Inteligencia Artificial

Estado del Arte: Problema Green Vehicle Routing Problem (G-VRP)

[Víctor Antonio Martínez Campos]

20 de julio de 2022

Evaluación

Mejoras 2da Entrega (20 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (30 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100 %):	_____

Resumen

El problema G-VRP trata sobre encontrar las mejores rutas posibles para una flota de vehículos que funcionen en base a combustible alternativos que deban viajar atendiendo a un conjunto de clientes, en base al costo, tiempo y distancia total. Debido a la limitada autonomía de combustible, es posible que cada vehículo deba detenerse en estaciones de combustible alternativo (AFS) durante su viaje. En la presente entrega, se trata el el estado del arte del problema G-VRP: sus orígenes, variantes y distintos enfoques de solución. Se proponen tres algoritmos para la resolución del problema: Simmulated Anelaing más heurísticas (swap / 2-opt) y búsqueda Greedy. Se explica en detalle cada algoritmo y su representación. Por último se realizan experimentos numéricos y se concluye que SA más Swap es el que permite obtener las soluciones de mejor calidad. ***Palabras clave: G-VRP, heurísticas, metaheurísticas, algoritmos exactos***

1. Introducción

El problema G-VRP surge de la necesidad de incluir vehículos de combustible alternativo (AFV) en flotas de reparto y logística en Europa y USA debido a diversas políticas y regulaciones que buscan disminuir la contaminación generada por vehículos que funcionan en base a combustibles fósiles [4][3][1]. Los AFV presentan la desventaja de contar con baja autonomía de combustible (lo cual implica recorrer menores distancias que los vehículos convencionales)

y contar con una baja disponibilidad de estaciones de recarga de combustible, debido al bajo desarrollo en infraestructura de este tipo en ambos territorios [4][3]. La presente entrega propone una visión general del problema G-VRP, su estado de resolución y principales enfoques y su posible desarrollo futuro.La sección *2* trata sobre la definición del problema, se explica en que consiste y cuales son sus principales variantes. La sección *3* trata sobre el estado del arte del problema G-VRP: se explican sus orígenes y evolución, y se exponen los principales enfoques de solución existentes. La sección *4* muestra el modelo matemático del problema G-VRP, mientras que en la sección *5* se exponen los algoritmos de resolución, a saber, Simmulated Anealing más heurísticas (Swap / 2-opt), y también la representación de las estructuras usadas en la implementación de los mismos. En la sección *6* se explican los algoritmos utilizados. En la sección *7* se explican los experimentos numéricos llevados a cabo para comparar el desempeño de los algoritmos propuestos en términos de la calidad de la solución obtenida y el tiempo de ejecución. Por último, las secciones *8* y *9* se exponen los resultados obtenidos y las conclusiones.

2. Definición del Problema

Se tomará como referencia el modelo propuesto por Erdogan y Miller-Hooks (2012). Considere que una empresa tiene una flota de vehículos que funcionan en base a combustibles alternativos (AFV: alternative fuel vehicles) los cuales permiten transportar objetos desde un almacén central a cada uno de los clientes que se encuentran en la ciudad. Se forman entonces rutas de reparto, las cuales comienzan y terminan en el almacén y constan de una lista de clientes a visitar [4]. Una manera sencilla de representar este escenario es a través de un grafo, donde los nodos componentes corresponden al conjunto de clientes, estaciones de servicio y un único almacén central. Una de las posibles rutas que uno de estos vehículos podría tomar es representada por un grafo conexo, donde cada arco que conecta un par de nodos tiene asociado una distancia y un tiempo de viaje. Considere que todos los vehículos de la flota son iguales. Cada uno posee un tanque de combustible de capacidad limitada y además utiliza el combustible a una tasa de consumo fija al moverse por la ciudad a velocidad constante. Para poder llegar a todos los clientes durante una jornada de trabajo de tiempo limitado, cada vehículo debe recargar combustible en algunas de las escasas estaciones de servicio especiales disponibles en la ciudad, donde cada una de estas tiene una capacidad de combustible ilimitada. Cuando se atiende a un cliente o se recarga combustible, se considera que el vehículo está detenido y por lo tanto solo es posible contabilizar el tiempo allí invertido, esto último se conoce tiempo de servicio [9]. Cada cliente es atendido una única vez por un solo vehículo de la flota y cada vehículo se usa como máximo una vez por ruta. El nivel de combustible en el tanque del vehículo debe ser suficiente como para permitir visitar cualquier par de nodos. Como la jornada diaria de trabajo tiene un tiempo limitado, la duración de la ruta asignada a un vehículo no puede superar este umbral de tiempo máximo [9]. En el modelo planteado por Erdogan y Miller-Hooks (2012) se consideran como variables las distancias y tiempos de viaje, además de una variable que permite la elección o no de un determinado arco. El modelo es capaz de compensar entre niveles altos de combustible y viajes cortos[1]. Existen también algunos parámetros como la tasa de consumo de combustible, capacidad de combustible del tanque y tiempos de servicio asociado a cada nodo (carga de combustible y atención del cliente) [4]. Las restricciones de autonomía de combustible y tiempo límite de trabajo son las más notables para este problema [4], además del resto de restricciones que ya fueron nombradas. Estás serán vistas en profundidad en la sección *4*.

El objetivo del problema es determinar el conjunto de rutas óptimas que debe seguir la flota de vehículos de reparto para minimizar el costo total. Esto le permite a la empresa ahorrar dinero y tiempo [4][9]. El problema G-VRP es NP-hard [4], lo cual indica que el esfuerzo computacional requerido para resolver una instancia del problema crece exponencialmente a medida que esta aumenta de tamaño. Lo anterior tiene como consecuencia que la solución exacta a problemas

reales de gran tamaño sea difícil de obtener.

2.1. Variantes del problema

El problema G-VRP tiene múltiples variaciones las cuales están estrechamente relacionadas entre si. En base al número de publicaciones entre 2006 y 2019 las variantes más estudiadas son: The pollution routing problem (PRP), The green heterogeneous vehicle routing problem (GHRP), Energy-minimizing vehicle routing problem (EMVRP), Time-dependent vehicle routing problem (TDVRP), Fuel consumption in VRPs (FCRVP) y Electric vehicle routing problem (EVRP) [6]. Estos problemas y G-VRP tienen en común en que todos son variaciones del problema VRP con enfoque medioambiental y operativo[6]. En esta entrega se presentarán dos de las más conocidas: PRP y EVRP.

Pollution routing problem: El PRP es una extensión del problema VRP que considera no solo la distancia de viaje, sino que también la emisión de gases de efecto invernadero, autonomía de combustible, tiempos de viaje y costos asociados [2]. El PRP se representa a través de un grafo completo, donde los nodos se refieren al conjunto de clientes y a un único almacén central o depósito. Se considera que la flota de vehículos es homogénea, donde cada uno tiene la misma capacidad de carga. Cada cliente debe ser atendido dentro de un intervalo de tiempo específico. Este modelo contabiliza la distancia y tiempo de viaje entre clientes, lo cual es representado a través de arcos con peso. Cada vehículo emite una cierta cantidad de gases de efecto invernadero al viajar entre nodos, los cuales dependen principalmente de la carga y velocidad del vehículo, esto en conveniente ya que ambos factores pueden ser controlados [2]. En la formulación del modelo se considera el costo de emisión, costos operacionales y costo de choferes. El costo de emisión refleja la cantidad de gases de efecto invernadero emitidos por uno de los vehículos. No se realiza una medición directa de esta emisión, sino que se plantea un modelo que relaciona la cantidad de energía requerida con la cantidad de combustible a consumir y esta última se traduce en emisiones de gases de efecto invernadero (GHG). La cantidad de energía requerida depende directamente de la carga del vehículo y su velocidad. En los costos operacionales se contabiliza el costo de combustible, este último es estimado usando el modelo asociado al costo de emisión (función de energía). El costo de choferes se asocia al tiempo de viaje y ubicaciones de los clientes [2].

El objetivo del problema es construir rutas de reparto que minimicen el costo total, el cual está compuesto por el costo de emisión, costos operacionales y costo de choferes [2]. Bektas y Laporte (2011) plantean en principio un modelo no lineal como consecuencia de la función de energía y el cálculo del costo de choferes, que luego vuelven lineal discretizando las velocidades de conducción. En este modelo se consideran como variables la distancia y tiempo de viaje entre nodos, costos de emisión, costos operacionales y costos de choferes, velocidad de manejo, demanda de cada cliente y la carga que cada vehículo lleva (considerando el peso del vehículo). Además, el problema está sujeto a las siguientes restricciones: Cada cliente puede ser visitado solo una vez (impide la existencia de subtours), la carga que cada vehículo puede llevar está sujeta a una capacidad máxima, la ventana de tiempo en que puede ser atendido cada cliente es finita y existe una velocidad mínima de manejo de 40(km/h). También se considera que las rutas deben comenzar y terminar en el almacén central [2]. Este problema se diferencia de G-VRP en que en PRP no se consideran puntos de recarga de combustible y si se consideran otros factores como por ejemplo velocidades variables y costos de emisión.

Electric vehicle routing problem: El EVRP encuentra las rutas óptimas de reparto con el mínimo costo de tiempo y energía, así como el número de vehículos eléctricos despachados. Este modelo considera el efecto de la carga de transporte del vehículo en el consumo de batería [5]. Los vehículos eléctricos son más eficientes en relación al consumo de energía que los vehículos que funcionan en base a petróleo, pero presentan una autonomía menor (40 a 100 millas) antes de tener que ser recargados [5]. Esto motiva la búsqueda de rutas óptimas que cuenten con estaciones de carga entre clientes durante la operación diaria. Lin, Zhou y Wolfson (2016) presentan un

modelo no-linear basado en el problema VRP el cual es representado por un grafo cuyos nodos representan al conjunto de clientes, estaciones de recarga y un único almacén central. Cada arco está asociado con un tiempo y distancia de viaje. La velocidad de viaje se asume constante sobre un arco. Al igual que en G-VRP, los vehículos comienzan con el nivel de batería completo, la cual se va agotando a medida que transitan por la ciudad. Se consideran algunos supuestos o condiciones: Existe un único almacén central donde las rutas comienzan y terminan, la tasa de recarga de batería es constante, la batería se recarga completamente al visitar una estación de recarga, el tiempo límite de trabajo por día es de 8 horas, no se cuantifica el tiempo de inactividad en un arco o parada (estación de recarga o cliente), no se considera una ventana de tiempo asociada a la atención de clientes. El modelo propuesto considera como variables de decisión la carga remanente de batería, carga de transporte, instante de llegada a un nodo y una variable que permite tomar o no un arco en particular [5]. Además, como parámetros del modelo se considera el costo de viaje como función del tiempo de viaje, costo de carga de la batería como función del consumo de energía, tiempo de recarga de batería, tiempo y distancia de viaje, velocidad de viaje, entre otras. A pesar de que en principio el modelo es no-lineal, Lin, Zhou y Wolfson (2016) proponen una linealización que permite resolver el problema como uno de variable entera mixta.

3. Estado del Arte

El primer modelo de G-VRP fue presentado por Erdoğan y Miller-Hooks (2012), este se diferencia del problema VRP en que el primero considera restricciones de combustible y estaciones de recarga (número y distribución) [1]. Esto tiene como consecuencia de que la autonomía de un vehículo y por ende la distancia máxima alcanzada por este, puede ser extendida al pasar por una estación de recarga[1]. En consecuencia, resulta clave para la adopción de dichos vehículos en flotas de reparto y logística contar con rutas predefinidas, que permitan a los vehículos cumplir con los objetivos de reparto sin quedar varados en la ruta producto de la falta de combustible, cumpliendo además con las restricciones de tiempo y costo [3].

El problema G-VRP ha tenido un aumento marcado en el número de publicaciones asociadas a nuevos enfoques de resolución [6], entre las cuales se encuentran: heurísticas, metaheurísticas, algoritmos exactos, solver comerciales, e híbridos [6]. Para los algoritmos metaheurísticos, los métodos más significativos empleados son: Algoritmos genéticos (GA), Simulated annealing, TS, ant colony optimization, artificial bee colony, greedy randomized adaptive search procedure, entre otros [6]. Para el uso de métodos exactos destacan Flagship algorithms, branch and cut, branch and price, entre otros. Entre los métodos heurísticos más usados se encuentran: the saving procedure, insertion procedure,improvement procedure y the clusterfirst route-second procedure, usado por Erdogan y Miller-Hooks (2012). Los métodos híbridos incluyen distintas combinaciones: metaheurísticas-exactas, metaheurísticas-metaheurísticas y heurísticas-heurísticas, con el objetivo de obtener mejores resultados [6]. Los algoritmos basados en metaheurísticas son los más utilizados para resolver el problema G-VRP y sus variantes, seguido por los métodos exactos, híbridos, solver y heurísticas [6]. Aproximadamente el 60 % de las publicaciones usan un enfoque de resolución aproximado (metaheurística, heurística, e híbrido) y el 32 % un enfoque exacto (algoritmos exactos y solver) [6].

En esta entrega se analizan los resultados obtenidos en 6 publicaciones diferentes respecto al tratamiento del problema G-VRP, desde el año 2012 al 2019. Erdoğan y Miller-Hooks(2012), los primeros en presentar el problema, proponen un enfoque de solución a través de heurísticas: 'modified Clarke and Wright savings algorithm (MCWS)' y 'Density-Based Clustering Algorithm (DBCA)', junto a un proceso de post-optimización. Schneider et.al (2014), resuelven el problema EVRPTW (electric vehicle-routing problem with time windows and recharging stations), el cual es una extensión al problema G-VRP, utilizando una metaheurística basada en la combinación de VNS (Variable neighborhood search), TS (Tabu search algorithm) y SA (simulated annealing),

la cual denominan VNS/TS. Montoya et.al (2016) proponen la heurística MSH (multi-space sampling heuristic) para resolver el problema G-VRP. Koc y Karaoglan(2016) proponen un enfoque de solución basado en la combinación de la *heurística de simmulated annealing* (SA) junto al *algoritmo branch and cut*(B&C) el cual permite obtener la solución exacta. Andelmin et.al (2017) proponen un algoritmo exacto (EA) como enfoque de solución. Bruglieri et.al (2019) proponen un enfoque de solución exacta Path Based Approach (PBA), el cual, para resolver instancias grandes del problema, es modificado utilizando la heurística MSH. Las publicaciones aquí tratadas utilizan las instancias de tamaño pequeño y grande generadas por Erdoğan y Miller-Hooks(2012) de manera integra o ligeramente modificada, además de otras instancias en algunos casos. Esto permite generar una comparativa aproximada entre los distintos métodos en relación al tiempo de CPU (o tiempo de computación) y número de soluciones óptimas/calidad de la solución encontrada. Aunque en algunas de estas publicaciones se generan comparativas directas entre métodos, existen casos donde esto no ocurre. A continuación se presenta un resumen general de cada publicación, donde se expone un breve extracto de la propuesta (modelo y enfoque de solución) y resultados de los experimentos numéricos realizados.

Erdogan y Miller-Hooks (2012) formulan un modelo de programación lineal entera mixta que toma en consideración restricciones de autonomía de combustible y el número y distribución de estaciones de recarga de combustible alternativo. Presentan como estrategia de solución las heurísticas modified Clarke and Wright savings algorithm (MCWS) y Density-Based Clustering Algorithm (DBCA), junto a un proceso de post-optimización que busca mejorar la calidad de la solución encontrada[3]. MCWS genera un conjunto de rutas posibles, las cuales son mezcladas entre si hasta obtener un conjunto solución[4]. DBCA genera un conjunto de rutas posibles explotando las propiedades espaciales de G-VRP, descomponiendo el VRP en dos subproblemas: clustering y ruteo, donde el número y distribución relativa de clientes y AFS afecta significativamente la factibilidad y el número requerido de AFS visitados[4]. Erdogan y Miller-Hooks (2012) realizan experimentos numéricos sobre instancias pequeñas del problema a través de 4 escenarios distintos, donde el primero presenta el impacto de una distribución uniforme de clientes, el segundo presenta el impacto de distintos agrupamientos de clientes, el tercero muestra el impacto de distintas configuraciones espaciales de los AFS y el último muestra el impacto de la densidad de AFS. En general, para cada escenario los resultados de ambas heurísticas son muy similares, pero siempre que existe una diferencia en la solución obtenida, el DBCA encuentra la mejor solución. Agregan que la similitud en las soluciones obtenidas puede ser causa del tamaño pequeño de las instancias[4]. También se evalúa el desempeño de ambas heurísticas con una instancia más realista, la cual se genera utilizando la ubicación de depósito de una empresa de suministros de textiles médicos en Virginia y un grupo de clientes basado en ubicaciones de hospitales en Virginia, Maryland y el Distrito de Columbia. Las estaciones de servicio corresponden a estaciones de biodiesel reales ubicadas en la región. En esta última se obtienen resultados muy similares con ambos métodos[4]. Con el fin de evitar posibles subciclos generados por el uso de puntos de recarga de combustible (AFS), se utilizan nodos dummy los cuales son clones de los nodos AFS[3]. El problema de este enfoque, es que el tamaño del grafo puede aumentar considerablemente debido al número de *nodos dummy* utilizados. En el peor caso, el número de nodos dummy para cada visita a una AFS puede ser igual al número de clientes. Esto implica un mayor tiempo de computación y memoria para instancias grandes del problema, ya que el número de variables binarias en la formulación aumenta con el número adicional de nodos dummy[9].

Schneider, Stenger y Goeke (2014), extienden el problema de G-VRP incluyendo ventanas de tiempo de atención de clientes y restricciones en la capacidad de carga de los vehículos, aunque ignoran la restricción de tiempo máximo para cada ruta (duración de la jornada de trabajo)[1]. Resuelven el problema de EVRPTW (electric vehicle-routing problem with time windows and recharging stations) utilizando una metaheurística basada en la combinación de VNS (Variable neighborhood search), TS (Tabu search algorithm) y SA (simulated annealing), la cual realiza una búsqueda local en vecindarios cada vez más grandes para explorar eficientemente el espacio de solución y evitar quedarse atascado en óptimos locales[8]. Se realizan tres

experimentos numéricos, el primero busca medir el desempeño de la metaheurística VNS/TS propuesta, el segundo busca medir el desempeño de los algoritmos componentes VNS, TS y SA en instancias de tamaño medio, y el tercero busca demostrar el desempeño de la metaheurística en instancias de problemas relacionados, entre ellos G-VRP [8]. En el primer experimento, para evaluar la calidad de los resultados se usan pequeñas instancias que pueden ser resueltas por medio del solucionador comercial CPLEX, los resultados muestran que la metaheurística resulta ser más eficiente y da mejores resultados que al utilizar CPLEX [8]. Los resultados del segundo experimento muestran que existe un efecto positivo al mezclar VNS,TS y SA en relación a la calidad de la solución e incluso los tiempos de computación con respecto a solo usar TS. En el tercer experimento se usan las instancias de referencia de G-VRP propuestas por Erdogan y Miller-Hooks (2012) y se comparan con los resultados obtenidos mediante MCWS y DBCA. Se obtiene que la metaheurística VNS/TS supera a ambos métodos tanto en las instancias pequeñas (4 escenarios nombrados anteriormente) como en la instancia realista de mayor tamaño. VNS/TS reduce el número de vehículos necesarios en casi la mitad de las instancias pequeñas, y lo logra en menos de 40[s] promedio de computación. Para la instancia realista, la distancia de viaje de las soluciones se redujo en aproximadamente un 15 %, además de requerir un número significativamente menor de vehículos [8].

Koc y Karaoglan (2016), plantean un nuevo modelo de programación lineal mixta para el problema G-VRP. Este se diferencia del modelo propuesto por Erdogan y Miller-Hooks (2012) en que no se utilizan nodos 'dummy' para representar las múltiples visitas realizadas a una AFS, y por ende evitan aumentar el tamaño del grafo, con lo cual esperan conseguir un mejor rendimiento en la resolución del problema. Plantean un nuevo enfoque de solución basada en la combinación de la *heurística de simmulated annealing* (SA) junto al *algoritmo branch and cut*(B&C). Esta combinación permite resolver el problema de manera óptima buscando la solución exacta[9]. En el modelo de Koc y Karaoglan (2016), el número de variables binarias es $O(|N_c|^2|F|)$, el número de continuas $O(|N_c|)$ y el número restricciones $O(|N_c|^2|F|)$, mientras que en el modelo de Erdogan y Miller-Hooks (2012), el número de variables binarias es $O(|N_c|^2|F|^2)$, el número de continuas $O(|N_c||F|)$ y el de restricciones $O(|N_c|^2|F|^2)$, lo cual demuestra el efecto de no utilizar nodos dummy[9] ($|N_c|$ es el número de clientes y $|F|$ el número de AFS). Para los experimentos numéricos utilizan el set de datos generados por Erdogan y Miller-Hooks (2012) con pequeñas variaciones. En estos buscan comparar distintos enfoques de resolución en términos de tiempo computacional y error relativo óptimo asociado a las soluciones de cada instancia. En el primer experimento comparan el error relativo porcentual Δ del valor obtenido para la función objetivo usando como valor de referencia los generados por el solver IBM ILOG CPLEX 12.5. Se usa la expresión $\Delta = 100 \cdot \frac{z^* - z^{LP}}{z^*}$, donde z^{LP} es el valor obtenido por el solver y z^* el valor obtenido por Erdogan y Miller-Hooks(2012) o B&C propuesto por Koc y Karaoglan (2016) según corresponda. Esto permite generar valores de Δ comparables. Los resultados muestran que para todas las instancias, el algoritmo de B&C presenta mejores resultados[9]. En el segundo experimento, se investigan los efectos de aplicar heurísticas (heurística de Clarke y Writght extendida (ECW) y SA) para encontrar la solución. Se utiliza la expresión del error relativo $\Delta = 100 \cdot \frac{z_{heur} - UB}{z_{heur}}$, donde z_{heur} es el valor calculado por la heurística (SA o ECW) y UB la solución óptima o mejor valor encontrado. Los resultados muestran que aunque ECW encuentra soluciones bastante alejadas de las óptimas, la heurística SA aplicada a lo obtenido por ECW mejora considerablemente los resultados en un tiempo de computación corto[9]. En el tercer experimento se analiza el desempeño del algoritmo B&C propuesto en función del tiempo de computación, instancias óptimamente resueltas y la calidad de varios componentes relacionados[9]. Indican que la heurística SA mejora la solución obtenida por B&C en un tiempo corto de computación, permitiéndole a esta encontrar soluciones de alta calidad.

Montoya et al.(2016) proponen la heurística MSH (multi-space sampling heuristic) la cual muestrea diferentes espacios de representación de soluciones y luego ensambla una solución con (partes de) los elementos muestreados [7]. El algoritmo se construye a partir de dos componentes principales: un conjunto de funciones de muestreo y un procedimiento de ensamblaje. Las

funciones de muestreo son heurísticas aleatorias de first-route, cluster-second. Utilizando esta heurística, MSH extrae una muestra del espacio de representación de la solución TSP y extrae de ella una muestra del espacio de representación de las rutas. Posteriormente, MSH utiliza las rutas muestreadas para ensamblar una solución final. El procedimiento de ensamblaje es un modelo de partición de conjuntos que se ejecuta sobre el conjunto de rutas muestreadas en la primera fase [7]. Para los experimentos numéricos se utiliza el set de datos generado por Erdogan y Miller-Hooks (2012). Los resultados son comparados con los mejores resultados obtenidos mediante MCWS/DBCA, y VNS/TS, más otros algoritmos que no se consideran en la presente entrega. El tiempo promedio de computación de MSH es mejor que el de VNS/TS para las instancias pequeñas, aunque se aproximan bastante (mismo orden de magnitud). Para las instancias más grandes, el tiempo promedio de computación de MSH es mejor que el de VNS/TS, y presentan una diferencia de un orden de magnitud. El tiempo no está registrado para MCWS/DBCA. En términos de calidad de solución, MSH supera a VNS/TS y MCWS/DBCA, para instancias de tamaño pequeño y grande. Además, MSH es uno de los métodos más simples para resolver G-VRP[7].

Andelmin y Bartolini (2017) proponen un algoritmo exacto (EA) como enfoque de solución, basado en una formulación de partición de conjuntos en la que las columnas corresponden a rutas factibles[1]. Se modela el problema utilizando un multigrafo que no requiere explícitamente modelar las AFS y en el que los nodos corresponden a clientes y cada arco representa una posible secuencia de estaciones de servicio visitadas por un vehículo cuando viaja de un cliente a otro. Se refuerza la formulación agregando tres clases de desigualdades válidas: desigualdades de fila de subconjunto débil, desigualdades de fila de subconjunto y cortes de ruta k. Se demuestra que un subconjunto de los cortes de ruta k corresponden a cortes de Chvátal-Gomory (CG) de rango uno. Usan esta observación para formular el problema de separación correspondiente como un problema de programación lineal de enteros mixtos (MILP) y se describe un método para incorporar tales cortes dentro de un algoritmo de generación de corte y columna[1]. Consideran dos conjuntos de instancias para llevar a cabo los experimentos numéricos. El primer conjunto corresponde a los cuatro escenarios propuestos por Erdogan y Miller-Hooks (2012) más el escenario realista (aunque consideran solo las instancias con un máximo de 111 clientes). El segundo conjunto se crea a partir de las instancias más grandes del primer conjunto[1]. Los resultados numéricos muestran que el enfoque de solución exacta propuesto es adecuado para resolver de manera óptima algunas instancias de referencia de gran tamaño en un tiempo computacional promedio de 3 horas[3].

Bruglieri, Mancini, Pezzella, Pisacane (2019) proponen un enfoque de solución exacta: Path Based Approach (PBA). Este método consta de dos etapas: en la primera se generan todos los caminos factibles y no dominados que conectan dos nodos en $F \cup v_0$ (considera todas las estaciones de recarga más el almacén central). En la segunda etapa, algunos de los caminos obtenidos se seleccionan y combinan en una secuencia determinada a través de una formulación MILP. En tal formulación, el objetivo consiste en minimizar la distancia total de viaje mientras se cumplen las restricciones de duración máxima de ruta y de que todos los clientes sean atendidos exactamente una vez[3]. Las secuencias de caminos generados en esta etapa permiten componer las rutas de la solución. Este método de solución exacta, puede ser transformado en un enfoque de solución heurístico considerando en la primera etapa solo un subconjunto limitado de todas las rutas factibles. Esto permite superar los incrementos exponenciales de caminos y abordar en un tiempo razonable también instancias de gran tamaño [3]. Se genera el conjunto de rutas proporcionadas al modelo MILP de la segunda fase aplicando la primera etapa de la heurística MSH (Multispace Sampling Heurístics) debido a su alta eficiencia[3]. Realizan dos experimentos numéricos: el primero utiliza los cuatro escenarios de 10 instancias cada uno propuesto por Erdogan y Miller-Hooks (2012) para medir el desempeño del enfoque de solución exacta PBA y el segundo, tiene como objetivo medir el desempeño del enfoque heurístico (PBAH) en instancias de gran tamaño, para el cual utilizan el mismo conjunto de datos usado por Andelmin y Bartolini (2017) [3]. Realizan comparaciones respecto a los resultados obtenidos por Erdogan y Miller-

Hooks(2012) usando MCWS y DBCA, Koc y Karaoglan (2016) usando B&C y Andelmin y Bartolini (2017) usando un algoritmo exacto (EA). También se comparan con los resultados obtenidos a través de otros métodos, los cuales no se consideran en la presente entrega. Los experimentos muestran que PBA y EA son los únicos métodos que permiten obtener soluciones óptimas para todas las instancias de los 4 escenarios, por otro lado, B&C obtiene cerca del 53 % de las soluciones óptimas posibles, mientras que usando MCWS y DBCA alcanzan el tiempo de ejecución máximo de 1[hr] sin obtener soluciones óptimas[3]. Además, PBA tiene el tiempo de ejecución más corto para cada instancia, seguido por el método de EA. Ambos presentan una diferencia de dos ordenes de magnitud respecto al tiempo de ejecución de B&C. En el segundo experimento, PBAH supera a la heurística más efectiva para el G-VRP (MSH) propuesta por Montoya (2016) en términos de calidad de solución[3]. Se utiliza el enfoque exacto propuesto por Andelmin y Bartolini (2017) para calcular los valores óptimos de todas las instancias del experimento. PBAH y MSH no encuentran los valores óptimos de solución para ninguna de las instancias, y al comparar lo obtenido con el enfoque exacto, se observa que el error obtenido por PBAH es menor al obtenido mediante MSH: 2,3 % y 7,2 % respectivamente. En cuanto al tiempo de cálculo, PBAH es más eficiente que el enfoque exacto, requiriendo en promedio solo unos 20 minutos frente a las casi 2 horas y media que requiere el enfoque exacto (aunque MSH sigue siendo el enfoque más rápido)[3].

No todos los enfoques de solución son comparados directamente. Se tiene que en términos de tiempo de CPU, el orden de mayor a menor en Bruglieri et.al (2019) es MCWS/DBCA > B&C > EA > PBA, para instancias pequeñas del problema y EA > PBAH > MSH para instancias grandes. Mientras que en Montoya et al.(2016) el orden es MCWS/DBCA > VNS/TS > MSH. En relación al número de soluciones óptimas/calidad de la solución, el orden de menor a mayor en Bruglieri et.al (2019) es MCWS/DBCA < B&C < EA = PBA, para instancias pequeñas y MSH < PBAH < EA para instancias grandes. Mientras que en Montoya et al.(2016) el orden es MCWS/DBCA < VNS/TS < MSH.

4. Modelo Matemático

Erdoğan y Miller-Hooks (2012) formulan G-VRP como un problema de programación lineal entera mixta (MILP) cuyo modelo es el que se presenta a continuación:

Notación del grafo:

- A: Almacén o depósito central, $A = \{v_0\}$
- I: Conjunto de clientes, $I = \{v_1, v_2, ..., v_n\}$
- F: Conjunto de estaciones de servicio de combustible alternativo (AFS's), $F = \{v_{n+1}, v_{n+2}, ..., v_{n+s}\}$
- $G = (V, E)$: Grafo completo no dirigido, donde $V = \{v_0\} \cup I \cup F = \{v_0, v_1, ..., v_{n+s}\}$, $|V| = 1 + n + s$ y E el conjunto de arcos, $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$.

La formulación del modelo distingue entre visitar AFS y visitar clientes, esto es porque los clientes deben ser visitados una única vez, mientras que las AFS pueden ser visitadas múltiples veces [4]. Para evitar subciclos y al mismo tiempo permitir múltiples o cero visitas a un subconjunto de nodos (AFS en este caso), se genera un grafo aumentado $G' = (V', E')$ que considera nodos dummy $\Phi = \{v_{n+s+1}, v_{n+s+2}, ..., v_{n+s+s'}, \}$, uno por cada posible visita a un AFS o al almacén sirviendo como AFS. El número de nodos dummy asociado a cada AFS, n_f, se establece en el número de veces en que el AFS asociado, v_f, puede ser visitado [4]. De esta manera, $v_f \in F$ tiene n_f nodos dummy para $f = 0, ..., n + s$. Note que n_f debe ser establecido con el valor más pequeño posible para reducir el tamaño de la red, pero lo suficientemente grande como para no

restringir sus múltiples beneficios [4].

Notación grafo aumentado:

- $V' = V \cup \Phi$
- $F' = R \cup \Phi$
- I_0 Conjunto de vértices y almacén, $I_0 = v_0 \cup I$
- F_0 Conjunto de vértices y almacén, $F_0 = v_0 \cup F'$

Notaciones de variables y parámetros:

- p_i Tiempo de servicio en el nodo i (tiempo de recarga de combustible o de atención a un cliente). Si $i \in F$ se asume un tiempo constante.
- r Tasa de consumo de combustible
- Q Capacidad del tanque de combustible
- x_{ij} Variable binaria. Toma el valor 1 si el vehículo viaja del nodo i al j, 0 en otro caso.
- y_j Remanente de combustible en el tanque cuando el vehículo está en el nodo j. Toma el valor Q cada vez que pasa por un AFS o almacén central.
- τ_j Variable de tiempo que especifica el instante de llegada de un vehículo al nodo j. Se inicializa en cero a la salida del almacén.

El almacén central puede ser usado para llenar combustible y todas la estaciones de combustible alternativos tienen capacidad ilimitada. Cada arco (v_i, v_j) está asociado con valores no negativos de tiempo de viaje t_{ij}, costo c_{ij} y distancia d_{ij}. Las velocidades de viaje se asumen constantes sobre un arco. No existe límite en el número de recargas de combustible. Cuando se recarga combustible, se asume que se llena a capacidad máxima [4].

El G-VRP busca encontrar a lo sumo m rutas, una para cada vehículo, las cuales comienzan y terminan en el almacén o depósito, y en donde se visitan clientes y AFS cuando se requiera, de manera que la distancia total viajada sea la mínima posible [4].

Función Objetivo:

$$\min \sum_{i,j \in V', i \neq j} d_{ij} x_{ij} \tag{1}$$

Sujeta a las siguientes restricciones:

$$\sum_{j \in V', j \neq i} x_{ij} = 1, \forall i \in I \tag{2}$$

$$\sum_{j \in V', j \neq i} x_{ij} \leq 1, \forall i \in F_0 \tag{3}$$

$$\sum_{i \in V', j \neq i} x_{ji} - \sum_{i \in V', j \neq i} x_{ij} = 0, \forall j \in V' \tag{4}$$

$$\sum_{j \in V' \setminus \{0\}} x_{0j} \leq m \tag{5}$$

$$\sum_{j \in V' \setminus \{0\}} x_{j0} \leq m \tag{6}$$

$$\tau_j \geq \tau_i + (t_{ij} - p_j)x_{ij} - T_{\max}(1 - x_{ij}) \tag{7}$$

$$0 \leq \tau_0 \leq T_{\max} \tag{8}$$

$$t_{0j} \leq \tau_j \leq T_{\max} - (t_{j0} + p_j), \forall j \in V' \setminus \{0\} \tag{9}$$

$$y_j \leq y_i - r \cdot d_{ij} x_{ij} + Q(1 - x_{ij}), \forall j \in I \text{ e } i \in V', i \neq j \tag{10}$$

$$y_j = Q, \forall j \in F_0 \tag{11}$$

$$y_j \geq \min \{r \cdot d_{j0}, r \cdot (d_{jl} + d_{l0})\}, \forall j \in I, \forall l \in F' \tag{12}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \tag{13}$$

La función objetivo (1) busca minimizar la distancia total viajada por la flota de vehículos un día dado. La restricción (2) busca que exista una única salida por nodo cliente o almacén. La restricción (3) busca que exista una única salida por nodo AFS (y sus dummies asociados). La restricción (4) asegura que todos los nodos tengan el mismo número de salidas y entradas, esto se conoce como conservación de flujo. La restricción (5) indica que a lo más m vehículos salen del almacén. La restricción (6) asegura que a lo más m vehículos retornan al almacén al finalizar un día de trabajo. Se genera una copia del almacén para distinguir instantes de salida y de llegada en el almacén, lo cual es necesario para llevar un registro del tiempo en cada nodo visitado y prevenir la formación de subtours. El tiempo de llegada de un vehículo a un nodo es registrada a través de la restricción (7). Las restricciones (7), (8) y (9) aseguran que la duración de un tour para cada vehículo no sea mayor a $T_{\max}$. La restricción (8) especifica que el instante de salida del almacén es cero ($\tau_0 = 0$) y define una cota superior en relación al instante de llegada al almacén de $T_{\max}$. Las cotas superiores e inferiores de tiempo de llegada dadas en la restricción (9) para los nodos de tipo cliente y AFS, aseguran que cada ruta es completada en un tiempo no superior a $T_{\max}$. La restricción (10) registra el nivel de combustible del tanque en función de la secuencia y el tipo de nodos visitados. Si el nodo j es visitado inmediatamente después del nodo i ($x_{ij} = 1$), y el nodo i es de tipo cliente, el primer término de la restricción (10) reduce el nivel de combustible al llegar al nodo j basado en la distancia viajada desde i (d_{ij}) y la tasa de consumo de combustible r. Las restricciones (7) y (10) sirven para eliminar la formación de subtours. La restricción (11) sirve para rellenar el tanque de combustible al máximo Q al llegar a un nodo AFS o almacén. La restricción (12) garantiza de que el nivel de combustible remanente del tanque sea suficiente como para volver al almacén, ya sea directamente o pasando a través de un nodo AFS, desde cualquier nodo cliente presente en la ruta. Esta restricción busca asegurar que los vehículos no queden varados a medio camino. Es posible extender esta restricción para permitir rutas de retorno que visiten más de un AFS. La restricción (13) asegura que la variable de decisión x_{ij} es de naturaleza binaria [4].

5. Representación

En esta entrega se propone la resolución del problema GVRP a través de tres mecanismos: Solo usando búsqueda Greedy, Simmulated anealing más heurística swap y Simmulated anealing más heurística 2-opt. Se utiliza la siguiente representación:

5.1. Estructuras

5.1.1. Objetos

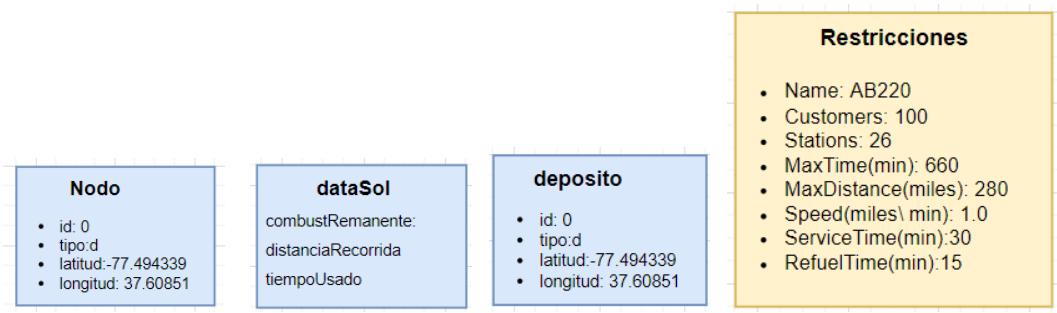
- nodo: El depósito central, así como cada cliente y estación de servicio, se representan a través de un objeto nodo, el cual contiene un id único, tipo, y ubicación (latitud, longitud).
- dataSol: Objeto que contiene información sobre la cantidad de combustible remanente, distancia recorrida y tiempo usado, en la solución.

5.1.2. Arreglos

- solución: Arreglo de una dimensión el cual contiene objetos nodo. Representa la ruta recorrida por un vehículo.
- listaClientes o clientesVisitables: Arreglo de una dimensión el cual contiene nodos tipo cliente.
- listaEstaciones: Arreglo de una dimensión el cual contiene nodos tipo fuel.
- restricciones: Estructura u objeto que contiene toda la información que caracteriza la instancia del problema.
- conjuntoSoluciones: Arreglo de dos dimensiones que contiene arreglos solución. Representa la solución a la instancia del problema, contiene todas las rutas seguidas por cada vehículo.
- conjuntoDataSoluciones: Arreglo de una dimensión que contiene objetos dataSol. Permite generar junto con conjuntoSoluciones toda la información de la solución final para una instancia del problema. La ruta en conjuntoSoluciones[i] está asociada a la información de conjuntoDataSoluciones[i].

Las Figuras 1 y 2 presentan esquemas explicativos de los objetos y arreglos usados. Un posible problema de esta representación es que es poco eficiente con el uso de memoria, ya que los arreglos contienen información que no siempre es utilizada. Por ejemplo, en el cálculo de distancia recorrida, solo basta conocer las coordenadas de los nodos contenidos en el arreglo, pero no su id y tipo. Aun así, esta representación resulta sencilla de implementar, ya que facilita la abstracción del problema. Por otro lado, la representación elegida no reduce el espacio de búsqueda, pero si es posible construir soluciones factibles e infactibles. Además, es posible representar la solución óptima para cada instancia del problema.

Al implementar el algoritmo de búsqueda Greedy, está representación es conveniente ya que facilita la comparación entre nodos. En el caso del algoritmo Simmulated annealing más heurísticas, por ejemplo 2-opt, ocurre un intercambio de nodos y luego un reordenamiento del arreglo, esta representación facilita la manipulación de los datos sin perder información en el proceso y evita errores de confusión en su implementación. La figura 3, presenta un ejemplo del uso de la representación en la slución final para la instancia AB220 usando el algoritmo SA con 2-opt, en la subfigura 3a se presenta el uso de los arreglos conjuntoSoluciones y conjuntoDataSolución, además de mostrar los resultados obtenidos al aplicar el algoritmo (calidad de la solución, número de clientes atendidos, etc). En la subfigura 3b, se muestra la ruta seguida por uno de los vehículos haciendo uso del arreglo solución, junto con la distancia recorrida, tiempo transcurrido y distancia excedida.



(a) Nodo (b) dataSol (c) depósito (d) restricciones de la instancia

Figura 1: Objetos utilizados en la representación del problema

listaClientes	Nodo	Nodo	....	Nodo	Nodo
	id: 1 tipo:c latitud:-76.076059 longitud: 39.068651	id: 2 tipo:c latitud:-76.025496 longitud: 39.031864		id: 99 tipo:c latitud:-79.597331 longitud: 39.016513	id: 100 tipo:d latitud: -77.631801 longitud: 36.684050

(a) listaClientes

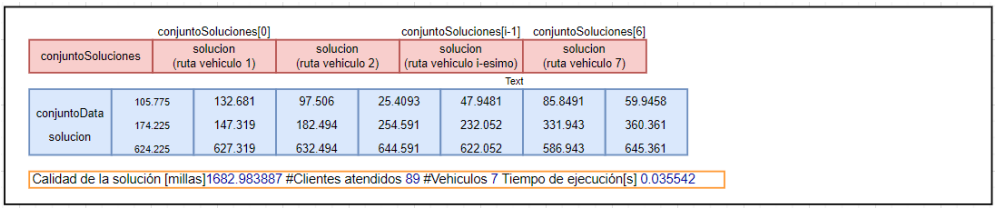
listaEstaciones	Nodo	Nodo	....	Nodo	Nodo
	id: 1 tipo:f latitud: -77.612242 longitud: 37.659873	id: 2 tipo:f latitud: -77.064593 longitud: 38.869453		id: 24 tipo: f latitud: -78.591852 longitud: 38.914102	id: 25 tipo: f latitud: -80.102566 longitud: 37.673553

(b) listaEstaciones

conjunto DataSolucion	dataSol	dataSol	....	dataSol	dataSol
	combustRemanente: distanciaRecorrida tiempoUsado	combustRemanente: distanciaRecorrida tiempoUsado		combustRemanente: distanciaRecorrida tiempoUsado	combustRemanente: distanciaRecorrida tiempoUsado

(c) conjuntoDataSolucion

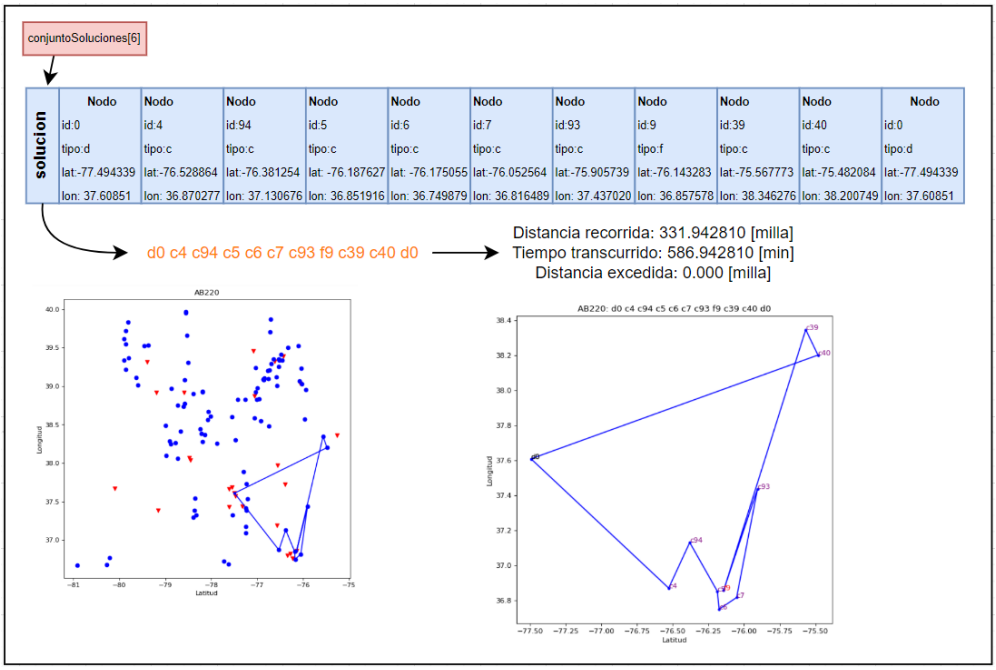
Figura 2: Arreglos utilizados en la representación del problema



	conjuntoSoluciones[0]			conjuntoSoluciones[i-1]	conjuntoSoluciones[6]	
conjuntoSoluciones	solucion (ruta vehiculo 1)	solucion (ruta vehiculo 2)		solucion (ruta vehiculo i-esimo)	solucion (ruta vehiculo 7)	
				Text		
conjuntoData solucion	105.775	132.681	97.506	25.4093	47.9481	85.8491	59.9458
	174.225	147.319	182.494	254.591	232.052	331.943	360.361
	624.225	627.319	632.494	644.591	622.052	586.943	645.361

Calidad de la solución [millas]1682.983887 #Clientes atendidos 89 #Vehiculos 7 Tiempo de ejecución[s] 0.035542

(a) Solución instancia AB220 datos generales



(b) Ruta de uno de los vehículos de la instancia AB220

Figura 3: Representación de la solución de la instancia AB220

6. Descripción del algoritmo

El ciclo principal del programa empieza con una búsqueda greedy (Algoritmo 5) la cual genera una primera solución a través de la búsqueda de óptimos locales. A esta solución se le aplica un algoritmo de búsqueda incompleta Simulated Annealing (Algoritmo 7), el cual no asegura encontrar la solución óptima global para el problema. Luego se chequea la factibilidad (Algoritmo 8) de la solución obtenida: si es factible, la solución es guardada y se modifica la lista de clientes visitables, de manera que todos los clientes que pertenecen a la solución de aquella iteración se eliminan de la lista (se consideran como visitados). El ciclo se reinicia con una solución vacía, y una lista de clientes visitables actualizada. Si la solución es infactible, significa que el algoritmo ya no encuentra nuevas posibles soluciones por lo que se detiene. Las soluciones guardadas se escriben en el archivo de salida.

El algoritmo de búsqueda Greedy (Algoritmo 5), comienza con una solución vacía, una lista de clientes visitables, una lista de estaciones de combustible visitables y una lista con las restricciones del problema. Se comienza agregando el nodo depósito y se define a este como nodo actual. De la lista de clientes visitables, se busca el nodo más cercano al actual y se mide el impacto en tiempo y distancia de agregar el nuevo nodo. En este punto se evalúa si el nodo anterior es igual al nuevo nodo o si el nodo actual es igual al nuevo nodo, si la respuesta es afirmativa significa que se está frente a un subciclo por lo que el vehículo vuelve al depósito. Si no se está frente a un subciclo, entonces se evalúa si desde el nuevo nodo es posible volver al depósito, si la respuesta es afirmativa entonces se agrega el nuevo nodo a la solución, caso contrario, se busca la estación de combustible más cercana y se evalúa si desde esta es posible volver al depósito. Si la respuesta es afirmativa, entonces se agrega el nuevo nodo de combustible y se continua con el recorrido, caso contrario, se vuelve al depósito. Cada vez que se agrega un nodo a la solución, se actualiza la distancia recorrida acumulada y el tiempo empleado, también se modifica la lista de clientes visitables, de la cual se retira el nodo recién agregado. Cuando se agrega una estación de combustible, se actualiza la distancia máxima disponible, lo que equivale a llenar el tanque del auto. Debe quedar claro que para cada nuevo nodo, se evalúa que el tiempo y la distancia disponibles sean tales que permitan al vehículo volver al depósito desde el nuevo nodo. El objetivo de este algoritmo es la minimizar la distancia total recorrida, es por ello que siempre se trata de agregar a la solución el nodo más cercano al actual, es decir, el óptimo local. Note que esta decisión corresponde a la de una función miope, ya que el algoritmo elige siempre el óptimo local aunque no necesariamente esta sea la mejor decisión en el largo plazo.

El algoritmo de búsqueda incompleta Simulated Annealing (Algoritmo 7), recibe una temperatura inicial, una tasa de enfriamiento alfa, una lista de clientes visitables, una lista de estaciones visitables y la solución obtenida del algoritmo de búsqueda greedy. Mientras la temperatura actual sea mayor a la temperatura de término, se genera un ciclo de búsqueda de soluciones, la cual comienza con la aplicación de un movimiento o modificación aleatoria a la solución. Para esta entrega se propone una modificación tipo swap (Algoritmo 6), la cual cambia de posición dos nodos de la solución al azar. Se propone este movimiento ya que es sencillo de implementar y permite generar un vecindario de soluciones relativamente cercano. Esto además es útil como un primer acercamiento a la solución del problema, ya que es fácilmente mezclable con otros métodos de alteración de soluciones o movimientos como bien proponen Koç y Karaoglan 2016, los cuales generan vecindarios para simulated Annealing usando Swap (Algoritmo 6), FSADD (agregar estaciones de servicio de manera aleatoria), Merge (mezcla de rutas), entre otras [9]. En la implementación se utiliza una temperatura inicial de 665 [9] y una tasa de enfriamiento constante de 0,8, lo cual permite una fuerte exploración en un principio y una fuerte intensificación o explotación al final del algoritmo.

Una vez obtenida una solución con Simulated Annealing, se evalúa su factibilidad, tal como se muestra en el algoritmo 8. Se calcula la distancia total del recorrido, así como también el tiempo total invertido. Para que la solución sea factible, estos valores no pueden ser mayores a la distancia Máxima y tiempo de servicio, lo cuales están presentes en la lista de restricciones

del problema. Si resulta que una de estas condiciones es quebrantada, el algoritmo retorna false y el ciclo general se detiene. Además, si la solución no contiene nodos tipo cliente también se considera como solución infactible.

Para esta entrega también se implementa Simulated Annealing con heurística 2-OPT, esto con el objetivo de realizar experimentos y realizar comparaciones. En la implementación de 2-OPT, se eligen dos arcos al azar del arreglo listaClientes (=clientesVisitables), cuya distancia entre si sea mayor a tres (caso contrario equivale a aplicar swap), se cortan, y los nodos que quedaron libres intercambian de lugar. Los elementos del arreglo entre ambos nodos invierten su dirección. Esto permite cambiar dos arcos de la configuración original. La implementación se muestra en el algoritmo 9.

Algorithm 1: setClientesVisitables

Data: solucion, clientesVisitables
Result: clientesVisitables
clientesVisitables ← Retira de clientesVisitables todos los nodos tipo fuel y cliente presentes en solucion;
Return clientesVisitables;

Algorithm 2: distanciaViaje

Data: nodo1, nodo2
Result: distancia
distancia ← Calcula la distancia de Haversine entre nodo1 y nodo2;
Return distancia;

Algorithm 3: tiempoViaje

Data: nodo1, nodo2, restriccionesInstancia
Result: tiempo de viaje
velocidad ← restriccionesInstancia.velocidad;
tiempo ← distanciaViaje(nodo1,nodo2)/velocidad;
Return tiempo de viaje

7. Experimentos

Los experimentos numéricos se llevan a cabo en el subsistema Linux WSL2 para Windows11 64-bits, utilizando un computador LENOVO 82H7 con un procesador 11th Gen Intel(R) Core(TM) i7-1165G7 de 2.8GHz, con una memoria RAM 8192 MB. El código fue escrito en C++ usando un compilador g++ (Ubuntu 9.4.0).

El set de datos utilizado corresponde al set AB usado por Andelmin y Bartolini [1], el cual contiene 40 instancias, con un número de clientes en el rango de 50 a 100, donde todos los clientes son alcanzables desde el depósito. Cada instancia posee información sobre el número de clientes, número de estaciones, tiempo máximo de cada ruta, velocidad (milla/min), tiempo de servicio(min) y tiempo de recarga(min). Además, contiene información sobre cada uno de los nodos que componen el mapa, a saber, id, tipo y coordenadas (latitud y longitud).

El set AB se divide en los subsets AB1 y AB2. Ambos poseen la misma estructura, pero se diferencian en que para las instancias del set AB2 los vehículos tienen menos autonomía de combustible (distancia máxima menor), viajan a mayor velocidad de 60 [milla/hora] y contienen

Algorithm 4: nodoMasCercano

```
Data: listaNodos, nodoActual
Result: nodoMasCercano
distmin ← Numero grande;
distaux ← 0;
initialize nodoMasCercano;
foreach nodo ∈ listaNodos do
    distAux ← distanciaViaje(nodo, nodoActual);
    if distaux < distmin then
        nodoMasCercano ← nodo;
        distmin ← distaux;
Return nodoMasCercano;
```

clientes que no pueden ser atendidos usando una sola recarga de combustible (esto último no implica que las soluciones encontradas deban contener más de una estación de recarga entre dos clientes) [1].El resto de las características son iguales para ambos sets. Se exponen en esta entrega una serie de experimentos numéricos que permitirán comparar el desempeño de cada algoritmo: Solo búsqueda Greedy, SA con swap y SA con 2-opt.

7.0.1. Experimento 1

El experimento 1 consiste en comparar el desempeño de los tres algoritmos por separado en las instancias del set AB1. Se compara la calidad de la solución, tiempo de computación, el número de clientes visitados, el número de vehículos usados y la factibilidad de la solución. Los parámetros de SA son $\alpha = 0{,}8$ y $T = 665$.

7.0.2. Experimento 2

El experimento 2 consiste en comparar el desempeño de los tres algoritmos por separado en las instancias del set AB2. Se compara la calidad de la solución, tiempo de computación, el número de clientes visitados, el número de vehículos usados y la factibilidad de la solución. Los parámetros de SA son $\alpha = 0{,}8$ y $T = 665$.

7.0.3. Experimento 3

Se mide el impacto de la temperatura T inicial con y $\alpha = 0{,}8$ fijo en SA. Para esto se compara el desempeño de SA con swap y con 2-opt en las instancias del set AB2, en términos de la calidad de la solución, tiempo de computación, número de clientes visitados y número de vehículos usados. Se utilizan las temperaturas $T = 100$, $T = 50$ y $T = 10$.

7.0.4. Experimento 4

Se mide el impacto de la tasa de enfriamiento α en SA con una temperatura $T = 100$ fija. Para esto se compara el desempeño de SA con swap y con 2-opt en las instancias del set AB2, en términos de la calidad de la solución, tiempo de computación, número de clientes visitados y número de vehículos usados. Se utilizan las tasas $\alpha = 0{,}99$, $\alpha = 0{,}80$ y $\alpha = 0{,}50$

Algorithm 5: Greedy

```
Data: solucion(vacía), clientesVisitables, estacionesVisitables, restriccionesInstancia
Result: solucion(completa), clientesVisitables, estacionesVisitables, tiempoSolucion,
        distanciaSolucion
tiempoMax ← restriccionesInstancia.MaxTime;
distanciaMax ← restriccionesInstancia.MaxDistance;
tiempoRecarga ← restriccionesInstancia.RefuelTime;
tiempoAtencion ← restriccionesInstancia.ServiceTime;
tiempoSolucion, distanciaSolucion, tiempoViaje(), distanciaViaje();
flag ← true;
solucion.push(deposito);
nodoActual ← deposito;
while flag do
    clientesVisitables ← setClientesVisitables(solucion,clientesVisitables);
    nodoNuevo ← nodoMasCercano(clientesVisitables,nodoActual);
    if nodoAnterior = nodoNuevo o nodoActual = nodoNuevo then
        solucion.push(deposito);
        tiempoSolucion ← tiempoSolucion + tiempoViaje(nodoActual,deposito);
        distanciaSolucion ← distanciaViaje(nodoActual,deposito);
        tiempoMax ← tiempoMax - tiempoViaje(nodoActual,deposito);
        distanciaMax ← distanciaMax - distanciaViaje(nodoActual,deposito);
        flag=false;
    else
        if tiempoMax-(tiempoViaje(nodoActual,nodoNuevo) + tiempoAtencion +
          tiempoViaje(nodoNuevo,deposito)) ≥ 0 y
          distanciaMax-(distanciaViaje(nodoActual,nodoNuevo) +
          distanciaViaje(nodoNuevo,deposito)) ≥ 0 then
            solucion.push(nodoNuevo);
            nodoActual ← nodoNuevo;
            tiempoSolucion ← tiempoSolucion + tiempoViaje(nodoActual,nodoNuevo) +
              tiempoAtencion;
            distanciaSolucion ← distanciaViaje(nodoActual,nodoNuevo);
            tiempoMax ← tiempoMax - (tiempoViaje(nodoActual,nodoNuevo) +
              tiempoAtencion);
            distanciaMax ← distanciaMax - distanciaViaje(nodoActual,nodoNuevo) ;
        else
            nodoNuevo←fuelMasCercano presente en estacionesVisitables;
            if tiempoMax-(tiempoViaje(nodoActual,nodoNuevo) + tiempoRecarga +
              tiempoViaje(nodoNuevo,deposito)) ≥ 0 y
              distanciaMax-(distanciaViaje(nodoActual,nodoNuevo) +
              distanciaViaje(nodoNuevo,deposito)) ≥ 0 then
                solucion.push(nodoNuevo) nodoActual ← nodoNuevo;
                tiempoSolucion ← tiempoSolucion +
                  tiempoViaje(nodoActual,nodoNuevo) + tiempoRecarga;
                distanciaSolucion ← distanciaViaje(nodoActual,nodoNuevo);
                tiempoMax ← tiempoMax - (tiempoViaje(nodoActual,nodoNuevo) +
                  tiempoRecarga);
                distanciaMax ← MaxDistance;
            else
                solucion.push(deposito);
                tiempoSolucion←tiempoSolucion + tiempoViaje(nodoActual,deposito);
                distanciaSolucion←distanciaViaje(nodoActual,deposito);
                tiempoMax←tiempoMax - tiempoViaje(nodoActual,deposito);
                distanciaMax←distanciaMax - distanciaViaje(nodoActual,deposito) ;
                flag=false;
Return solucion
```

Algorithm 6: swapNodos

```
Data: solucion
Result: solucion
size ← # nodos que contiene la solucion;
posicion1←random[1,size-2];
posicion2←random[1,size-2];
solucion[posicion1←solucion[posicion2];
solucion[posicion2←solucion[posicion1];
Return solucion
```

Algorithm 7: simulatedAnnealing

```
Data: temperatura, alfa, solucion, clientesVisitables, estacionesVisitables
Result: solucion
temperaturaTermino←1;
Sc←solucion;
Sn← [];
Sbest←Sc;
while temperatura > temperaturaTermino do
    Sn←swapNodos(Sc);
    distanciaSn←distanciaTotalRecorrida(Sn);
    distanciaSc←distanciaTotalRecorrida(Sc);
    distanciaSbest←distanciaTotalRecorrida(Sbest);
    deltaEvaluacion← −|distanciaSn − distanciaSc|;
    if distanciaSn < distanciaSc then
        Sc←Sn;
        distanciaSc ← distanciaTotalRecorrida(Sc);
    else
        if rand[0,1] < exp(deltaEvaluacion/temperatura);
         then
            Sc←Sn;
            distanciaSc←distanciaTotalRecorrida(Sc);
    if distanciaSc < distanciaSbest then
        Sbest←Sc;
    temperatura←temperatura ∗ alfa;
tiempoSolucion←tiempoTotalRecorrido(Sbest);
distanciaSolucion←distanciaSbest;
clientesVisitables←setClientesVisitables(Sbest,clientesVisitables);
solucion←Sbest;
Return solucion
```

Algorithm 8: factible

Data: solucion, restriccionesInstancia
Result: booleano
solucionFactible←true;
if *distanciaMax-distanciaTotalRecorrida(solucion)<0 o tiempoMax-tiempoTotal(solucion) < 0* **then**
    solucionFactible←false;
else
    if *en la solucion no existen nodos clientes* **then**
        solucionFactible←false;
Return solucionFactible

Algorithm 9: twoOPT

Data: solucion
Result: nodoMasCercano
$aux \leftarrow []$; $n \leftarrow solucion.size() - 2$;
$pos1 \leftarrow$ random[1,n];
$pos2 \leftarrow$ random[1,n];
while *pos1 == pos2 || abs(pos1-pos2)<3* **do**
    elegir pos1 y pos2 random hasta que sean diferentes y su distancia sea mayor a 2;
    Si no es posible retorna Solucion;
Aplicar swap(solucion,pos1,pos2);
solucion ← reverse(solucion,pos1,pos2);
//reverse(), reversa los elementos de 'solucion' entre pos1 y pos2, sin considerar pos1 y pos2;

Return *Solucion*;

Valores promedio	CalidadSolucion	nClientes	nClientes sin atender	nvehiculos	tiempo Ejecucion
AB1Greedy	1683,781690	71,10	10,15	7,6	0,001428550
AB1_2opt	1525,566571	73,25	8	6,2	0,018343150
AB1_Swap	1510,167380	73,25	8	6,2	0,014878200

Cuadro 1: Valores promedio para las instancias del set AB1 del experimento 1

Valores promedio	CalidadSolucion	nClientes	nClientes sin atender	nvehiculos	tiempo Ejecucion
AB2Greedy	1537,813425	73,25	8	6,2	0,00134450
AB2_2opt	1525,566571	73,25	8	6,2	0,01834315
AB2_Swap	1510,167380	73,25	8	6,2	0,01487820

Cuadro 2: Valores promedio para las instancias del set AB2 del experimento 2

8. Resultados

Para el experimento 1, en base a los resultados de las tabla 1, se observa que el algoritmo de SA con Swap es el que genera la mejor calidad de solución, donde el número de clientes atendidos y vehículos usados coincide entre SA con Swap y SA con 2-opt. Aunque el algoritmo de búsqueda Greedy genera la peor calidad de solución, esta sigue siendo muy cercana a la obtenida por los otros algoritmos, pero el tiempo de computación es un orden de magnitud menor, por lo que parece ser una buena alternativa del punto de vista del costo computacional en relación a la calidad de solución. Por último indicar que el número de clientes promedio sin atender es similar en los tres algoritmos. Para el experimento 2, en base a los resultados de la tabla 2, se observa resultados similares al del experimento 1. Parece ser que el efecto del aumento de velocidad y menor autonomía de combustible en las instancias AB2 implica un mejor resultado inicial para Greedy y resultados más homogéneos entre los tres algoritmos. Por último indicar que todas las soluciones son factibles.

En el experimento 3, en base a la tabla 3 se observa que la calidad de la solución se mantiene casi constante, presentando una leve mejoría en $T = 50$, y no se observan cambios en el número de clientes promedio atendidos, número de vehículos usados ni tiempo de ejecución. En la tabla 4 se observa que para SA más Swap a medida que la temperatura disminuye, la calidad de la solución no presenta una tendencia clara y no se observan cambios en el número de clientes promedio atendidos, número de vehículos usados ni tiempo de ejecución. Los resultados de las tablas 1 y 2, se obtienen usando SA con una temperatura inicial $T = 665$ y $\alpha = 0,8$, mientras que las de las tablas 3 y 4 se obtienen con temperaturas entre 100 y 10. Se observa que la calidad de las soluciones para AB2 usando SA es muy similar, pero el tiempo de ejecución disminuye en un orden de magnitud con temperaturas iniciales más bajas. Una temperatura de inicio menor supone una intensificación más temprana, lo cual genera una menor probabilidad de aceptar soluciones de peor calidad, pero también una menor exploración del espacio de búsqueda. Al comparar 2opt y swap, se tiene que SA más Swap es el que genera las soluciones de mayor calidad, esto se debe a que Swap genera una mayor variación en las soluciones, ya que modifica una mayor cantidad de arcos que 2-opt. Esto permite compensar la baja exploración y la pronta intensificación generada al comenzar con temperaturas más bajas.

En base a los resultados de las tablas 5 y 6 del experimento 4, se observa que una tasa de enfriamiento mayor implica una solución de mayor calidad. Esto ocurre debido a que una tasa de enfriamiento mayor supone un mayor número de iteraciones, con lo que mejora la exploración e intensificación. Además, se observa que los mejores resultados se obtienen con SA más Swap. Esto se debe a que Swap genera soluciones más variadas que 2-Opt lo que favorece la exploración,

Temperatura AB2_2opt	CalidadSolucion Prom	nClientes Prom	nvehiculos Prom	tiempo Ejecución Prom
100	1533,486646	73,25	6,2	0,00769455
50	1524,600549	73,25	6,2	0,00691155
10	1533,213696	73,25	6,2	0,0047513

Cuadro 3: Resultados del experimento 3 para SA más 2-opt con $\alpha = 0{,}8$ constante

Temperatura AB2_Swap	CalidadSolucion Prom	nClientes Prom	nvehiculos Prom	tiempo Ejecucion Prom
100	1522,133582	73,25	6,2	0,00635715
50	1517,010974	73,25	6,2	0,00585845
10	1526,411865	73,25	6,2	0,00405385

Cuadro 4: Resultados del experimento 3 para SA más Swap con $\alpha = 0{,}8$ constante

y permite conseguir mejores resultados en la etapa de intensificación.

9. Conclusiones

No todas las técnicas resuelven el mismo problema. MCWS/DBCA resuelve el modelo original de G-VRP propuesto por Erdoğan y Miller-Hooks (2012), mientras que el modelo propuesto por Koc y Karaoglan (2016), resuelto con B&C no considera los nodos dummy del modelo original (con ello evita resolver el problema basado en un grafo de mayor tamaño). Schneider et.al (2014) propone un modelo diferente, ya que considera en este ventanas de tiempo de atención de clientes, por lo que VNS/TS resuelve un problema que se considera como una extensión del problema G-VRP. Andelmin y Bartolini (2017) proponen un modelo basado en un multigrafo que no requiere explícitamentemente modelar las AFS, por lo que el algoritmo exacto propuesto resuelve un problema con una estructura diferente al original. Bruglieri et.al (2019) resuelve el mismo modelo planteado por Erdoğan y Miller-Hooks (2012) con una estructura de resolución algorítmica diferente.

El enfoque que mejores resultados a dado en calidad de la solución/número de óptimos y tiempo de computación hasta el 2019, es el enfoque de solución exacta Path Based Approach (PBA) para instancias pequeñas del problema [3], mientras que para instancias más grandes, el enfoque que obtiene mejor tiempo de CPU es la heurística multi-space sampling heuristic (MSH) y el que mejor calidad de solución arroja es el enfoque de algoritmo exacto propuesto por Andelmin et.al (2017)[3].

Se concluye que algoritmo de búsqueda Greedy genera una calidad de solución similar a los otros algoritmos, pero con un tiempo de computación un orden de magnitud menor. El efecto

Tasa Enfriamiento AB2_2opt	CalidadSolucion Prom	nClientes Prom	nvehiculos Prom	tiempo Ejecucion Prom
0,5	1534,6283750	73,25	6,2	0,00382370
0,8	1533,4866460	73,25	6,2	0,00769455
0,99	1510,693958	73,25	6,2	0,12637840

Cuadro 5: Resultados del experimento 4 para SA más 2opt con $T = 100$ constante

Tasa Enfriamiento AB2_Swap	CalidadSolucion Prom	nClientes Prom	nvehiculos Prom	tiempo Ejecucion Prom
0,5	1529,209500	73,25	6,2	0,00318150
0,8	1522,133582	73,25	6,2	0,00635715
0,99	1470,317276	73,25	6,2	0,10087005

Cuadro 6: Resultados del experimento 4 para SA más Swap con $T = 100$ constante

del aumento de velocidad y menor autonomía de combustible en las instancias AB2 implica un mejor resultado inicial para Greedy y resultados más homogéneos entre los tres algoritmos. Además, el efecto del aumento de velocidad y menor autonomía de combustible en las instancias AB2 implica un mejor resultado inicial para Greedy y resultados más homogeneos entre los tres algoritmos.

Se concluye que el algoritmo que mejores resultados genera es SA más swap, el cual se desempeña mejor con una tasa de enfriamiento α cercana a 1. Swap genera una mayor variación en las rutas de vehículos, esto combinado con una tasa de enfriamiento α cercana a 1 (mayor número de iteraciones) permite una fuerte exploración inicial.

Se concluye que para esta implementación en particular, no es claro el efecto de la temperatura en los algoritmos de SA más heurísticas.

Se concluye que para los algoritmos SA más swap y SA más 2opt, una tasa de enfriamiento mayor implica una solución de mayor calidad, ya que supone un mayor número de iteraciones, pero también implica un mayor tiempo de ejecución.

10. Bibliografía

Referencias

[1] Juho Andelmin and Enrico Bartolini. An exact algorithm for the green vehicle routing problem. *Transportation Science*, 51(4):1288–1303, 2017.

[2] Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011. Supply chain disruption and risk management.

[3] M. Bruglieri, S. Mancini, F. Pezzella, and O. Pisacane. A path-based solution approach for the green vehicle routing problem. *Computers & Operations Research*, 103:109–122, 2019.

[4] 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.

[5] Jane Lin, Wei Zhou, and Ouri Wolfson. Electric vehicle routing problem. *Transportation Research Procedia*, 12:508–521, 2016. Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain.

[6] 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.

[7] 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: Emerging Technologies*, 70:113–128, 2016.

[8] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, 48(4):500–520, 2014.

[9] Çağrı Koç and Ismail Karaoglan. The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing*, 39:154–164, 2016.