Método de solución problema CARP utilizando Dijkstra

Universidad de los Andes
Isabella Bermúdez Gutiérrez
201923469
i.bermudez@uniandes.edu.co

Resumen

Este documento pretende aplicar una nueva heurística al problema CARP basada en una regla de decisión que tiene como base el algoritmo Dijkstra. A lo largo del documento, se estudian algunas de las soluciones exploradas por distintos autores y se comparan los resultados con las mismas. Se analizan distintos puntos de mejora que se podrían implementar en futuras ocasiones para mejorar las soluciones de la heurística planteada.

I. INTRODUCCIÓN

Dadas las demandas en los arcos de una red que debe ser atendida por una flota de vehículos idénticos con capacidad limitada, la idea del problema a resolver consiste en encontrar las rutas que deben hacerse y juntas minimicen el costo total generado. Esto, teniendo en cuenta que las rutas siempre deben empezar y terminar en un nodo específico, denominado depósito. Por lo tanto, la pregunta a resolver es la siguiente: ¿Cómo atender estas demandas en el menor costo y sin agotar las capacidades de los vehículos? En términos generales, esta pregunta es precisamente abordada por el problema de enrutamiento de arco capacitado (CAPR) propuesto por Golden y Wong (1981). El CARP pertenece al conjunto de problemas NP-duros, por lo que se han desarrollado numerosos enfoques de solución heurísticos y metaheurísticos para resolverlo. En este artículo se propone una heurística constructiva, cuyo enfoque está basado principalmente en el método Dijkstra. En este sentido, la idea es ir construyendo rutas, según la ruta más corta (menos costosa) generada por Dijkstra, hasta que la capacidad del vehículo se agote. Ya con esto, desde del último nodo de dicha ruta hasta el depósito se busca la ruta con el algoritmo Dijkstra. Así, el objetivo de este informe será evaluar los resultados de la heurística planteada por medio del análisis de resultados alcanzados, junto con la comparación de estos con los ya alcanzados por otros autores.

II. DESCRIPCIÓN DEL PROBLEMA

Formalmente, el problema CARP puede definirse de la siguiente manera. Sea $G(V, E)$ un grafo no dirigido con un costo o longitud no negativa $c(v_i, v_j)$ y una demanda no negativa $d(v_i, v_j)$ asignada a cada arista $(v_i, v_j) \in E$. Sea $E_R \subseteq E$ el conjunto de aristas con demanda positiva, llamadas aristas requeridas. Se utiliza una flota de vehículos idénticos con capacidad limitada D para dar servicio a todos los bordes requeridos. Mientras atraviesa un borde (v_i, v_j), un vehículo podría: (1) prestar servicio a (v_i, v_j), lo que reduce su capacidad en $d(v_i, v_j)$ y aumenta el costo de la solución en $c(v_i, v_j)$ o (2) solo pasar por (v_i, v_j), que únicamente aumenta el costo de la solución en $c(v_i, v_j)$. Sea $V_R \subseteq V$ el conjunto de nodos (vértices) con alguna conexión a una arista $(v_i, v_j) \in E_R$.

En la literatura para la resolución del CARP se han presentado múltiples heurísticas constructivas que han alcanzado diversas soluciones. Estas permanecen importantes en esta área de investigación, ya que se caracterizan por su bajo tiempo de procesamiento y por su facilidad para adaptarse a diferentes instancias del problema.

La primera heurística constructiva fue propuesta por Golden y Wong (1981). Esta heurística comienza con el vehículo en el depósito y luego presta servicio progresivamente al arco sin servicio más cercano. El vehículo se detiene y regresa al depósito cuando la capacidad restante del vehículo no es suficiente para dar servicio a más arcos. Luego, se emplea un nuevo vehículo para dar servicio a los arcos restantes y se repite el proceso. En este caso, la heurística emplea cinco criterios diferentes para decidir qué borde no atendido $(v_i, v_j) \in E_R$ se atenderá a continuación: 1) minimizar $\frac{c(v_i,v_j)}{d(v_i,v_j)}$; (2) maximizar $\frac{c(v_i,v_j)}{d(v_i,v_j)}$; (3) minimizar el costo del nodo v_j de regreso al nodo de depósito; (4) maximizar el costo del nodo v_j de regreso al nodo de depósito; (5) utilizar el criterio 4 si el vehículo tiene más de la mitad de su capacidad; de lo contrario, utilice el criterio 3. Una instancia de problema se resuelve cinco veces, usando un criterio de selección diferente cada vez. La mejor de las cinco soluciones generadas es la solución final.

Luego de esto, Pearn (1989) propuso una versión modificada mediante la selección aleatoria de uno de los cinco criterios originales como desempate cada vez que se identifican múltiples bordes sin servicio con una distancia mínima. Una de las principales ventajas de esta heurística es que produce mucho más de cinco soluciones diferentes debido a su comportamiento estocástico. Comparando resultados, esta metodología obtuvo un mejor rendimiento que el escaneo de ruta original para todas las instancias experimentadas.

Posteriormente, Belenguer y Benavent (2003) propusieron otra versión modificada de exploración de rutas (PS-RE), la cual simplifica la decisión de desempate simplemente seleccionando aleatoriamente una de las aristas atadas. Al comparar los resultados, se identificó que obtuvo soluciones muy parecidas a las encontradas con el algoritmo de Pearn (1989).

Finalmente, Santos, et al. (2009) propusieron la regla de exploración de caminos con elipse (PS-Elipse) que funciona de manera similar a PS-RE, pero también considera una "regla de elipse". Esta regla se activa cuando la capacidad restante del vehículo es lo suficientemente baja y restringe el servicio del vehículo solo a un subconjunto de bordes que se encuentran dentro de una elipse. Aunque con respecto a las anteriores este algoritmo aumentó un poco su tiempo de procesamiento, este presentó una clara mejora en sus resultados.

III. METODOLOGÍA

El método propuesto de solución está basado en la estructura de las primeras heurísticas constructivas descritas anteriormente y en la utilización del algoritmo Dijkstra.

Sea $aristas\_req = |E_R|$ el número de aristas requeridas, $demanda\_total = \sum_{(v_i,v_j)\in E_r} d(v_i, v_j)$ la demanda total, $costo\_total = \sum_{(v_i,v_j)\in E} c(v_i, v_j)$ el costo total de las aristas, (v_g, v_h) el último borde atendido en la ruta, $Dijkstra(v_p, v_q, 1)$ el camino más corto desde el nodo v_p al nodo v_q, $Dijkstra(v_p, v_q, 2)$ el costo del camino más corto desde el nodo v_p al nodo v_q, cap_rest la capacidad

restante del vehículo y v_o el nodo de depósito. Dadas estas definiciones, la heurística propuesta se ilustra en el Algoritmo 1. La heurística comienza con un vehículo vacío en el depósito (líneas 2 y 3) e inserta iterativamente en la ruta el borde más cercano sin servicio (v_i, v_j) que no viola la cap_rest. Así, la solución es construida añadiendo una arista a la vez. La decisión de qué arista se va a servir se basa en el algoritmo Dijkstra, el cual minimiza el costo de un nodo a otro. Así, en cada iteración se recorren todas las aristas $(v_i, v_j) \in E_R$, se escoge la más cercana (menos costosa), se añade a la ruta y se repite el proceso hasta que $cap_{rest} = 0$ (línea 5).

Con esto en mente, el vehículo se detiene cuando la capacidad restante de este no es suficiente para dar servicio al siguiente arco requerido, de manera que allí se define el último arco atendido (v_g, v_h). En este punto, el algoritmo procede a utilizar Dijkstra (línea 6), de manera que se encuentre la ruta más corta entre el último nodo de la ruta ya marcada v_h y el depósito v_0. Por lo tanto, se hallará la ruta $Dijkstra(v_h, v_0, 1)$ y al costo total de la ruta se le sumará $Dijkstra(v_h, v_0, 2)$ (línea 7). A medida que se suple la demanda de la arista (v_i, v_j), esta sale del conjunto E_R (línea 8). En este sentido, una vez alcanzada la capacidad del vehículo, aunque este siga recorriendo aristas requeridas para devolverse al depósito (arrojadas por el algoritmo Dijkstra), este no prestará servicio, por lo que estas seguirán siendo parte del conjunto E_R.

Ahora bien, el algoritmo está sujeto a un máximo de rutas (cantidad de vehículos disponibles) y a que todas las aristas requeridas sean servidas por algún vehículo ($|E_R| = 0$), de manera que se asegure que ningún arco quede sin atender. Estas condiciones son las que iteran la construcción de cada ruta por vehículo (línea 1).

Algoritmo 1

(1) while (rutas < vehículos and $|E_R| > 0$):
(2) Scur = [v_0]
(3) cap_{rest} = D
(4) cabeza = v_0
(5) while (cap_rest > 0):
   1. Se recorren todas las aristas requeridas, evaluando cada nodo $v \in V_R$. Se escoge v_h que represente el menor costo $c(cabeza, v_h)$.
   2. Ya seleccionado, se actualiza la $d(cabeza, v_h) = 0$, $cap_{rest} -= d(cabeza, v_h)$, el $costo\ total\ +=\ c(cabeza, v_h)$, se actualiza qué nodo es la cabeza de la ruta y se vuelve al paso 1 si la $cap_{rest} > 0$.

   End while
(6) Encontrar ruta más corta $Dijkstra\ (v_h, depósito)$.
(7) Agregar al costo total según la ruta encontrada con Dijkstra.
(8) Recalcular $|E_R|$ según lo realizado por el algoritmo.

End while

IV. EXPERIMENTOS COMPUTACIONALES Y ANÁLISIS DE RESULTADOS

La implementación de esta nueva heurística se realizó en el lenguaje de programación Python versión 4.1.5. Para la evaluación de esta se utilizaron las instancias propuestas por Golden y Wong (1981).

Aunque los resultados alcanzados claramente sobrepasan las cotas inferiores que se han encontrado en la literatura, esta nueva heurística propone soluciones factibles y cumple con todas las limitaciones y restricciones del problema. Para cada una se logra suplir todas las aristas con los vehículos disponibles sin sobrepasar las capacidades de estos.

Ahora bien, tomando como punto de referencia las soluciones encontradas por Belenguer y Benavent (2003), las cuales hasta ahora representan la mejor solución encontrada, en la Tabla 1 se calcularon los gaps (%) para cada una de las instancias evaluadas, así como el tiempo de cómputo en segundos del algoritmo propuesto.

Tabla 1. Instancias Golden y Wong (1981)

	Belenguer y Benavent (2003)	**Heurística evaluada**	**Gap (%)**	**Tiempo CPU**
gdb1	316	384	21.52%	0.000
gdb2	339	419	23.60%	0.002
gdb3	275	372	35.27%	0.000
gdb4	287	348	21.25%	0.002
gdb5	377	475	25.99%	0.003
gdb6	298	387	29.87%	0.004
gdb7	325	365	12.31%	0.002

Analizando cada una las instancias, es claro que en promedio se obtuvo un gap de 24.26% con respecto a las mejores soluciones encontradas por Belenguer y Benavent (2003). Por este motivo, es claro que la heurística tiene muchos puntos de mejor que deben ser revisados e implementados en futuras ocasiones.

Con los resultados encontrados, es claro que la solución podría mejorar. Por su parte, toda la literatura sobre el problema ha encontrado progresivamente mejoras que ayudan a definir rutas que implican un menor costo. Los detalles que son prioritarios para definir esta mejora pueden reducirse al criterio utilizado para definir qué arco escoger a la hora de estar generando las rutas. En este caso, siempre se utilizó Dijkstra como criterio, es decir, se elegía qué arista recorrer de acuerdo a si su costo era el mínimo desde donde el camión estaba parado. Sin embargo, esto en algunas ocasiones genera que el camión se aleje mucho del depósito, aun cuando este ya tiene muy poca capacidad para seguir recogiendo basura. Por lo tanto, bajo esta regla de decisión lo que el camión va a recoger en esas últimas aristas realmente no es tan significativo en comparación con el costo que se asume por ir hasta allá. Esto genera que el costo de la función objetivo aumente innecesariamente, pues se generan unas rutas muy largas para suplir pocas aristas.

En este sentido, la heurística presentada podría mejorar en cuanto al planteamiento de su regla de decisión. No solo basta con evaluar qué aristas se deben recorrer según Dijkstra, sino que es necesario plantear otras condiciones que permitan que la relación demanda/costo se maximice. En este sentido, se podría imponer una regla, la cual establezca que cuando el vehículo alcance un determinado nivel de carga, se active una regla de eficiencia que restrinja el servicio del vehículo solo a las aristas consideradas "rentables". Asimismo, otra regla para tener en cuenta es cuando hay un empate entre dos aristas. En esta heurística se escoge la primera que el algoritmo evalúa, sin embargo, los resultados de otros autores, que han sido mejores, tienen esta regla de escogencia para que sea aleatorio entre las candidatas.

V. CONCLUSIONES

CARP ha sido uno de los problemas más estudiados por la literatura. Este documento presentó un nuevo enfoque basado en la selección de rutas por medio del criterio del algoritmo de Dijkstra. Los resultados de esta no superaron los de los métodos ya estudiados por otros autores, pero alcanzaron resultados factibles, acordes a las restricciones del problema y cercanos a los óptimos ya encontrados. En promedio, presentaron un gap de solo el 24.26%.

Al analizar los resultados, fue posible encontrar varias áreas de mejora para la heurística planteada. Principalmente, se encontró que la regla de decisión debe tener en cuenta la eficiencia que se tiene, es decir, debe escoger aquellas aristas que cumplan con cierto nivel de "rentabilidad". Asimismo, otro posible punto de mejora se reduce a la regla de desempate cuando hay más de una arista que representa el mismo costo para la ruta. Comparando con las soluciones de otros autores, esta podría mejorar si el algoritmo no solo escoge el primero que evalúa, sino que lo escoge totalmente aleatorio.

REFERENCIAS

Arakaki, R. K., & Usberti, F. L. (2019). An efficiency-based path-scanning heuristic for the capacitated arc routing problem. *Computers & Operations Research*, *103*, 288-295.

Belenguer, J. M., & Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, *30*(5), 705-728.

Golden, B. L., & Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, *11*(3), 305-315.

Pearn, W. L. (1989). Approximate solutions for the capacitated arc routing problem. *Computers & Operations Research*, *16*(6), 589-600.

Santos, L., Coutinho-Rodrigues, J., & Current, J. R. (2009). An improved heuristic for the capacitated arc routing problem. *Computers & Operations Research*, *36*(9), 2632-2637.