Inteligencia Artificial

Informe Final: Nombre Proyecto

Ignacio Rojas Vega

11 de enero de 2021

Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

El *Green-Vehicle Routing Problem* (o G-VRP por su sigla) busca poder encontrar las mejores rutas para vehículos que salen de un mismo almacén, los cuales realizan entregas hacia un número de clientes/consumidores y, finalmente, retornan al mismo almacén de origen. Para ello, el problema considera ciertos factores a la hora de encontrar las mejores rutas, como, por ejemplo, el consumo de combustible de los vehículos y lugares estratégicos de abastecimiento de combustible para que dichos vehículos puedan recargarlo. En el presente documento, se realizará su estado del arte abarcando su origen y revisando los estudios que se han realizado para describirlo y encontrar su solución. También se estudiarán ciertos algoritmos y heurísticas para la búsqueda de soluciones, junto con los modelos matemáticos que tienen asociados.

1. Introducción

El *Green-Vehicle Routing Problem* (G-VRP) es una extensión del problema *Vehicle Routing Problem* (VRP) con el cual se pretende realizar una serie de rutas de repartición de productos, utilizando vehículos de **combustible alternativo** [9]. En este tipo de problemas, lo que se intenta es buscar y/o detectar rutas para que una flota de vehículos homogénea (es decir, vehículos iguales/idénticos) comience y termine el recorrido en un almacén y logre satisfacer las demandas de clientes, reduciendo los costos de combustible al minimizar la distancia total de viajes realizados.

Los autores propusieron el problema para hacer más viable el uso de combustibles ecológicos, debido el limitado número de estaciones que operan este tipo de combustible. De esta forma, se piensa obtener recorridos óptimos considerando la necesidad de parar en estos puntos de recarga antes de quedar varados por la falta de combustible. Específicamente se refieren a los camiones,

el cual corresponde al tipo de vehículo que más ha aumentado durante los últimos años. Por lo tanto, al tener la viabilidad de cambiar a un combustible más amigable con el medio ambiente, se estará limitando la contaminación por dióxido de carbono.

Este documento tiene por propósito entregar una descripción del problema junto con su importancia. Además, desarrollar una técnica para resolver las instancias del problema y analizar los resultados.

La estructura del presente documento es la siguiente: en la Sección 2 se explicará el G-VRP de forma general, junto con los parámetros y restricciones que lo componen. En la Sección 3 se hará un recorrido por el estado del arte, estudiando tanto los orígenes de la definición del problema como los primeros métodos diseñados para darle solución. Se revisará una gama de técnicas que faciliten el objetivo de encontrar soluciones factibles para el problema. En la Sección 4 se revisarán algunos modelos matemáticos que se han planteado para definir formalmente el problema, y finalmente en la Sección 5 se expondrán las conclusiones sobre el trabajo realizado.

2. Definición del Problema

G-VRP tiene como objetivos el minimizar la distancia que se recorre en cada tours, el consumo de combustible o, en efecto, la utilización de vehículos propulsados por combustibles alternativos (*Alternative-Fuel Vehicles* o AFV) comenzando y terminando el recorrido en el mismo punto, la detección de rutas que satisfagan la demandas de clientes y añadir ubicaciones en las que los vehículos puedan recargar combustible, llamadas estaciones de combustible alternativo (*Alternative-Fuel Station* o AFS). Por lo tanto, la gran diferencia entre G-VRP y VRP se basa en que el primero emplea logística "amigable" con el medio ambiente a la hora de realizar las rutas mencionadas.

Para este problema, se cuenta con los siguientes supuestos:

- Una flota de vehículos con combustible alternativo
- Cada vehículo posee una capacidad de combustible máximo.
- Los vehículos salen del almacén a hacer sus recorridos con el estanque de combustible lleno.
- Cuando un vehículo se mueva de un punto a otro (ya sea entre clientes o hacia el almacén), se consume una cantidad de combustible que es directamente proporcional a la distancia entre dichos clientes.
- Cuando un vehículo llega a una estación de servicio, recarga combustible hasta llenar su estanque.
- Cada vehículo tiene un consumo de combustible/distancia definido.
- Cuando un vehículo se mueve de un punto a otro (ya sea entre clientes o hacia el almacén), se agrega un tiempo de operación al vehículo, el cual es directamente proporcional a la distancia entre A y B.
- Los vehículos cuentan con una velocidad de conducción definida.
- Las visitas hacia un cliente y a una estación de servicio agregan tiempo de servicio al tiempo de operación del vehículo.

Las variables a considerar son las siguientes:

- Variable binaria: esta variable indicará si un vehículo viaja desde un nodo a otro. Si es 1, significa que realiza el viaje. Si es 0, significa que no lo toma en consideración.
- Variable de nivel de combustible: esta variable indicará el nivel de combustible al llegar a otro nodo. Va restaurando su valor a medida que se visiten AFS y/o depósito de vehículos.
- Variable de tiempo: esta variable va a representar la hora de llegada de un vehículo a un nodo. Se inicializa en 0 cuando se llega al depósito de vehículos.

Los parámetros que se consideran dentro del planteamiento del problema son los siguientes:

- Un número de consumidores y de estaciones de servicio.
- Coordenadas de almacén, consumidores y estaciones de servicio.
- Un tiempo de conducción máximo por vehículo.
- Capacidad máxima de combustible.
- Consumo de combustible por distancia.
- Velocidad de los vehículos.
- Tiempo de servicio a consumidor y a estación de combustible.

Por otro lado, las restricciones a considerar son las siguientes:

- El vehículo no puede moverse de un punto a otro si no cuenta con el combustible necesario.
- Cada cliente debe ser visitado solo una vez.
- Cada vehículo debe comenzar y terminar el recorrido en el almacén.
- Los vehículos no pueden exceder su tiempo máximo de conducción entre que inician el recorrido y lo terminan.
- No pueden realizarse más *tours* que cantidad de vehículos posea la flota (en la literatura no se aplica esta restricción).

Dentro de este problema existen diversas variantes. A continuación, se nombrarán algunas de ellos:

- **Metodología de vehículos ecológicos con múltiples tecnologías y recargas parciales (GVRP-MTPR)**

  En el artículo *A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges* [10] se propone un modelo VRP en el cual se utilizan vehículos eléctricos con autonomía limitada, los cuales necesitan recarga durante sus funciones. En esta metodología, se considera inicialmente la posibilidad de realizar una recarga parcial de energía en una estación.

- **Sistema de prevención de congestión vehicular basado en hormigas (AVCAS) que utiliza el modelo de consumo de combustible y emisión de ayuda de investigación y diseño de intersecciones señalizadas (SIDRA)**

  En el artículo *Green vehicle traffic routing system using ant-based algorithm* [6] se propone un modelo G-VRP que utiliza varios criterios, como el tiempo de viaje promedio, la velocidad, la distancia, la densidad del vehículo junto con la segmentación de la hoja de ruta para reducir el consumo de combustible tanto como sea posible al encontrar los caminos más cortos y menos congestionados para reducir el tráfico de vehículos, congestión y sus emisiones contaminantes.

3. Estado del Arte

3.1. Origen del Problema

Tal como se menciona en una de las secciones previas, el *Green-Vehicle Routing Problem* es una extensión del problema *Vehicle Routing Problem* (VRP). El VRP es una optimización combinatoria que implica encontrar un diseño óptimo de rutas recorridas por una flota de vehículos para servir a un conjunto de clientes [3], el cual tiene su origen en el estudio del impacto económico de las rutas de los vehículos en la organización que realiza el servicio de distribución [7].

El G-VRP se introduce por S. Erdogan y E. Miller-Hooks [5] como motivación para disminuir la emisión de gases nocivos para la salud y para el entorno ecológico. Como se menciona en el artículo *Survey of Green Vehicle Routing Problem: Past and future trends* [4], el uso excesivo de energía y contaminación del aire impusieron una amenaza a nuestro entorno ecológico. En Estados Unidos, el sector del transporte contribuía con el 28 % (EPA de EE.UU., 2009) de las emisiones nacionales de gases de efecto invernadero (GEI). Esto se debía en gran parte a que el 97 % de la energía de transporte de EE. UU provenía de combustibles derivados del petróleo (US DOT, 2010). Esto, encendió las alarmas tanto de los gobierno como del sector privado para comenzar a ser partícipes de una "campaña verde". Dentro de esa campaña, el gobierno de los Estados Unidos elaboró ciertas políticas energéticas (por ejemplo, la Ley de Política Energética (EPAct), 1992, 2005; Orden Ejecutiva (EO) 13423 y la Ley de Seguridad e Independencia Energética (EISA), 2007) con el fin de reducir el consumo de combustible fósil y así dar hincapié a los **combustibles alternativos**. En dicho país, durante décadas se realizaron esfuerzos con la intención de alejar a los conductores de los automóviles personales y llevarlos al transporte público y al transporte de mercancías de los camiones al ferrocarril, lo cual, tenía como objetivo reducir las millas recorridas por los vehículos por carretera y, por lo tanto, el uso de combustibles fósiles. Con la nueva "campaña verde", se centraron esfuerzos en la introducción de combustibles más limpios (por ejemplo, diésel ultra bajo en azufre) y tecnologías de motor eficientes, que reducen las emisiones durante las mismas millas recorridas y un mayor kilometraje por galón de combustible utilizado.

Al seguir un enfoque multifacético, se puso más atención en la explotación de fuentes alternativas de combustible más ecológicas (biodiésel, electricidad, etanol, hidrógeno, etc). Por lo tanto, distintos organismos comenzaron a utilizar flotas de vehículos de combustible alternativo (AFV). Lamentablemente, la falta de una infraestructura nacional para el reabastecimiento de combustible de los AFV presentó un obstáculo significativo para la adopción de tecnologías de combustibles alternativos por parte de empresas y agencias que buscaron la transición de las flotas tradicionales de vehículos a gasolina a las flotas de AFV (Melaina y Bremson, 2008). Para esa fecha, aproximadamente el 98 % del combustible utilizado en la flota de 138,000 AFV del gobierno federal (de los cuales, el 92.8 % en 2008 eran vehículos de combustible flexible que

podían funcionar con gasolina o combustible E85 a base de etanol) continúo siendo gasolina convencional como resultado de la falta de oportunidades para reabastecerse de combustible alternativo para el que fueron diseñados los vehículos (US DOE, 2010).

Por otro lado, como fue mencionado en el artículo *The green vehicle routing problem: A systematic literature review* [8], a pesar de que la implementación de las "tecnologías verdes" para reducir los gases de efecto invernadero han traído mejoras, las estadísticas muestran que la cantidad de contaminantes, principalmente, el equivalente de dióxido de carbono, ha aumentado y los problemas de salud y bienestar humanos derivados de los contaminantes ambientales se han intensificado drásticamente.

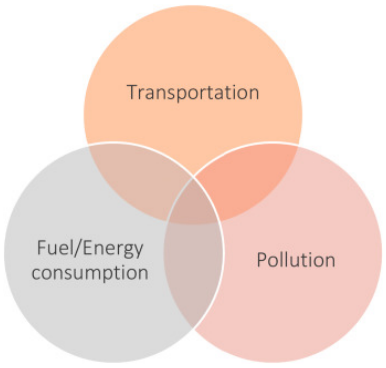


Figura 1. Tres categorías de términos de revisión estructurados de Green-VRP. [1]

Siguiendo la misma linea, se buscaba explorar la relación entre el efecto ambiental y el transporte a través de la **planificación de rutas** que permitieran implementar prácticas con un impacto positivo en la campaña verde. Es por esto que, **reducir el consumo de combustible y mejorar la eficiencia del transporte a nivel operativo** serían las acciones a seguir, estudiar e implementar. Para ello, también se debió tener en mente los siguientes factores:

- La velocidad de desplazamiento de los vehículos.
- El peso de la carga que transportan.
- La distancia de transporte.

Lo anterior concluyó en que, S. Erdogan y E. Miller-Hooks [5] enfocaran su estudio sobre el G-VRP hacia aquellas empresas y/o agencias que emplean una flota de vehículos para atender a clientes u otras entidades ubicadas en una amplia región geográfica. Dichas entidades trabajan con herramientas para ayudar a organizar recorridos de bajo costo, a fin de ahorrar dinero y tiempo como resultado del viaje a las ubicaciones de los clientes.

3.2. Metodologías de resolución propuestas por los autores

Desde su planteamiento, sus creadores [5] presentan al G-VRP a través de una formulación del tipo MILP (*Mixed-integer linear programming*). También, VRP pertenece a la familia de los problemas *NP-Hard*, es decir, el esfuerzo computacional requerido para su solución crece exponencialmente con el aumento del tamaño del problema. Entonces, como G-VRP nace de VRP, se puede decir que G-VRP también es un problema del tipo *NP-Hard*. Por lo tanto, encontrar una solución que permita resolverlo no es una tarea del todo sencilla.

En un intento de solución del problema, se buscó por técnicas que tuviesen la capacidad de brindar apoyo a la toma de decisiones para aquellas empresas o agencias que operaran una flota de AFV para los que existen estaciones de servicio limitadas. Es así como los autores ponen sobre la mesa las siguientes heurísticas:

- *Modified Clarke and Wright Savings* (MCWS).
- *Density-Based Clustering Algorithm* (DBCA).

Estas heurísticas proporcionan una capacidad de solución rápida. Sus pasos muestran cómo se pueden abordar las limitaciones adicionales del problema dentro de la heurística de construcción y mejora. Además, proporcionan intuición para el desarrollo de implementaciones más sofisticadas [5].

3.2.1. Heurística de MCWS

Paso 1: Crear n recorridos de vehículos de ida y vuelta $(v_0 - v_i - v_0)$, cada uno comenzando en el depósito v_0, visitando un vértice de clientes $v_i \in I$ y terminando en el depósito. Agregar cada recorrido creado a la lista de recorridos.

Paso 2: Calcular la duración y la distancia del recorrido para todos los recorridos en la lista de recorridos. Verificar la viabilidad de todos los recorridos iniciales de ida y vuelta con respecto al campo de prácticas y las restricciones de limitación de la duración del recorrido y categorícelos como factibles o inviables. Colocar todos los recorridos factibles en la lista de recorridos factibles y el resto en la lista de recorridos no factibles.

Paso 3: Para cada recorrido en la lista de recorridos no factibles, calcular el costo de una inserción de AFS entre los vértices del cliente v_i y el depósito $v_0, c(v_i, v_0) = d(v_i, v_f) + d(v_f, v_0) - d(v_i, v_0)$ para cada AFS $(v_f \in F'')$. Para cada uno de estos recorridos, insertar un AFS con el menor costo de inserción. Si después de la inserción de un AFS se cumplen las limitaciones de duración del recorrido y del campo de prácticas, agregar el recorrido resultante a la lista de recorridos factibles. Si la restricción del campo de prácticas no se cumple con la adición de ningún AFS, descartar el recorrido. No se considera ningún tour inicial que contenga más de un AFS.

Paso 4: Calcular los ahorros asociados con la fusión de cada par de recorridos en la lista de recorridos factibles. Para ello, primero identificar todos los vértices adyacentes al depósito en un recorrido. Crear una lista de pares de ahorro (SPL) que incluya todos los pares posibles de estos vértices (v_i, v_j) con la condición de que cada par esté formado por vértices que pertenecen a diferentes recorridos. Calcule los ahorros asociados con cada par de vértices en el SPL , $s(v_i, v_j) = d(v_0, v_i) + d(v_0, v_j) - d(v_i, v_j)$, donde $((v_i, v_j) \in I \cup F'')$. Clasificar los pares en el SPL en orden descendente de ahorros $s(v_i, v_j)$.

Paso 5: Mientras SPL no esté vacío:

i) Seleccionar y eliminar el par superior de vértices (v_i, v_j) en el SPL y combinar sus recorridos asociados.

ii) Para el seleccionado (v_i, v_j), verificar el rango de conducción y las restricciones de limitación de duración del recorrido.

iii) Si se cumplen ambas restricciones, agregar el recorrido resultante a la lista de recorridos factibles.

iv) Si la duración del recorrido resultante es menor que T_{max}, pero viola la restricción del rango de conducción, calcular el costo de inserción $c(v_i, v_j) = d(v_i, v_f) + d(v_f, v_j) -$

$d(v_i, v_0) - d(v_j, v_0)$ para el par de ahorros $((v_i, v_j) \in I \cup F'')$ para cada AFS. Insertar el AFS entre v_i y v_j con el menor costo de inserción para el cual el recorrido resultante sea factible. Comprobar la redundancia: si el recorrido contiene más de un AFS, considerar si es posible eliminar uno o más de los AFS del recorrido. Eliminar cualquier AFS redundante. Agregar el recorrido resultante a la lista de recorridos factibles.

v) Si algún recorrido ha sido añadido a la lista de recorridos posibles, retornar al paso 4. De lo contrario, detenerse.

Esta heurística termina entregando una solución factible para el G-VRP en el que se relajan ciertas restricciones que se mostrarán más adelante en el **modelo matemático**. La heurística continúa hasta que no se puedan fusionar más recorridos en la lista de recorridos factibles. El número de recorridos en la lista final de recorridos factibles es el más pequeño que se puede lograr mediante el proceso de fusión del Paso 5. Este procedimiento es consistente con la inclusión de un objetivo secundario de minimizar el tamaño de la flota.

Por otro lado, G-VRP permite la formación de ciclos los vértices AFS pueden ser visitados más de una vez, por más de un vehículo o no visitados en absoluto.

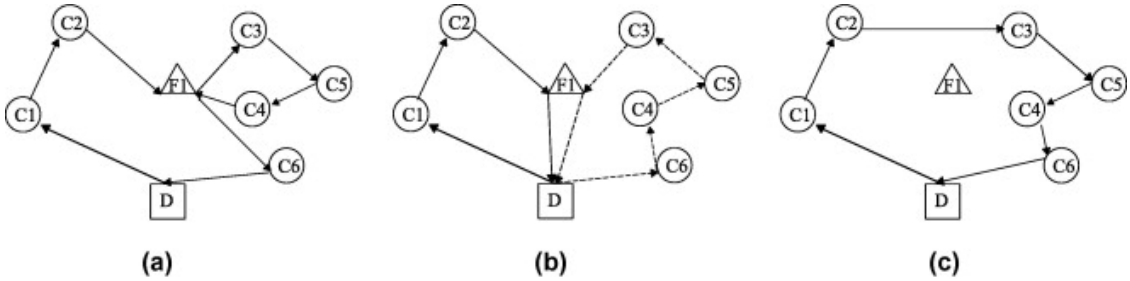


Figura 2. Posibles soluciones G-VRP factibles [5].

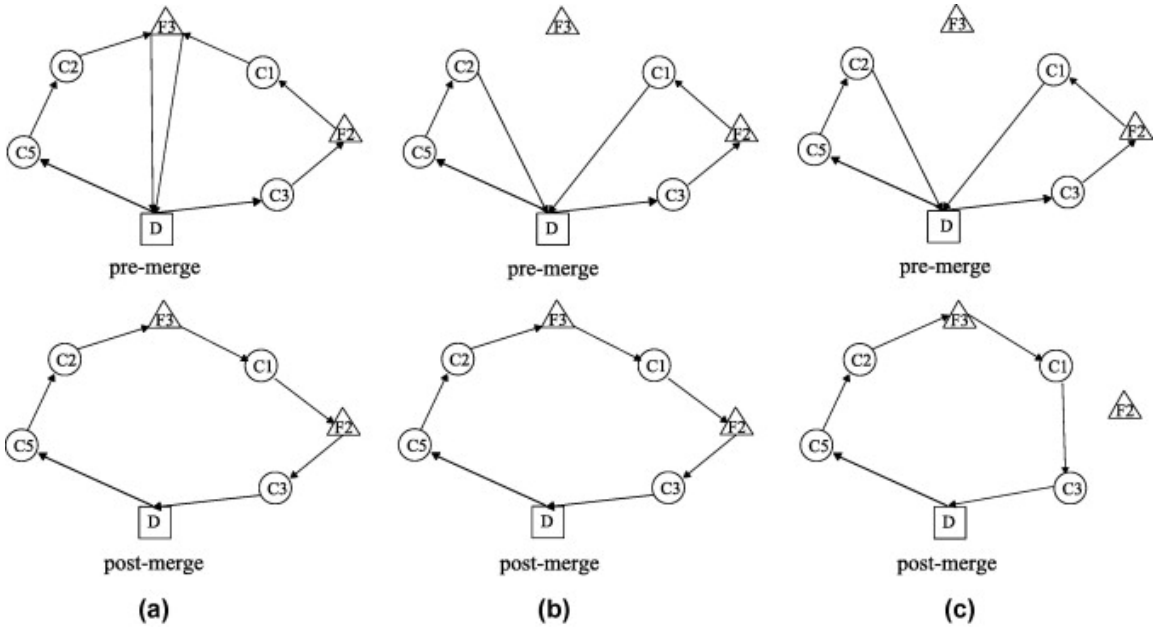


Figura 3. Características de la fusión en el G-VRP [5].

3.2.2. Heurística DBCA

Esta heurística aparece en G-VRP con la intención de explotar sus propiedades espaciales. La ubicación relativa de los clientes y AFS, así como sus distribuciones en el espacio, afectan significativamente la viabilidad y el número de visitas AFS requeridas. La idea clave del DBCA es que para cada vértice de un grupo, la vecindad de un radio dado debe contener al menos un número mínimo de vértices (*minPts*).

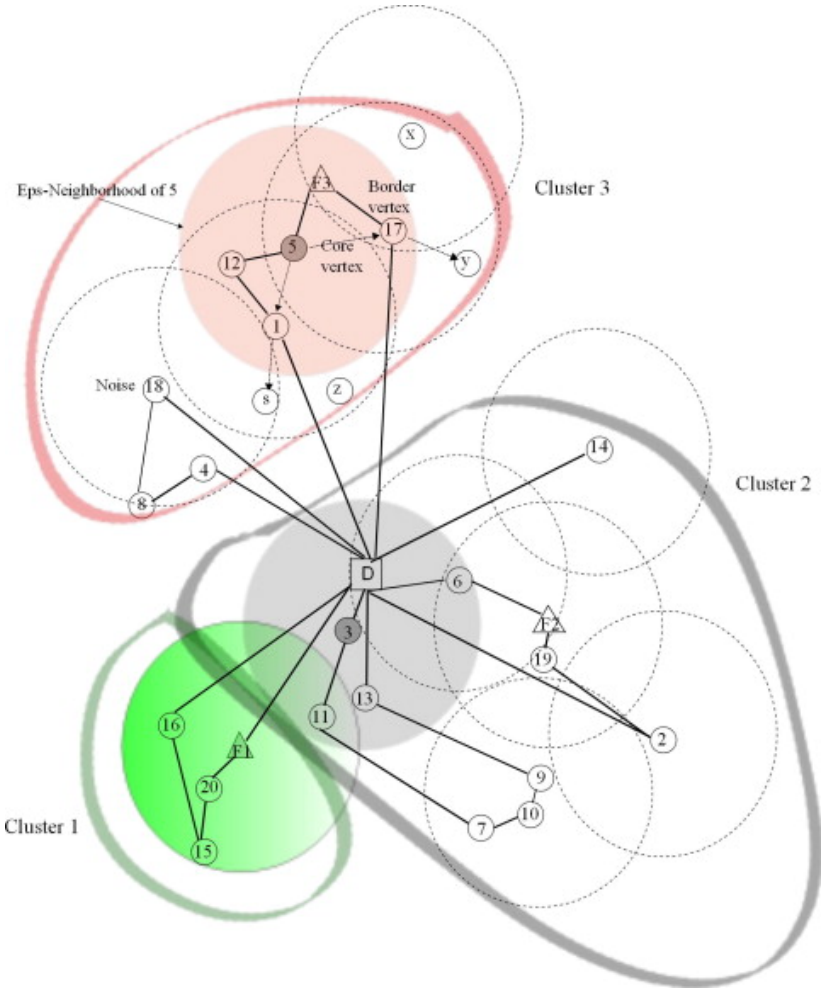


Figura 4. Formación de *clusters* mediante el algoritmo DBSCAN [5].

La notación utlizada en DBCA es la siguiente:

- m = número de rutas requeridas correspondiente al número de *clusters*.
- ϵ = radio parámetro utilizado para determinar un vértice $\epsilon-$vecindario.
- *minPts* = número mínimo de vértices en un $\epsilon-$vecindario de un vértice.
- $[\epsilon_{min}, \epsilon_{max}]$ = intervalo de búsqueda de ϵ.
- $[minPts_{min}, minPts_{max}]$ = el intervalo para el umbral de densidad para el cual DBCA busca diferentes esquemas de agrupamiento
- DBCA($[\epsilon_{min}, \epsilon_{max}]y[minPts_{min}, minPts_{max}]$)

Por lo tanto, los pasos a seguir para esta heurística son los siguientes:

1) **Agrupación**

- Para cada combinación de ϵ y *minPts* :
  - Para todo $v_i \in V'$:
    - Determinar el $\epsilon-$vecindario del vértice v_i con respecto a ϵ y *minPts*.
    - Si v_i satisface la condición $|N_\epsilon(v_j)| \geq minPts$ (condición vértice central), asignar un ID de *cluster* al vértice v_i y a todos los vértices de su vecindario ϵ.
- Para cada vértice v_i sin ID de *cluster*:
  - Para cada v_j sin ID de *cluste*r que esté conectado por densidad al vértice v_i, asignar el ID de *cluster* de v_i a v_j.
- Para cada vértice v_i sin ID del *cluster*, asignar el ID del *cluster* del vértice v_j con el ID de *cluster* más cercano a v_i.

Este paso termina con un conjunto de grupos para cada combinación de pares ϵ y *minPts*. El depósito se agrega a cualquier grupo en el que aún no esté incluido.

2) **Enrutamiento**

- Para cada conjunto de *clusters* correspondiente a cada emparejamiento de ϵ y *minPts*, ejecutar MCWS para construir recorridos en vehículos.

3) **Identificar el conjunto de rutas**

- Calcular la distancia total recorrida por todos los vehículos para el conjunto resultante de recorridos correspondiente a cada par (ϵ , *minPts*) del Paso 2 e identificar la combinación de parámetros (ϵ , *minPts*) que resulte en la menor distancia recorrida y generar el conjunto correspondiente de recorridos .

Al igual que la heurística de MCWS, el DBCA termina con un conjunto de recorridos que forman una solución factible para el G-VRP para el que se han relajado las mismas restricciones de MCWS.

3.2.3. Mejora Heurística

Las heurísticas antes mencionadas logran en conjunto construir recorridos factibles. Sin embargo, es posible aplicar una nueva técnica que permite mejorar el conjunto resultante de recorridos factibles al realizar un esfuerzo de reducir la distancia total que se debe recorrer. Aquí, S. Erdogan y E. Miller-Hooks [5] mencionan que los conceptos que involucran el intercambio de vértices entre recorridos y el intercambio y reordenamiento de dos vértices dentro del recorrido se personalizan para el G-VRP. De esta forma, se toma un conjunto de recorridos a los cuales se les aplica un un intercambio de vértices considerando un intercambio de vértices entre cada par de recorridos. Para cada uno de estos pares, se seleccionan dos vértices para un intercambio de posición. Así, si la distancia total de ambos recorridos juntos se reduce como resultado del intercambio y se pueden tomar medidas para mantener la viabilidad, se ejecuta el intercambio.

Como ejemplo, se aplica el intercambio y el reordenamiento de dos vértices dentro del recorrido, en el que cada par de vértices se considera para un intercambio. Se intercambia la posición dentro del recorrido de los dos vértices elegidos, creando un nuevo orden de recorrido. Si el nuevo pedido de tour no es factible, el intercambio no se realiza. De lo contrario, si uno o ambos

vértices elegidos para el intercambio son AFS, se verifica la redundancia de AFS y se considera la reubicación o intercambio de AFS con un AFS alternativo no programado para minimizar la duración del recorrido.

La **heurística de mejora** termina con un conjunto de recorridos para el G-VRP para los cuales las restricciones del posterior modelo se han relajado. La distancia total requerida para realizar los recorridos no será peor que la requerida de los recorridos iniciales a los que se aplica el procedimiento.

3.3. Experimentos Numéricos

Con el fin de evaluar la calidad de las soluciones obtenidas con las heurísticas posteriormente propuestas, se realizó una cantidad considerable de experimentos numéricos en casos de problemas pequeños, generados de forma aleatoria mediante la comparación con soluciones exactas obtenidas mediante la solución directa de la formulación G-VRP. Cabe destacar que, se consideró en dichos experimentos el impacto de la configuración de ubicación de AFS y del cliente y la densidad de AFS en la solución.

También, se diseñó un G-VRP más grande y realista utilizando la ubicación del depósito de una empresa de suministros textiles médicos en Virginia. Se creó un grupo de clientes para esta empresa en función de las ubicaciones de los hospitales en Virginia (VA), Maryland (MD) y el Distrito de Colombia (DC) utilizando Google Earth. Se consideró la conversión a biodiesel (B20 o superior), debido a la modesta densidad de estaciones de servicio de biodiesel en la región.EPA de EE. UU., 2002). Las estaciones reales de biodesel ubicadas en la región en el verano de 2009 se obtuvieron de un sitio web del DOE de EE.UU. (DOE de EE.UU., 2009).

En ambos conjuntos de experimentos, se estableció una capacidad del tanque de combustible de 60 galones y una tasa de consumo de combustible de 0.2 galones por milla en base a los valores promedio de los AFV propulsados por biodiesel (Fraer et al., 2005). Se asume que la velocidad promedio del vehículo es de 40 millas por hora (mph) y se asumió que la limitación de duración total del recorrido es de 11 h. Además, se asumió que los tiempos de servicio eran de 30 minutos en las ubicaciones de los clientes y de 15 minutos en las ubicaciones de AFS.

Con el fin de analizar resultados más acordes a la realidad, se tomará en cuenta solo el experimento de G-VRP más grande y realista.

3.3.1. Resultados de Experimentos en instancias realistas

Hay 21 estaciones de biodiesel disponibles públicamente en VA, MD y DC consideradas juntas (US DOE, 2009). Se consideraron cuatro escenarios basados en el cliente, como se describe en la Figura 5, en los que las 21 ubicaciones de AFS se consideran opciones.

Guión	Descripción	Detalles
1	Transición a AFV	111 clientes
2	Impacto del creciente número de clientes	El número de clientes aumentó en incrementos de 50 de 200 a 500, agregando clientes en ubicaciones aleatorias dentro del área de estudio al grupo de clientes del Escenario 1, manteniendo fijas las ubicaciones de AFS
3	Impacto de una mayor disponibilidad de AFS	Idéntico al Escenario 1, pero con AFS adicionales ubicados estratégicamente, aumentados en incrementos de 2 de 22 a 28
4	Impacto de los límites del campo de prácticas	Idéntico al escenario 1, pero el rango de conducción aumentó de 200 millas a 500 millas en incrementos de 50 millas

Figura 6. Escenarios de estudios de casos del mundo real [5].

Para los siguientes resultados, se emplearon las heurísticas MCWS y DBCA:

Ejemplo	Sin límite de rango de conducción (MCWS)			Algoritmo de Clarke y Wright modificado (MCWS)			Algoritmo de agrupamiento basado en densidad (DBCA) $\leqslant \varepsilon \leqslant 150, 1 \leqslant min$ Pts $\leqslant 30$		
	Costo total (millas)	Numero de recorridos	Clientes atendidos	Costo total (millas)	Numero de recorridos	Clientes atendidos	Costo total (millas)	Numero de recorridos	Clien atend
111c	4745.90 *4731.22*	17	109	5750,62 *5626.64*	20	109	5750,62 *5626.64*	20	109
200c	9358.63 *9355.56*	32	196	10617.02 *10428.59*	35	190	10617.83 *10413.59*	36	191 [a]
250c	11691.43 *11668.388*	40	244	11965.10 *11886.61*	41	235	11965.10 *11886.61*	41	236 [a]
300c	14782.08 *14762.41*	50	293	14331.30 *14242.56*	49	281	14331.30 *14229.92*	49	282 [a]
350c	17677.70 *17661,00*	59	343	16610.25 *16471.79*	57	329	16610.25 *16460.30*	57	329
400c	19968.97 *19936.75*	67	393	19568.56 *19472.10*	67	378	19196.71 *19099.04*	66	373
450c	23168.02 *21336.91*	77	443	21952.48 *21854.17*	75	424	21952.48 *21854.19*	75	424
500c	25032.38 *25024.94*	83	492	24652.15 *24527.46*	84	471	24652.15 *24517.08*	84	471

Figura 7. Resultados de la solución heurística [5].

La figura 7 muestra que se requieren 20 AFV para atender la misma cantidad de clientes atendidos por 17 vehículos para los cuales no se aplicarían limitaciones de autonomía. Por otro lado, se requiere un aumento del 19 % en la distancia de conducción para atender al mismo grupo

de clientes cuando se imponen limitaciones de autonomía (es decir, mediante la conversión de la flota de vehículos a AFV de biodiésel). A medida que el número de clientes aumentó de 200 a 500, la diferencia entre los clientes que no pudieron ser atendidos cuando no se aplicaron limitaciones del campo de prácticas en comparación con cuando se exigieron tales limitaciones aumentó de 2 a 21.

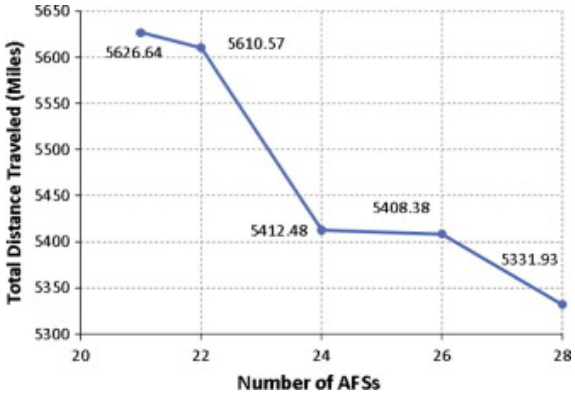
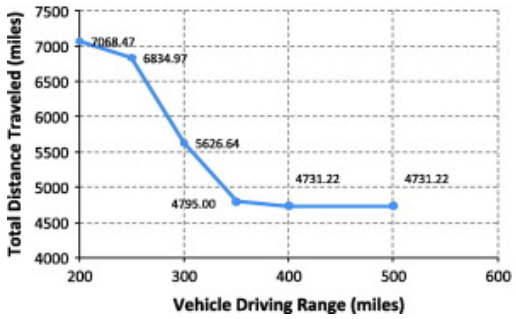


Figura 8. Efecto de aumentar los AFS [5].

La figura 8 indica que a medida que el número de AFS aumenta de 21 a 28 (un aumento de aproximadamente el 33 %), la distancia total recorrida disminuye en 295 millas (una disminución de aproximadamente el 5 %). Una mayor disponibilidad de AFS puede reducir los costos operativos de la flota de AFV; sin embargo, el ahorro de costos depende en gran medida de las ubicaciones específicas de las estaciones agregadas. Por lo tanto, puede ser beneficioso para la empresa buscar asociaciones con agencias o empresas que posean estaciones de servicio privadas en lugares bien posicionados o que mantengan una o más de sus propias instalaciones de servicio ubicadas estratégicamente dentro de un área operativa.



Driving range (miles)	Total Cost (miles)	Number of tours	Customers Served
200	7068.47	28	98
250	6834.97	25	107
300	5626.64	20	109
350	4795.00	17	109
400	4731.22	17	109
500	4731.22	17	109

Figura 9. Efecto de la autonomía de conducción del vehículo en la distancia total recorrida[5].

Finalmente, la figura 9 dice que a medida que el rango de conducción aumenta de 200 a 400 millas, la distancia de viaje requerida disminuyó en 2337 millas. Cualquier aumento en el rango de conducción más allá de las 400 millas no resultó en una solución mejorada, lo que indica que todos los clientes podrían ser atendidos dados los 21 AFS ubicados en la región.

3.4. Otras Metodologías

En la siguiente figura, se hace mención a aquellas metodologías y heurísticas que distintos investigadores han ido probando/implementando en la búsqueda de soluciones factibles para G-VRP.

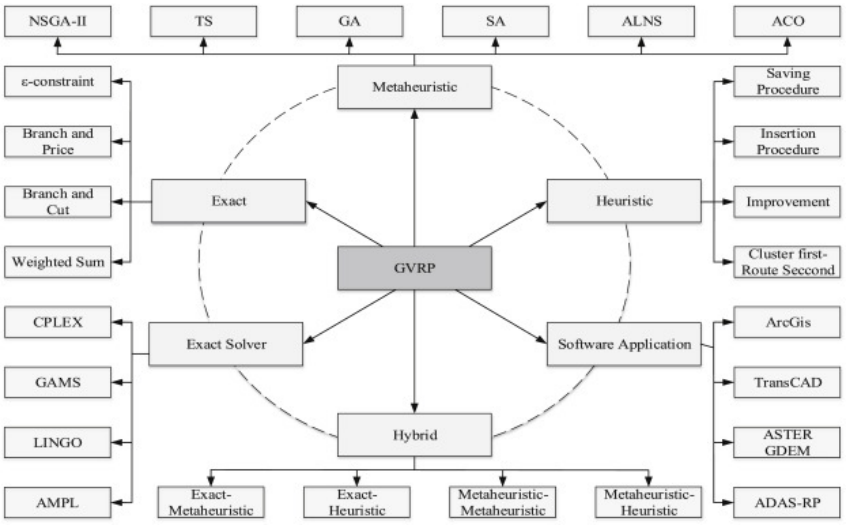


Figura 10. El marco general de las metodologías de solución [8].

Se explicará en breve (de forma general) cada categoría mencionada:

3.4.1. Algoritmos Metaheurísticos

De esta categoría, los métodos más significativos empleados para resolver GVRP aquí son: algoritmo genético (GA) (Xiong, 2010), SA (Omidvar y Tavakkoli-Moghaddam, 2012), TS (Ehmke et al., 2016), NSGA-II (Alinaghian y Zamani , 2019), optimización de colonias de hormigas (Li et al., 2019), ALNS (Demir et al., 2012 , colonia de abejas artificial (Zhang et al., 2014a), PSO (Norouzi et al., 2017), codiciosos aleatorizados procedimiento de búsqueda adaptativa (García-Álvarez et al., 2018), y el algoritmo del virus de la influenza (IVA) (Psychas et al., 2018). Por otra parte, se debe tener en mente que el uso de aplicaciones de software es un enfoque de solución extremadamente poderoso para aplicar a situaciones del mundo real. Entre otros, TransCAD (Christie et al., 2006), ArcGIS (Ericsson et al., 2006), el paquete de soluciones de enrutamiento de vehículos (Palmer, 2007), ASTER GDEM (Corréïa et al., 2010) y ADAS-RP (Minett et al., 2011) es el principal software utilizado en varios estudios [8].

3.4.2. Métodos Exactos

Los métodos de esta categoría son **extremadamente inadecuados** para resolver problemas de optimización a gran escala. Entre estos algoritmos, se consideran como emblemáticos el método de restricción (Soysal et al., 2014), rama y corte (Cheng et al., 2017), rama y precio (Hiermann et al., 2016), y suma ponderada (Zeng et al. , 2016). Otro investigadores han intentado resolver estos problemas con solucionadores exactos generales, como CPLEX (Taha et al., 2014), Lingo (Paksoy y Özceylan, 2014) y GAMS (Alkawaleet et al., 2014). Los métodos heurísticos se han adaptado a problemas específicos. Esta categoría puede incluir el procedimiento de ahorro (Aranda Uson et al., 2012), el procedimiento de inserción (Rao y Jin, 2012), el procedimiento de mejora (Oberscheider et al., 2013) y el procedimiento de agrupación-primera ruta-segunda (Erdoğan y Miller-Hooks, 2012) [8].

3.4.3. Algoritmos Híbridos

El uso de un enfoque específico puede llegar a provocar ciertas dificultades, como una solución de baja calidad, atrapamiento en óptimos locales en el espacio de búsqueda o tiempo de cálculo elevado; por lo tanto, los académicos hibridan dos o más algoritmos para emplear

simultáneamente fortalezas. Los métodos híbridos incluyen algoritmos exactos-metaheurísticos (Qian y Eglese, 2016), metaheurísticos-metaheurísticos (Jabir et al., 2015) y metaheurísticos-heurísticos (Maden et al., 2010) para obtener mejores resultados [8].

Finalmente, en el siguiente gráfico se puede apreciar la aplicación de diferentes enfoques de solución en la literatura.

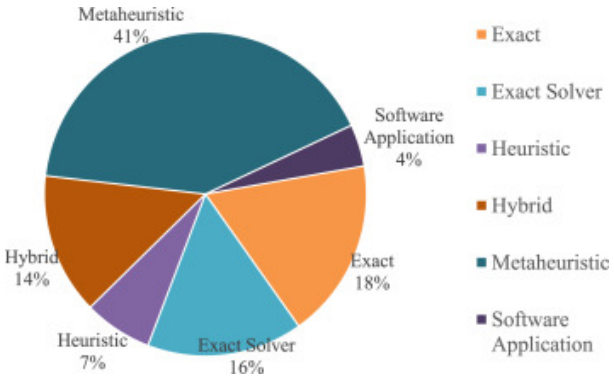


Figura 11. El porcentaje de uso de diferentes metodologías de solución en la literatura [8].

4. Modelo Matemático

A partir de los artículos *'A Green Vehicle Routing Problem'* [5] y *'A new Mathematical Programming Model for the Green Vehicle Routing Problem'* [2], se procederá a plantear un modelo matemático del problema asignado.

El modelo matemático que se define para el G-VRP consta de un grafo $G = (N, A)$, donde el conjunto de vértices $N = I \cup \{0\}$ corresponde a una combinación del conjunto de clientes $I = \{v_1, v_2, ..., v_n\}$ y 0 el depósito de los vehículos, y $A = \{(i, j) : i \in N, j \in N, i \neq j\}$ corresponde a las aristas que conectan los vértices de N. El conjunto F de las estaciones de combustible alternativo (*Alternative Fuel Stations* o AFS) es conocido y cada estación que se encuentra en dicho conjunto se representa por s. Luego, se entregan los siguientes datos: $\forall i \in N \cup F, \forall j \in N \cup F, i \neq j$, tiempo de viaje t_{ij} (que representa el tiempo de viajar de i hacia j), la distancia d_{ij} (que representa la distancia entre i y j); $\forall i \in I, p_i$ es el tiempo de servicio de los vehículos en el vértice i, mientras que $\forall s \in F$ viene a representar el tiempo que demora volver a llenar el estanque de los vehículos; Q es la capacidad del estanque de éstos para volver a llenarse de combustible.

Se sabe que G-VRP busca encontrar un máximo de m recorridos, uno para cada vehículo, que comience y termine en el depósito, visitando así un subconjunto de vértices que incluyen AFS cuando sea necesario, de tal forma que la distancia total recorrida sea mínima. Luego, se aplican las restricciones de la autonomía de conducción del vehículo dictadas por las limitaciones de la capacidad del tanque de combustible y las restricciones de duración del recorrido destinadas a restringir la duración del recorrido a un límite T_{max} preestablecido.

Cabe señalar que, se tiene como supuesto que todos los clientes pueden ser atendidos por un vehículo que comienza su recorrido en el depósito y regresa al depósito después de visitar al cliente directamente dentro de T_{max}. Otro supuesto del problema es que todos los clientes pueden ser visitados directamente por un vehículo que comienza y regresa al depósito con, como máximo, una visita a un AFS, lo cual no excluye la posibilidad de elegir un *tour* que atienda a

varios clientes y contenga más de una visita a un AFS.

De esta forma, el modelo planteado se basa en calcular los conjuntos L_{ij} de AFS's que podrían ser convenientes para que un AFV (*alternative fuel vehicle* o vehículo de combustible alternativo) se mueva de manera óptima desde i hacia j, $\forall (i,j) \in A$.

Así, los conjuntos recién mencionados son 'computados' de la siguiente forma:

- Sea $s* = arg \ \ min_{s \in F: d_{is} \leq \frac{Q}{r}, d_{sj} \leq \frac{Q}{r}} \{d_{is} + d_{sj}\}$. Dicha función buscar reducir la desviación entre los vértices i y j.
- Al comienzo del recorrido, $L_{ij} := F$ y para cada $s \in F$, si $d_{is} > d_{is*}$ y $d_{sj} > d_{s*j}$, o $d_{is} > \frac{Q}{r}$, o $d_{sj} > \frac{Q}{r}$, la AFS 's' es removida de L_{ij}.
- Además, $\hat{t}_{ijs} = t_{is} + t_{sj} - t_{ij}$ viene siendo el tiempo de desvío de un AFV para visitar $s \in L_{ij}$, pasando de i a j.
- Finalmente $\hat{d}_{ijs} = d_{is} + d_{sj} - dij$ es la distancia de desvío de un AFV para visitar $s \in L_{ij}$, pasando de i a j.

También, se modelará G-VRP utilizando las siguientes variables binarias, :

- $x_{ij} = \begin{cases} 1 & \text{si el nodo } j \text{ es visitado después del nodo } i \text{ (de forma directa o con una AFS entre medio)} \\ 0 & \text{en otro caso} \qquad \forall (i,j) \in A \end{cases}$
- $z_{ijs} = \begin{cases} 1 & \text{si la AFS } s \text{ es visitada para ir de } i \text{ a } j. \\ 0 & \text{en otro caso} \qquad \forall (i,j) \in A, \forall s \in L_{ij} \end{cases}$

Además, se usarán las siguientes variables continuas $\forall i \in N$:

- El nivel de combustible restante y_i de un AFV en i.
- El tiempo τ_i en el que se llega al nodo i.

Finalmente, el MILP (*mixed–integer linear programming* o programa lineal de enteros mixtos) se detalla de la siguiente forma:

$$min \sum_{i,j \in A} \left(d_{ij} x_{ij} + \sum_{s \in L_{ij}} \hat{d}_{ijs} z_{ijs} \right) \tag{1}$$

sostenido a

$$\sum_{s \in L_{ij}} z_{ijs} \leq x_{ij} \quad \forall (i,j) \in A \tag{2}$$

$$\sum_{j \in N: j \neq i} x_{ij} = 1 \quad \forall i \in I \tag{3}$$

$$\sum_{i \in N: i \neq j} x_{ji} = \sum_{i \in N: i \neq j} x_{ij} \quad \forall j \in N \tag{4}$$

$$\sum_{j \in N: j \neq 0} x_{0j} \leq m \tag{5}$$

$$\sum_{j \in N: j \neq 0} x_{j0} \leq m \tag{6}$$

$$\tau_j \geq t_i + (t_{ij} + p_i)x_{ij} + \sum_{s \in L_{ij}} (\hat{t}_{ijs} + p_s)z_{ijs} - l_0(1 - x_{ij}) \quad \forall \in N, \forall j \in I, i \neq j \tag{7}$$

$$\tau_j \leq T_{max} - (t_{j0} + p_j) - \sum_{s \in L_{j0}} (\hat{t}_{j0s} + p_s)z_{j0s} \quad \forall j \in N \setminus \{0\} \tag{8}$$

$$y_i \leq \sum_{s \in Lij} (Q - r \cdot d_{sj})z_{ijs} + Q(1 - \sum_{s \in L_{ij}} z_{ijs}) \quad \forall j \in I, \forall i \in I, i \neq j \tag{9}$$

$$y_j \leq y_i - r \cdot d_{ij} + 2Q(1 - x_{ij} + \sum_{s \in L_{ij}} z_{ijs}) \quad \forall j \in I, \forall i \in N, i \neq j \tag{10}$$

$$y_i \geq r \cdot d_{i0}(x_{i0} - \sum_{s \in L_{ij}} z_{i0s}) \quad \forall i \in I \tag{11}$$

$$\sum_{s \in L_{ij}} r \cdot d_{s0} z_{i0s} \leq Q \quad \forall i \in I \tag{12}$$

$$y_i \geq \sum_{s \in L_{ij}} (r \cdot d_{is} z_{ijs}) \quad \forall (i,j) \in A \tag{13}$$

$$y_0 \leq Q \tag{14}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i,j) \in A \tag{15}$$

$$y_i \geq 0, \tau_i \geq 0 \quad \forall i \in N \tag{16}$$

$$z_{ijs} \in \{0, 1\} \quad \forall (i,j) \in A, \forall s \in L_{ij} \tag{17}$$

Entonces, tras plantear el modelo matemático del problema, se procede a explicar en qué consiste:

- La función objetivo (1) minimiza la distancia total de viaje.

- Entre cada par de clientes, se puede visitar como máximo un AFS (2) mientras que cada cliente debe ser visitado exactamente una vez (3).
- La continuidad de la ruta está asegurada por (4).
- El número de AFV está limitado por (5) - (6).
- La hora de llegada a cada nodo se rige por (7) que también excluye los sub-recorridos.
- La duración máxima de la ruta se aplica mediante (8).
- El nivel de combustible, en cada nodo, está regido por (9) - (10).
- Un AFV, después de visitar a su último cliente, debe tener suficiente combustible para regresar al depósito sin repostar (11) o repostar (12).
- En caso de repostar, (13) garantiza que el AFV puede alcanzar el AFS seleccionado.
- Los AFV salen completamente recargados del depósito gracias a (14).
- La naturaleza de las variables se especifica en (15) - (16) (17).

5. Representación

5.1. Nodos

Para la representación del problema, se decidió expresar cada nodo como una estructura, la cual posee todos los datos de dicho nodo:

```
1        struct nodo
2        {
3            string ID;
4            string type;
5            double longitude;
6            double latitude;
7        };
8
```

Código 1: Definición de la clase nodo

Cada nodo presente en la instancia es almacenado en un vector de estructuras, el cual contiene todas las estructuras que corresponden a los nodos junto a la información de cada nodo. Dicha acción se realiza a la hora de leer el archivo de la instancia. En el *main* se crea una variable llamada *'n'*, la cual es un vector de estructuras de tipo nodo y en ella se almacenaran todos los datos de los nodos de la instancia. Por ejemplo, el valor *n[0]* vendría siendo el primer nodo de la instancia y para saber a que tipo de nodo corresponde se le aplica *n[0].type*.

Se opta por esta estrategia ya que, a la hora de calcular distancias y/o aplicar restricciones, se puede acceder al nodo a través de índices, lo cual facilita el acceso a los valores de éste (ya sea para ver la *ID* del nodo o de si se trata de un nodo de tipo cliente o de tipo estación de servicio).

5.2. Parámetros

De la misma forma, los parámetros del archivo también se almacenan en una estructura, la cual posee la información de los vehículos de dicha instancia:

```
struct parametros
{
  int Q;
  float r;
  int TL;
  int v;
  int m;
};

```

Código 2: Definición de la estructura parámetros

En el *main*, se define una variable llamada *'parametros'*, la cual es una estructura de tipo parametros que contiene todos los datos sobre los vehículos. Para usar dichos parámetros, se llaman de la siguiente forma en el *main*:

```
  double Q = parametros.Q; //Capacidad maxima del estanque
double r = parametros.r; //Tasa de consumo de combustible por distancia
double TL = parametros.TL; //Tiempo m ximo del recorrido
double v = parametros.v; //Millas por hora
v = v/60; //Millas por minuto
double m = parametros.m; //Cantidad de veh culos

```

Código 3: Variables de la estructura parámetros

5.3. Matriz de distancia entre nodos

La distancia entre nodos se almacena en un vector de vectores de datos tipo *'double'* llamado *'distances'*, en el cual se almacenan todas las distancias que existen entre los nodos. Para calcular las distancias se utiliza la fórmula *'Haversine'*.

$$dist_{n,n} = \begin{pmatrix} 9999 & d_{1,2} & \cdots & d_{1,n} \\ d2,1 & 9999 & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & 9999 \end{pmatrix} \tag{18}$$

Para las distancias, se coloca como supuesto que en caso de calcular la distancia entre un nodo consigo mismo, esta sea de 9999, ya que más adelante se aplicará un algoritmo tipo *Greedy* que buscará distancias mínimas. Por lo mismo, si se considera la distancia entre un nodo y otro igual a 0, siempre tenderá a ir hacia si mismo, generando un bucle sin salida.

Para realizar en código la representación matemática de las variables binarias de las distancias d_{ij} entre nodos, se le asigna un identificador de tipo entero a cada nodo, el cual corresponde a su ubicación en el vector de estructuras *'n'*. En el vector de vectores *'distances'* se calcula la distancia del nodo i al nodo j, de tal forma que se genera una matriz simétrica de distancias, donde la diagonal corresponde a distancias iguales a 9999 (tal como se explica anteriormente). Para calcular las distancias de los nodos, se accede al valor *'longitude'* y *'latitude'* de cada estructura i y j, luego dichos valores se ingresan en la función *'Haversine'*.

Al tener todas las distancias entre los nodos, se hará uso de un algoritmo *'Greedy'* que buscará cual nodo es el más cercano al nodo actual y, en caso de que se cumplan las restricciones de tiempo, combustible, que no se esté visitando una *AFS* tras visitar a otra *AFS* y/o depósito y que se pueda llegar al depósito desde la posición actual, dicho nodo será visitado y almacenado en un vector de *'strings'*, el cual retornará la solución inicial del problema. Se decide usar un vector de *strings* ya que se irán almacenando en él los índices de los nodos del vector de estructuras *'n'*, por lo que a través de dicho índice se mostrará el *ID* del nodo que será parte de la solución inicial.

Finalmente, se buscará minimizar la distancia d_{ij} total de la solución inicial obtenida con la técnica incompleta *Tabú Search*.

6. Descripción del algoritmo

El algoritmo inicia su labor al realizar procesamiento de texto con la función *'generaNodosY-Parametros'*. Esto tiene como fin poder obtener los nodos, información de los nodos y parámetros de dicha instancia del problema. Tras dicho procesamiento, es posible saber la cantidad de nodos que hay, el nombre de éstos, tipo de nodo (cliente, *AFS* o depósito), latitud y longitud, cantidad de combustible máximo de los *AFV* (vehículos), tiempo máximo del recorrido, tasa de consumo de combustible, velocidad promedio de los vehículos y cantidad de vehículos. Como supuesto, los vehículos pueden salir y volver al depósito todas las veces que sean necesarias para completar el recorrido.

6.1. Creación de la solución inicial

Tras realizar el procesamiento de texto, cada nodo de la instancia se almacena en un vector de estructuras, de tal forma que cada elemento de dicho vector contiene información de un nodo. Luego se recorre este vector de estructuras y se calculan la distancias entre todos los nodos, de tal forma que dichos valores se almacenan en un vector de vectores (que simula una matriz de 2 dimensiones) por fila y luego por columnas, donde la diagonal es la distancia entre un nodo consigo mismo, la cual es 9999. Luego, los parámetros del archivo se almacenan en variables, las cuales se usarán para aplicar restricciones más adelante.

Ahora, a través de un algoritmo *Greedy* se irá buscando el nodo más cercano al actual. De esta forma, se generará un camino que comience y termine en el depósito, el cual se irá almacenando en una variable llamada *'nodosVisitados'*. Para ello, se creará una variable auxiliar llamada *'nodos'*, la cual almacenará todos los nodos de la instancia. Se irán eliminando de *'nodos'* todos aquellos que sean de tipo cliente y que sean seleccionados por *Greedy* para ser añadidos a *'nodosVisitados'*. Además, se creará otra variable auxiliar llamada *'Fs'* en la que se guardará la distancia que tiene cada nodo con la estación de servicio más cercana.

Se buscará, además, que la solución inicial cumpla con las restricciones de combustible y tiempo.

Algorithm 1: Creación solución inicial

```
1  Se agrega el depósito como primer nodo de nodosVisitados
2  for i = último nodo visitado, i < número de nodos, i++ do
3      for j = nodo candidato a visitar, j < número de nodos, j++ do
4          if nodo i es distinto al nodo j then
5              if Q alcanza para volver al depósito desde j || El nodo j es distinto de cliente
                || Q alcanza para llegar al AFS más cercano then
6                  if Al visitar j no se supera el tiempo límite TL then
7                      if distancia de i a j < distanciaMínima then
8                          if ni i ni j son AFS y/o depósito then
9                              distanciaMinima = distancia de i a j
10                             nodoFueVisitado = true
11     if Q alcanza para ir de i a j && nodoFueVisitado then
12         Se usa la cantidad Q de i a j
13         Se considera el tiempo de viaje de i a j
14         Se agrega j a los nodosVisitados
15         if j es un cliente then
16             Se agregan 30 minutos al tiempo
17             Eliminar j de nodos
18             Eliminar la distancia de j a algún AFS en Fs
19         else if j es un AFS then
20             Se agregan 15 minutos al tiempo
21             Se llena el estanque Q
22         else
23             Se reinicia el tiempo
24             Se llena el estanque Q
25     if Ya no quedan clientes en nodos then
26         Se termina el algoritmo
27 Se agrega el depósito al final del recorrido de nodosVisitados
28 Se calcula la distancia de la solución inicial
29 return nodosVisitados
```

6.2. Factibilidad de la Solución

Para la posterior evaluación de las soluciones en *Tabú Search*, se crea la función *esFactible*, la cual verifica si se cumplen las restricciones del problema. En caso de que se cumplan, retorna *true*, en caso contrario retorna *false*.

Algorithm 2: Factibilidad de la Solución

```
for i < cantidad de nodos de la solución do
    if distancia nodo anterior y nodo i = 9999 then
        se pasa al siguiente valor de i
    Q -= combustible usado del nodo previo al nodo i
    tiempo += tiempo usado en viajar del nodo previo al nodo i
    if Q < 0 then
        return false
    if El nodo i es un AFS then
        Se recarga Q
        Se agregan 15 minutos al tiempo
    else if El nodo i es un cliente then
        Se agregan 30 minutos al tiempo
    else
        Se recarga Q
        Se reinicia el tiempo
    if El tiempo es mayor a TL then
        return false
return true
```

6.3. Iteración en *Tabú Search* mediante el movimiento *2OPT*

Al tener creada una solución inicial, comenzará el ciclo de iteraciones para crear el vecindario dentro de *Tabú Search*. Para ello, se aplicará el movimiento *2OPT*, el cual va cambiando el orden de nodos de la solución inicial para posteriormente comprobar si dicha solución será factible. En caso de no ser factible, se le aplicará un castigo, el cual se basa en agregarle 200 millas a su distancia de recorrido. Dicho movimiento *2OPT* permite mantener la extensión original de la solución inicial y también permite obtener un nivel aceptable de soluciones factibles.

Algorithm 3: Tabú Search

```
1  [=0.5]
2  solInicial = solución inicial obtenida con Greedy
3  mejorSolución = mejor solución encontrada
4  LT = lista tabú de tamaño 15
5  for i iteraciones do
6      Generar vecino actual con 2OPT
7      Calcular distancia del recorrido del vecino actual
8      solucionVecino = distancia del recorrido del vecino actual
9      if solucionVecino < solInicial && solucionVecino < mejorSolucion then
10         if solucionVecino no es factible then
11             mejorSolucion = solucionVecino + castigo
12         else
13             mejorSolucion = solucionVecino
14         if LT está llena then
15             Se elimina el primer movimiento tabú de LT
16             Se agrega el movimiento realizado en vecino a LT
17         else
18             Se agrega el movimiento realizado en vecino a LT
19     else if solucionVecino < solInicial then
20         if solInicial no es factible then
21             solInicial = solucionVecino + castigo
22         else
23             solInicial = solucionVecino
24         if LT está llena then
25             Se elimina el primer movimiento tabú de LT
26             Se agrega el movimiento realizado en vecino a LT
27         else
28             Se agrega el movimiento realizado en vecino a LT
29     else
30         if solInicial no es factible then
31             solInicial = mejor solucion de las peores encontradas + castigo
32         else
33             solInicial = mejor solucion de las peores encontradas
34         if LT está llena then
35             Se elimina el primer movimiento tabú de LT
36             Se agrega el movimiento realizado en vecino a LT
37         else
38             Se agrega el movimiento realizado en vecino a LT
39 return mejorSolucion
```

7. Experimentos

El algoritmo descrito en la sección 6 fue implementado en C++ y compilado con g++ 7.1.1. Los distintos experimentos fueron llevados a cabo en un computador Acer Nitro 5 con 16 GB de memoria RAM y un procesador AMD Ryzen 5 3550H de 4 núcleos, todo esto bajo el sistema operativo Ubuntu 20.04.1 LTS.

Los experimentos con los cuales fue probado el código se basan en los datasets 20c3sU1, 20c3sU2, 20c3sU3, Large VA Input_111c_21s, Large VA Input_200c_21s, Large VA Input_250c_21s,

S1_2i6s, S1_4i2s, S1_4i4s, S1_4i6s, los cuales aparecen en [5]. Cada uno de ellos posee un número de clientes que varía entre 20 y 250. De esta forma, se experimentó con la siguiente cantidad de nodos:

- 20c3sU1: 24 nodos
- 20c3sU2: 24 nodos
- 20c3sU3: 24 nodos
- Large VA Input_111c_21s: 132 nodos
- Large VA Input_200c_21s: 222 nodos
- Large VA Input_250c_21s: 271 nodos
- S1_2i6s: 27 nodos
- S1_4i2s: 23 nodos
- S1_4i4s: 25 nodos
- S1_4i6s: 27 nodos

A cotinuación, se adjunta una tabla que contiene las instancias con sus nombres, cantidad de nodos, cantidad de combustible, tasa de consumo de combustible, tiempo máximo del recorrido y velocidad de los vehículos:

Instancia	Nodos	Q	r	TL	v
20c3sU1	24	60	0.2	14	40
20c3sU2	24	60	0.2	14	40
20c3sU3	24	60	0.2	14	40
Large VA Input_111c_21s	132	60	0.2	15	40
Large VA Input_200c_21s	222	60	0.2	15	40
Large VA Input_250c_21s	271	60	0.2	15	40
S1_2i6s	27	60	0.2	14	40
S1_4i2s	23	60	0.2	14	40
S1_4i4s	25	60	0.2	14	40
S1_4i6s	27	60	0.2	14	40

Cuadro 1: Instancias del problema

Se puede notar que no aparece la cantidad de vehículos de cada instancia. Lo anterior se debe a que se permitió que los vehículos entraran y salieran del depósito las veces que fueran necesarias para terminar el recorrido, por lo que no hubo restricción de vehículos.

Debido a que se implementa el algoritmo *Tabú Search*, el único parámetro que se modificó fue la cantidad de iteraciones (de 5 a 1000 iteraciones). El objetivo es observar como se comporta el resultado obtenido a medida que se van aumentando las iteraciones en el algoritmo y ver como se comparta la exploración de este.

Para la parte de resultados, se va a comparar los resultados que obtuvieron los autores de [5] en sus instancias, los cuales son los siguientes:

	Costo Total con MCWS [millas]
20c3sU1	1843,52
20c3sU2	1614,15
20c3sU3	1969,64
Large VA Input$_1$11$c_2$1s	5750,62
Large VA Input$_2$00$c_2$1s	10617,02
Large VA Input$_2$50$c_2$1s	11965,19
S1_2i6s	1614,15
S1_4i2s	1589,6
S1_4i4s	1599,6
S1_4i6s	1599,6

Cuadro 2: Resultados de la instancias de los autores

8. Resultados

Se ejecutó el algoritmo Tabu Search con Greedy, utilizando una cantidad de i iteraciones con $i \in [5, 1000]$. Los resultados obtenidos se adjuntan en la siguiente tabla:

Distancia/Instancia	Sol. Inicial	5 Iteraciones	10 Iteraciones	50 Iteraciones	100 Iteraciones	500 Iteraciones	1000 Iteraciones
20c3sU1	1727,12	1619,1	1619,1	1619,1	1619,1	1619,1	1619,1
20c3sU2	1555,4	1429,23	1474,77	1474,77	1474,77	1474,77	1474,77
20c3sU3	1553.39	-	-	-	-	-	-
Large VA Input_111c_21s	4040.66	-	-	-	-	-	-
Large VA Input_200c_21s	7435.79	-	-	-	-	-	-
Large VA Input_250c_21s	8164.96	-	-	-	-	-	-
S1_2i6s	1590.61	1494.09	1494.09	1494,09	1494,09	1494,09	1494,09
S1_4i2s	1325,39	-	-	-	-	-	
S1_4i4s	1380.25	-	-	-	-	-	-
S1_4i6s	1626.42	1622,42	1622,42	1622,42	1622,42	1622,42	1622,42

Cuadro 3: Distancias obtenidas según cantidad de iteraciones

Se aprecia que en varias instancias, desde cierto número de iteraciones en adelante se repite el mismo valor de distancia, es decir, el algoritmo se queda atrapado en un óptimo local, lo cual le impide explorar otros vecindarios.

De la misma forma, se calculan los tiempos que el algoritmo demora en entregar un resultado según la cantidad de iteraciones recibidas:

Tiempo[s]/Instancia	5 Iteraciones	10 Iteraciones	50 Iteraciones	100 Iteraciones	500 Iteraciones	1000 Iteraciones
20c3sU1	0,468	0,9531	4,8	9,8	49.6719	97.5625
20c3sU2	0.3125	0.609375	3.26562	6.57812	32.75	63.625
20c3sU3	-	-	-	-	-	-
Large VA Input_111c_21s	-	-	-	-	-	-
Large VA Input_200c_21s	-	-	-	-	-	-
Large VA Input_250c_21s	-	-	-	-	-	-
S1_2i6s	0.484375	1.04688	5.125	10.5156	51.5625	106.5
S1_4i2s	-	-	-	-	-	
S1_4i4s	-	-	-	-	-	-
S1_4i6s	0.4375	0.90625	4.28125	8.57812	44.2188	86.7344

Cuadro 4: Tiempo que tarda el algoritmo en terminar según la cantidad de iteraciones

Por lo tanto, en una tabla se colocan el porcentaje de la diferencia en millas que hay entre los resultados de [5] y los mejores resultados obtenidos en el experimento:

Distancia/Instancia	Diferencia Mejora
20c3sU1	12,17 %
20c3sU2	8,63 %
20c3sU3	22,31 %
Large VA Input_111c_21s	29,72 %
Large VA Input_200c_21s	29,97 %
Large VA Input_250c_21s	31,76 %
S1_2i6s	7,43 %
S1_4i2s	16,62 %
S1_4i4s	13,71 %
S1_4i6s	-1,426 %

Cuadro 5: Diferencia de los resultados obtenidos versus los resultados de los autores

Se puede notar que se mejoró la mayoría de las distancias que los autores obtuvieron. Esto se puede deber al tipo de lógica que usa el algoritmo de ellos comparados con el algoritmo de este proyecto, junto a otras consideraciones que aquí se pudieron pasar por alto.

Hubo problemas con las siguientes instancias:

- 20c3sU3
- Large Va Input_111c_c_21s
- Large Va Input_200c_c_21s
- Large Va Input_250c_c_21s
- S1_4i2s
- S1_4i4s

Al parecer, en las instancias grandes hubo un error a la hora de realizar el movimiento *2OPT* y encontrar una mejor solución, ya que al tener una cantidad gigantesca de nodos, hubo algún problema de combinación de nodos generado por la lógica del código. Por otro lado, en las instancias pequeñas hubo algún error aleatorio que no permitió aplicar *Tabú Search* de forma correspondiente, ya que en instancias de valor similar, se aplica el algoritmo *Tabú Search* sin problema.

9. Conclusiones

A pesar de ser un problema reciente, han habido una gran cantidad de de *papers* que se han basado en el *GVRP*. A partir de ello, se han ido desarrollando distintas técnicas y/o algoritmos con el fin de poder obtener una solución cada vez más óptima, siempre y cuando se considere el escenario en el que se aplica el problema.

En el caso particular de la técnica *Tabú Search*, el desafío consiste en obtener una solución inicial para luego aplicarle uno o más movimientos que permitan generar vecindarios con distintas soluciones, las cuales deben ser factibles para las restricciones del problema, que en este caso, principalmente son combustible y tiempo.

La elección de una solución inicial demostró ser bastante desafiante (de hecho más del 60 % del tiempo empleado en este proyecto fue destinado a la creación de dicha solución) debido a

que existen escenarios en los que, por ejemplo, el tiempo no alcanza para llegar devuelta al almacén o la cantidad de combustible no alcance para terminar el tour, por lo que había que optar siempre por el siguiente nodo que permitiera que dichas variables se aplicaran de forma factible.

Por otro lado, el movimiento que se aplique debe ser pensado para sacarle provecho a la solución inicial, lo cual facilite la obtención de nuevas, mejores y factibles soluciones, de tal forma que se pueda explotar y explorar a lo largo del espacio de búsqueda. Sin embargo, en este proyecto no se logró explorar del todo bien ya que el algoritmo *Tabú Search* queda atrapado en óptimos locales, dificultando así la exploración. Pero, a pesar de que el algoritmo no logre ubicar otros vecindarios con posibles mejores soluciones, calcula rápido una solución mejor que la inicial, por lo que el resultado puede usarse como una solución inicial para futuros algoritmos.

En conclusión, utilizar movimientos que mantengan el mismo tamaño de la solución no son los suficientemente óptimos para este problema, debido a que se puede dar el caso de tener que agregar alguna estación de combustible y/o algún otro depósito, lo cual puede optimizar de mejor manera la solución obtenida. Finalmente, si bien en las soluciones se visitan a todos los clientes, éstas no son del todo óptimas debido a que se falla a la hora de explorar otros vecindarios. Además, este proyecto permitió la cercanía hacia problemas del tipo NP-completos y hacia la exploción combinatorial.

Referencias

[1] Mohammad Asghari and S. Mohammad J. Mirzapour Al-e-hashem. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, page 107899, 2020.

[2] Mauricio Blugeri, Simona Mancini, Ferdinando Pezzela, and Ornella Pisacane. A new mathematical programming model for the green vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 55:89–92, 2016.

[3] M. Bruglieri, S. Mancini, and O. Pisacane. The green vehicle routing problem with capacitated alternative fuel stations. *Computers Operations Research*, 112:104759, 2019.

[4] M. Bruglieri, S. Mancini, and O. Pisacane. More efficient formulations and valid inequalities for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, 105:283 – 296, 2019.

[5] Sevgi Erdoğan and Elise Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100 – 114, 2012. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

[6] Mohammad Reza Jabbarpour, Rafidah Md Noor, and Rashid Hafeez Khokhar. Green vehicle traffic routing system using ant-based algorithm. *Journal of Network and Computer Applications*, 58:294 – 308, 2015.

[7] Canhong Lin, K.L. Choy, G.T.S. Ho, S.H. Chung, and H.Y. Lam. Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, 41(4, Part 1):1118 – 1138, 2014.

[8] Reza Moghdani, Khodakaram Salimifard, Emrah Demir, and Abdelkader Benyettou. The green vehicle routing problem: A systematic literature review. *Journal of Cleaner Production*, 279:123691, 2021.

[9] Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza, and Juan G. Villegas. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C*, 70:113–128, 2016.

[10] Ángel Felipe, M. Teresa Ortuño, Giovanni Righini, and Gregorio Tirado. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111 – 128, 2014.