Solución del problema del recojo de basura – Capacited Arc Routing Problem (CARP)

Luis E. Tarazona[1]

[1] Universidad de los Andes
Carrera 1 Este # 19 A – 40, Bogotá D.C., Colombia
le.tarazona@uniandes.edu.co

Resumen

En este documento se considera un algoritmo constructivo heurístico para resolver el CARP. cuyo objetivo es buscar las rutas más cortas que permitan recoger toda la demanda existente, consiguiendo la minimización del costo de transporte. Se discutirá el rendimiento del algoritmo considerando el tiempo computacional de solución y se comparará los resultados obtenidos de costo respecto a los obtenidos en otras investigaciones.

1 Introducción

El problema del recojo de basura en la vida real es un problema que afecta cada vez más a la población en diversas partes de Colombia y del mundo, esto debido a una ineficiente gestión en el servicio de recojo de residuos sólidos por parte de los gobiernos actuales, que no planifica y programa debidamente las unidades que dispone para realizar esta actividad, además, en los últimos años la generación de basura per cápita solamente en Bogotá para el 2018 fue de 515 kg, cifra importante que muestra un crecimiento por año del 4% (DANE, 2020), lo que para el 2022 se estaría estimando una generación per cápita de 602 kg, esto muestra que la capacidad vehicular para recoger basura se vería afectado y por tanto se evidenciará basura con mayor tiempo de permanencia en las calles, lo que generará a su vez inconformidad en la población.

La programación de los vehículos encargados del recojo de basura es de vital importancia, ya que se debe aprovechar recoger a su máximo de capacidad para cubrir toda la demanda de basura y al menor costo posible. Estudios anteriores han realizado investigaciones de este tipo de problema de recojo de basura como un problema denominado "*Capacited Arc Routing Problem* (CARP)", el cual consiste en demandas colocadas en las aristas, y cada arista debe satisfacer la demanda, así hacen mención (Gendreau & Laporte, 1994) donde ponen el ejemplo al recojo de basura, donde cada ruta puede requerir tanto una recolección de basura como una recolección reciclable. Pueden surgir problemas en las aplicaciones de la vida real si hay problemas de tiempo, como el caso en el que ciertas rutas no pueden ser atendidas debido a conflictos de tiempo o programación, o restricciones, como un periodo de tiempo limitado.

(Golden & Wong, Capacitated Arc Routing Problems, 1981) realizaron una investigación donde evaluaron una heurística basándose en el problema del cartero chino capacitado (CCPP) planteado por (Christofides, 1973) además (Golden B. , 1978) lo formula como un problema de cobertura de conjuntos. (Clarke & Wroght, 1964) realizaron una investigación donde realizaron la programación de vehículos desde un depósito hasta puntos de entrega y retorno. Estudios más recientes cómo el de (Bautista & Pereira, 2003) que realizaron un estudio comparativo entre dos fases constructivas de las metaheurísticas ACO y GRASP para el diseño de itinerarios con servicio asociado a los arcos y restricciones de capacidad de los vehículo, considerando además el algoritmo del vecino más cercano, (Chen, Hao, & Glover, 2016) que realizaron una metaheurística híbrida para poder solucionar este problema que incorpora un procedimiento de refinamiento local efectivo, acoplando un procedimiento de umbral tabú aleatorio con un procedimiento de descenso inviable, en el marco memético. (García, 2017) en su investigación modeló un problema de ruteo de vehículos por arcos capacitados también tomando en cuenta el algoritmo de vecinos más cercanos.

Es por eso que el presente trabajo presenta una solución heurística constructiva aplicando la lógica del

algoritmo del vecino más cercano para poder dar solución al CARP, en el siguiente apartado se muestra la descripción del problema CARP, en el apartado 3 se muestra el proceso heurística planteado para este trabajo, en el apartado cuarto se muestra los resultados obtenidos y comparados con los resultados de la aplicación del modelo matemático que obtuvieron (Golden & Wong, 1981) con sus respectivas instancias, por lo que se usará las mismas instancias. Finalmente se muestra las conclusiones del trabajo.

2 Descripción del problema

El problema consiste en recoger una cantidad definida de basura "q_{ij}" de un grupo de clientes con un grupo de vehículos p con capacidad definida "W". Los vehículos deben de recoger toda la cantidad de basura sin sobrepasar la capacidad que puede llevar por viaje y al menor costo posible "c_{ij}". Los vehículos salen de un depósito y siempre deben regresar cuando completan su capacidad y/o cuando culminaron de recoger toda la demanda.

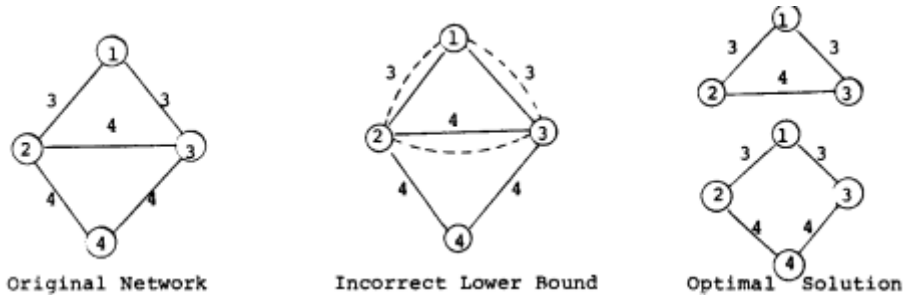

Figura 1: Ejemplo de ruteo para el recojo de basura

Como se puede ver en la Figura 1, los vehículos salen del nodo 1, recorren todos los arcos y retornos siempre al nodo 1. Para un mejor entendimiento, a continuación, se presenta la formulación matemática descrita en (Golden & Wong, 1981) para el problema estudio de este trabajo:

$$minimizar \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{p=1}^{K} c_{ij}x_{ij}^{p} \tag{1}$$

$$Sujeto\ a: \sum_{k=1}^{n} x_{ki}^{p} - \sum_{k=1}^{n} x_{ik}^{p} = 0 \quad para\ i = 1,\dots,n \quad p = 1,\dots,K \tag{2}$$

$$\sum_{p=1}^{K}\left(l_{ij}^{p} + l_{ji}^{p}\right) = \left[\frac{q_{ij}}{W}\right]\ para\ (i,j) \in E \tag{3}$$

$$x_{ij}^{p} \geq l_{ij}^{p} \quad para\ (i,j) \in E \quad p = 1,\dots,K \tag{4}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} l_{ij}^{p}\, q_{ij} \leq W \ \ para\ p = 1,\dots,K \tag{5}$$

$$\sum_{i\in\tilde{Q}}\sum_{i\in\tilde{Q}} x_{ij}^{p} - n^{2}\, y_{1\tilde{q}}^{p} \leq \left|\tilde{Q}\right| - 1 \tag{6}$$

$$\sum_{i\in\tilde{Q}}\sum_{i\in\tilde{Q}} x_{ij}^{p} + y_{2\tilde{q}}^{p} \geq 1$$

$$y_{1\tilde{q}}^{p} + y_{2\tilde{q}}^{p} \leq 1; y_{1\tilde{q}}^{p}, y_{2\tilde{q}}^{p} \in \{0,1\}$$

$$x_{ij}^{p}, l_{ij}^{p} \in \{0,1\}$$

$$for\ p = 1,\dots,k \quad \tilde{q} = 1,\dots,2^{n-1}-1 \quad y\ todo\ no\ vacío \quad subconjunto\ \tilde{Q}\ de\ \{2,3,\dots,n\} \tag{7}$$

Donde n es el número de nodos, K es el número de vehículos, q_{ij} la demanda del arco (i,j), W es la capacidad del vehículo ($W \geq maxq_{ij}$), c_{ij} $es\ la\ longitud\ del\ arco$ (i,j), $x_{ij}^{p} = 1$ si el arco (i,j) es atravesado por el vehículo p, es igual a 0 si es de otra manera, $l_{ij}^{p} = 1$ si ya realizó el servicio en el arco (i,j), 0 de otra manera, $[z]$ es un número entero más pequeño o mayor a z.

La ecuación (1) muestra la función objetivo, la cual es minimizar la distancia total recorrida. La ecuación (2) garantiza la continuidad de la ruta, la ecuación (3) indica que cada arco con demanda positiva es atendido exactamente una vez, la ecuación (4) garantiza que el arco (i,j) puede ser atendido por el vehículo p si solo cubre el arco (i,j). La capacidad del vehículo no es vulnerada por la ecuación (5). La ecuación (6) prohíbe la formación ilegal de subrutas. Se señala también que cada índice $\tilde{q}$ corresponde a un conjunto $\tilde{Q}$. Las restricciones de integralidad se dan en la ecuación (7).

3 Procedimiento heurístico

Para realizar la solución de este problema se planteó una heurística constructiva tomando en cuenta el algoritmo del vecino más cercano, para ello se desarrolló el siguiente procedimiento verbal antes de desarrollar el algoritmo de forma computacional considerando las primeras 7 instancias de (Golden, DeArmon, & Baker, 1983) utilizadas también por (Belenguer & Benavent, 2003):

1) Del nodo depósito (nodo 1) sale un vehículo y debe dirigirse al siguiente nodo que sea el menos costoso para transportarse (línea 0 a línea 21).
2) Se asume el costo de ese arco que conecta ambos nodos y se recoge el valor de la demanda igual a 1 (línea 22 a línea 32).
3) Del nodo seleccionado se vuelve aplicar la misma lógica, que el vehículo vaya al nodo que sea el menos costoso para transportarse, en este caso se debe tener en cuenta que, si la capacidad del vehículo aún no ha sido alcanzada, el vehículo no debe retornar al nodo depósito (línea 33 a línea 44).
4) Se asume el costo de ese arco que conecta ambos nodos y se recoge el valor de la demanda igual a 1.
5) Cuando el vehículo haya alcanzado su máxima capacidad, debe regresar al nodo depósito buscando la distancia menos costosa desde el nodo localizado, sin recoger más demandas y acumulando costo de transporte por el arco o los arcos que deberá pasar para volver al nodo depósito (línea 44 a línea 45).
6) Este procedimiento debe repetirse hasta que se haya recogido la demanda de todos los arcos. La cantidad de repeticiones realizadas será el total de veces que sale un solo vehículo o en su defecto la cantidad de vehículos que se necesita para satisfacer la demanda, esto puede denominarse como rutas de recojo (línea 46 a línea 48).

Se desarrolló un código que presenta 3 partes importantes, la primera parte es el código para la obtención de la información de las instancias, dada la forma en cómo se encontraba distribuidos los datos se tuvo que hacer unos ajustes a los documentos antes de poder llamar la información. la segunda parte la determinación de las rutas más cortas aplicando disjktra obteniendo valores de costos para cada ruta, y la tercera parte la heurística constructiva.

A continuación, se presenta el siguiente pseudocódigo planteado:

1. Input → (nodos, arcos, demanda, capacidad, costos, número de nodos, número de arcos)
2. Determinar el grafo "g" teniendo en cuenta número de nodos, arcos, atributos de arco = "weight" equivalente a costos
3. Aplicar disjktra:
4. Crear lista de arcos pendientes

```
5. Crear lista de nodos pendientes
6. Mientras longitud(arcos pendiente) > 0:
7.     Mientras no esté la Ruta_lista:
8.         Ruta_lista = True
9.         Para nodo candidato en lista de nodos pendientes:
10.            Si existe ruta = False
11.            Para arco en lista de arcos pendientes:
12.                Si arco[0] = candidato o arco[1] = candidato y demanda recogida + demanda
                   del arco <= Capacidad:
13.                    Si existe ruta = True
14.            Si Distancia[Nodo_inicio,Nodo_final] < Distancias y si existe ruta:
15.                Distancia recorrida = Distancia[Nodo_inicio,Nodo_final]
16.                Lista de costos[Número de rutas] = Distancia[Nodo_inicio,Nodo_final]
17.                Nodo_final = candidato
18.                Ruta_lista = False
19.        Si no existe Ruta_lista:
20.            Para cada arco en lista de arcos pendientes:
21.                Si arco[0] = Nodo_final o arco[1] = Nodo_final y demanda de arco + demanda
                   recogida <= 0:
22.                    Si costos[arco] < mejor_costo:
23.                        Mejor arco = arco
24.                        Mejor costo = costos[arco]
25.            Si Nodo_final es distinto de Nodo_inicio:
26.                Nodo definitivo = Mejor arco[1]
27.            Si no:
28.                Nodo definitivo = Mejor arco[0]
29.            Agregar Nodo definitivo a Rutas[Número de rutas]
30.            Agregar las Distancias del mejor arco a Lista de costos
31.            Agregar Mejor arco a Barrido de arcos
32.            Remover mejor arco de lista de arcos pendientes
33.            Si existe aún arcos pendientes = False
34.            Para arco en listas de arcos pendientes:
35.                Si arco[0] = Nodo_final o arco[1] = Nodo_final:
36.                    Si existe aún arcos pendientes = True
37.                Si no existe aún arcos pendientes:
38.                    Remover Nodo_final de lista de arcos pendientes:
39.            Para arco en listas de arcos pendientes:
40.                Si arco[0] = Nodo definitivo o arco[1] = Nodo definitivo:
41.                    Si existe aún arcos pendientes = True
42.                Si no existe aún arcos pendientes:
43.                    Remover Nodo definitivo de lista de arcos pendientes:
44.                Agregar a demanda recogida la demanda del mejor arco
45.                El Nodo definitivo será el Nodo_inicio
46.    Cerrar Rutas[Número de rutas] retornando a Nodo_inicio (añadirlo a Rutas[Número de ru-
       tas]
47.    Cerrar lista de costos añadiendo el costo de la distancia más corta a Nodo_inicio
48.    Output → (Rutas, costos)
```

4 Resultados

En la Tabla 1 se muestra los resultados obtenidos para la aplicación de la heurística planteada. Se observa un tiempo computacional bastante pequeño por lo que en este ítem es importante la heurística planteada para obtener resultados inmediatos. El total de rutas indicadas es igual a la cantidad de vehículos indicados en las instancias. Los resultados de este trabajo comparados con el resultado FO que mencionan (Belenguer & Benavent, 2003), se obtienen valores de GAP ($Resultados - FO/FO$) menores de 30% para 6 de las 7 instancias, 1 de las instancias tiene un GAP superior a 40%.

Instancias	**Tiempo computacional**	**Rutas**	**Resultados**	**FO** (Belenguer & Benavent, 2003)	**GAP**
1	0.0153	1: [0, 11, 5, 6, 7, 9, 0] 2: [0, 1, 8, 10, 7, 9, 10, 9, 0] 3: [0, 3, 1, 2, 4, 5, 6, 0] 4: [0, 6, 11, 4, 10, 9, 8, 1 0] 5: [0, 9, 0, 3, 2, 1, 0]	394	316	24,68%
2	0.0157	1: [0, 11, 5, 6, 7, 9, 0] 2: [0, 8, 1, 3, 11, 4, 11, 0] 3: [0, 1, 2, 4, 5, 7, 6, 0] 4: [0, 3, 2, 4, 10, 7, 9, 6, 0] 5: [0, 6, 11, 0, 9, 10, 8, 0] 6: [0, 8, 9, 0]	377	339	11,21%
3	0.0156	1: [0, 11, 5, 6, 7, 9, 0] 2: [0, 8, 1, 3, 0, 1, 0] 3: [0, 6, 5, 4, 2, 11, 4, 11, 0] 4: [0, 9, 10, 7, 9, 8, 10, 8, 0] 5: [0, 3, 2, 3, 2, 4, 10, 8, 0]	389	275	41,45%
4	0.0155	1: [0, 5, 4, 2, 1, 8, 1, 0] 2: [0, 1, 3, 0, 6, 7, 6, 0] 3: [0, 9, 7, 10, 9, 8, 1, 0] 4: [0, 5, 6, 7, 10, 8, 1, 3, 2, 4, 10, 9, 0]	364	287	26,83%
5	0.0156	1: [0, 11, 5, 6, 7, 9, 0] 2: [0, 1, 8, 10, 7, 9, 10, 9, 0] 3: [0, 3, 12, 4, 2, 1, 0] 4: [0, 6, 11, 4, 5, 11, 12, 3, 0] 5: [0, 9, 8, 1, 3, 2, 12, 3, 0] 6: [0, 11, 4, 10, 9, 0]	458	377	21,49%
6	0.0160	1: [0, 11, 5, 6, 7, 9, 0] 2: [0, 1, 8, 9, 0, 3, 0] 3: [0, 6, 11, 4, 2, 1, 0] 4: [0, 11, 10, 3, 1, 3, 2, 10, 3, 0] 5: [0, 6, 5, 4, 10, 3, 0]	363	298	21,81%
7	0.0156	1: [0, 11, 4, 2, 1, 8, 1, 0] 2: [0, 1, 3, 0, 10, 5, 6, 0] 3: [0, 6, 5, 10, 7, 9, 10, 0] 4: [0, 9, 8, 10, 4, 2, 3, 0] 5: [0, 11, 6, 7, 6, 0]}	373	325	14,77%

Tabla 1: Resultados obtenidos y GAP

En general estos valores elevados pueden ser un detalle de no haber considerado dentro del planteamiento el total de vehículos. Otro detalle a considerar es el algoritmo del vecino más cercano, ya que quizá considerar hacer un recojo iniciando por los vecinos más lejanos podría reducir el costo de transporte que para el planteamiento de este trabajo fueron elevados comparando al óptimo.

5 Conclusiones

Se logró realizar una heurística constructiva para poder solucionar el problema de *Capacitated Arc Routing problema* que cumpla con dar resultados factibles en tiempos computacionales aceptables y que cumplan con un GAP menor al 30%.

Para mejorar los valores de GAP se sugiere realizar ajustes a la heurística constructiva, asignando tal vez los clientes que debe visitar cada vehículo, o ejecutar la visita a los clientes más distantes como nivel de prioridad para luego completar la recolección con los vecinos más cercanos.

Referencias

Bautista, J., & Pereira, J. (2003). Comparativa de las fases constructivas de las metaheurísticas ACO y GRASP para el problema CARP. *V Congreso de Ingeniería de Organización*.

Belenguer, J., & Benavent, E. (2003). *A Cutting Plane Algorithm for the Capacitated Arc Routing Problem.* Computers and Operations Research. Obtenido de https://www.uv.es/belengue/carp.html

Chen, Y., Hao, J.-K., & Glover, F. (2016). A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, 25-39.

Christofides, N. (1973). The optimum tranversal of a graph. *Omega*(1), 719-732.

Clarke, G., & Wroght, J. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations research*, 568-581.

DANE. (2020). *Cuenta ambiental y económica de flujo de materiales - residuos sólidos.* Obtenido de https://www.dane.gov.co/files/investigaciones/pib/ambientales/cuentas_ambientales/cuentas-residuos/Bt-Cuenta-residuos-2018p.pdf

García, N. (2017). *Métodos numéricos con python: Problema de ruteo de vehículos por arcos.* Coahuila: Universidad Autónoma de Coahuila.

Gendreau, M., & Laporte, G. (1994). *Arc routing problemas, part II: The rural postman problem.* Canadá: Survey, expository & tutorial. Obtenido de https://pubsonline.informs.org/doi/epdf/10.1287/opre.43.3.399

Golden, B. (1978). Recen developments in vehicle routing in computer and mathematical programming. *National Bureau of Standars Special Publication*, 233-240.

Golden, B., & Wong, R. (1981). *Capacitated Arc Routing Problems.* Networks.

Golden, B., DeArmon, J., & Baker, E. (1983). *Computational Experiments with Algorithms for a Class of Routing Problems.* Computers and Operations Research.