y con una desviación del 6.9%. La distribución del GAP se puede observar con más claridad en la Figura 3. De igual manera, se observa la variación en los tiempos de computo.

En cuanto a la eficiencia de la heurística, no se observa un impacto en la respuesta según el tamaño del problema. En la figura 2 se observa que no existe una gran relación, dado que tiene buenas aproximaciones tanto con problemas pequeños como en los grandes de las instancias evaluadas. De esta manera,

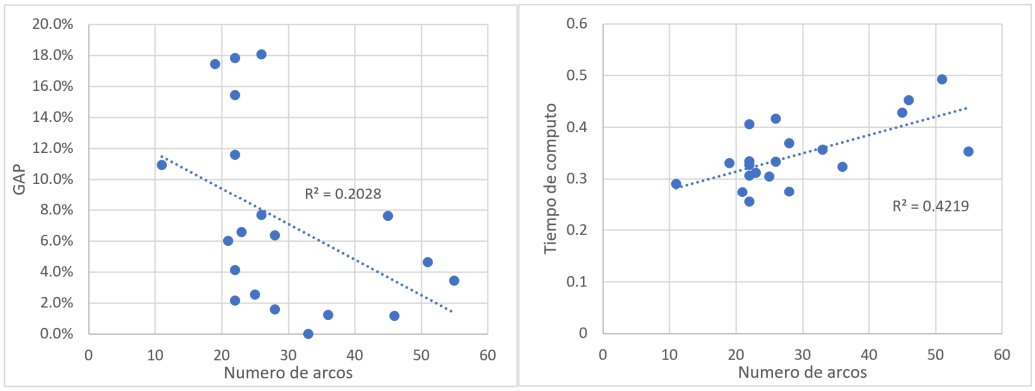

Figure 2: Desempeño y el tiempo de computo de la heurística en función del tamaño del problema

la relación entre la eficiencia medida por el GAP del modelo y el tamaño de la red solo presentan una correlación de 0.202.

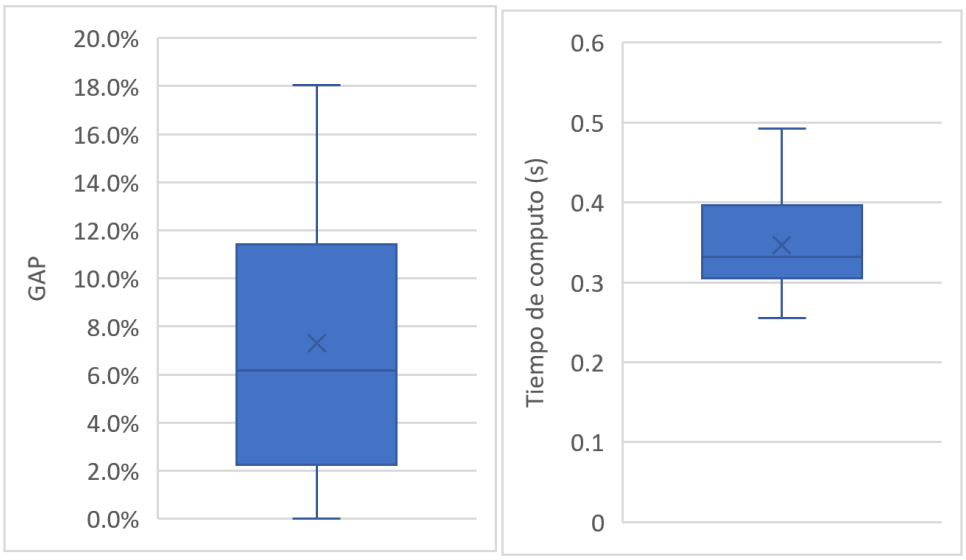

Figure 3: Distribución del GAP de las soluciones encontradas para las 20 instancias

En cuanto al tiempo de computo, la heurística utilizada requiere una cantidad mínima de tiempo de cálculo para encontrar la solución en cada una de las 20 instancias. El tiempo promedio para resolver una instancia fue de 0.34 segundos. Como se esperaba el tiempo de computo si es dependiente del tamaño del problema, como se puede observar una en la Figura 2

5 Conclusiones

En este artículo se estudio el problema de enrutamiento de arcos capacitados (DCARP), en el que un conjunto de vehículos debe visitar un conjunto de arcos sobre una red. En primer lugar, se tomaron las bases de literatura y de formulaciones matemáticas para identificar la estructura del problema. Hemos sugerido una estructura con un método heurístico constructivo bastante sencillo basado en el método del vecino más cercano. El método toma de base la identificación de la matriz de distancias más cortas entre todos los nodos y una matriz de distancias más cortas entre arcos. Los resultados mostraron que el algoritmo se comporta bastante bien en los instancias utilizadas, con una desviación respecto al óptimo de

11.8% y adicionalmente se resuelven con tiempos de computo inferiores a 1 segundo. En las instancias evaluadas se identificó que la heurística no es dependiente del tamaño del problema para encontrar buenas soluciones, de igual manera la capacidad de los vehículos no afecta en estos casos. Por último, se evaluó la dispersión en las respuestas la cual se encontró 6.9 %, incluso en una instancia la heurística logró encontrar la respuesta óptima para el problema.

References

[1] E Babaee Tirkolaee, M Alinaghian, M Bakhshi Sasi, and MM Seyyed Esfahani. Solving a robust capacitated arc routing problem using a hybrid simulated annealing algorithm: a waste collection application. *Journal of Industrial Engineering and Management Studies*, 3(1):61–76, 2016.

[2] Feng Chu, Nacima Labadi, and Christian Prins. The periodic capacitated arc routing problem linear programming model, metaheuristic and lower bounds. *Journal of Systems Science and Systems Engineering*, 13(4):423–435, 2004.

[3] Ángel Corberán, Richard Eglese, Geir Hasle, Isaac Plana, and José María Sanchis. Arc routing problems: A review of the past, present, and future. *Networks*, 77(1):88–115, 2021.

[4] Moshe Dror. *Arc routing: theory, solutions and applications*. Springer Science & Business Media, 2012.

[5] Bruce L Golden and Richard T Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[6] Amin Khajepour, Majid Sheikhmohammady, and Ehsan Nikbakhsh. Field path planning using capacitated arc routing problem. *Computers and Electronics in Agriculture*, 173:105401, 2020.

[7] Humberto Longo, Marcus Poggi De Aragao, and Eduardo Uchoa. Solving capacitated arc routing problems using a transformation to the cvrp. *Computers & Operations Research*, 33(6):1823–1837, 2006.