Inteligencia Artificial

Informe Final: Vehicle Routing Problem with Backhauls

Vicente Perelli Tassara

1 de diciembre de 2022

Evaluación

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

Resumen

En VRPB, el conjunto de clientes es particionado en dos: clientes linehaul que requieren entregas, y clientes backhaul, que requieren retiros. Ambos tipos de clientes deben ser visitados de manera contigua, respetando el orden de precedencia, donde toda ruta debe tener al menos un cliente linehaul. Las entregas son cargadas en el depósito inicial y todos los retiros, entregados en el punto de partida. Este documento se centra en definir el problema, recopilar las estrategias utilizadas para resolver VRPB, presentar un modelo matemático, explicar la representación utilizada, describir el algoritmo implementado y realizar experimentos comparando variaciones del algoritmo. Se ha observado que para este problema, la mayoría de las soluciones existentes son implementadas utilizando heurísticas, y que aplicar una técnica completa de búsqueda de soluciones puede resultar contraproducente.

1. Introducción

El problema de enrutamiento de vehículos es uno de los desafíos más críticos que las empresas de logistíca deben enfrentar al día de hoy, considerando el robusto mercado de entregas y retiros a domicilio que desde la pandemia mundial en 2019 ha crecido sin precedentes, reportando un 81,2 % de crecimiento para el año 2021 en comparación con el año anterior [1]. Investigadores han estudiado este problema y agendado las entregas desde 1959, cuando Danzig y Ramser (1959) introdujeron por primera vez el Truck Dispatching Problem (TDP), considerando el caso donde un centro de distribución (o depósito) debe satisfacer la demanda de ciertos clientes a diferentes distancias geográficas, buscando minimizar las rutas efectuadas por los vehículos para reducir los costes dado la distancia viajada [2]. Con este mismo propósito Deif y Bodin proponen VRPB, una variante del problema anteriormente descrito, que particiona los clientes en dos subconjuntos: aquellos que requieren la entrega de un producto, y aquellos clientes que tienen alguna existencia a retirar.

El propósito de este documento es definir el problema de VRPB (Vehicle Routing Problem with Backhauls, o VRPwB) desde los múltiples artículos encontrados en la literatura disponible, para facilitar el entendimiento del lector sobre éste, conociendo primeramente sus orígenes, las diferentes variantes derivadas del problema en cuestión y los métodos más utilizados para obtener un conjunto de rutas óptimo, presentando un modelo matemático general para su posterior implementación en las próximas entregas.

A continuación, se detalla la estructura del documento. En la sección 2, se define VRPB, presentando una explicación detallada del problema estudiado, las dificultades relacionadas y las diferentes variantes surgidas a partir del problema en cuestión. En la sección 3, se da cuenta del origen histórico del problema, los diferentes métodos de resolución hallados en la literatura, junto con los algoritmos que han obtenido mejor desempeño a la fecha y la tendencia actual de las técnicas para la resolución del problema. En la sección 4 se detalla un modelo matemático que describe el problema de manera general. Finalmente, en la sección 5 se presentan conclusiones obtenidas.

2. Definición del Problema

El Problema de Enrutamiento de Vehículos con Backhauls (conocido como VRPB, por sus síglas en inglés), tiene como objetivo obtener las rutas de coste mínimo para abastecer y retirar todos los productos que el servicio requiera, dada una flota determinada y homogénea de vehículos con capacidad de carga fija y un conjunto de clientes a una distancia conocida.

En VRPB, la variable de decisión considera viajar de un cliente i a un cliente j, el conjunto de clientes es sub agrupado en clientes linehaul y backhaul, donde en linehaul cada cliente requiere la entrega de cierta cantidad de productos y cada backhaul un retiro. Los clientes linehaul deben ser visitados primero, seguido de los clientes backhaul, por otro lado, todas las entregas deben ser primeramente cargadas en el depósito y los retiros deben ser transportados al depósito inicial. El especial cuidado en el orden los retiros y entregas, es para evitar la re-distribución de la carga durante los viajes, dado su coste económico implicado en la operación [3]. Para este problema, es importante considerar que cada vehículo hace exactamente una ruta, y cada cliente es visitado por un sólo vehículo. Además, las rutas pueden ser compuestas sólamente por clientes linehaul, pero no únicamente por clientes backhaul.

Dentro de las dificultades encontradas al obtener una solución para este problema de tipo NP-hard [4], se debe considerar que los computadores tienen una capacidad de cómputo limitada, donde al incrementar el tamaño de las instancias existe una explosión combinatorial de soluciones candidatas que permiten resolver el problema, haciendo cada vez más difícil el converger en una solución factible.

Con los años, una multitud de variantes de este problema han surgido. A continuación se procederá a detallar algunas de las más importantes.

El problema de vehículos mixto, o mixed VRPB (VRPMB) fue primeramente estudiado por Wade y Salhi (2002) [5] y considera el caso en que los backhauls no tienen la restricción de precedencia en las visitas, es decir, cuando los vehículos no necesitan realizar una entrega en un cliente linehaul previamente. Otra variante, es la propuesta Salhi y Nagy (1999), el VRPB con multi-depósito [6]. En esta variante, existen múltiples depósitos los cuales tienen asociados cada uno un conjunto de clientes. Además de las mencionadas anteriormente, Thangiah et al. (1996) introdujo una versión basada en intervalos temporales (VRPB with Time Windows, o VRPBTW por sus síglas en inglés). La idea bajo esta propuesta es considerar una ventana temporal y un tiempo de servicio por cada cliente, donde el coste de cada arco es interpretado como el tiempo

de viaje y su objetivo principal es minimizar el número de vehículos utilizados, dejando como objetivo secundario el minimizar el tiempo máximo empleado en el servicio. Finalmente, está la variante de VRPB con flota fija heterogénea (heterogeneous fixed fleet VRPB, o HFFVRPB), propuesta por Tavakkoli-Moghaddam et al.(2006), donde además de las restricciones impuestas para el problema VRPB, se debe decidir sobre la composición de la flota vehicular [7].

3. Estado del Arte

Para establecer una línea temporal en términos de métodos, algoritmos y soluciones para VRPB, primero se debe exponer el origen del problema, ya que VRPB es en sí mismo una variante de VRP (Vehicle Routing Problem) que a su vez nace de una generalización del TSP (Travelling Salesman Problem). Para entender este concepto, se debe conocer desde la definición de TSP estudiada en el curso, el origen de VRP y como se plantea, para luego explicar las razones del origen de la variante en estudio, VRPB.

En 1959, se publicó el artículo 'The Truck Dispatching Problem' [2], el cual planteó por primera vez una generalización del TSP, agregando una nueva condición correspondiente a la carga máxima que el viajero puede transportar. Junto a esto, se planteó la idea de que ciertos encargos deban ser entregados en ciertos puntos. Teniendo esto en cuenta, y considerando una capacidad de carga del viajero siempre mucho menor al peso total de todos los paquetes a enviar, se da origen al VRP.

El Vehicle Routing Problem (VRP por sus siglas en inglés) busca construir rutas para una flota de vehículos homogénea, con el propósito servir a un conjunto de clientes, minimizando el coste estas. Cada cliente es visitado sólo una vez por un vehículo y cada ruta inicia y termina en el depósito, tal que todos los clientes sean visitados y, que la suma de las demandas para cada vehículo no supere su capacidad.

El problema estudiado en este documento, corresponde a una conocida variante del problema anteriormente mencionado, el VRP con Backhauls (conocido como VRPB por sus siglas en inglés), introducido por Deif y Bodin (1984) [8], descrito en la sección anterior y considerado un problema *NP-hard* ya que extiende al VRP [4].

3.1. Métodos de resolución para VRPB

Diversos enfoques se han utilizado para buscar la forma más eficiente de dar respuesta a este tipo de problemas. A continuación, se detallarán los principales encontrados en la literatura.

3.1.1. Heurísticas

Junto con la problemática, los autores Deif y Bodin propusieron una extensión de la heurística previamente implementada por Clarke y Wright para resolver los problemas de tipo VRP, donde el algoritmo primero construye una solución a partir de rutas para sólo un cliente, y luego iterativamente considera el ahorro logrado al agregar mas clientes dentro de una misma ruta, en comparación al coste comprometido con hacer viajes individualmente [7].

Goetshalckx y Jacobs-Blecha (1989) [3] desarrollaron una heurística basándose en el concepto de *space-filling curves*, utilizado por Bartholdi y Platzman para encontrar soluciones en un problema de tipo TSP. El método particiona los dos tipos de clientes en secuencias de puntos para conformar una solución factible, cada solución conformada sólamente por clientes de un tipo. Luego, según los resultados obtenidos a través de la técnica de *space-filling mapping*, cada ruta de clientes tipo linehaul es fusionada con la ruta de clientes tipo backhaul más cercana,

para obtener un conjunto final de rutas que minimicen el coste. En general, este método obtiene peores resultados que los logrados por el método de Deif y Bodin, pero se desempeña mejor en términos de velocidad para instancias de mayor tamaño.

En Goetschalckx y Jacobs-Blecha (1992) se propone una extención de la heurística *cluster-first, route-second* para determinar las rutas óptimas. El método primeramente genera rutas de manera iterativa considerando los clientes más cercanos a un valor llamado *K seed radial*, obtenido mediante la solución de un problema de ubicación/asignación capacitado (capacitated location-allocation problem), secuenciando los clientes por su distancia al centro de distribución. Luego, se resuelven los problemas de asignación generalizados (GAP) heurísticamente para agrupar los clientes en K clusters. Finalmente, determina las rutas a través de una heurística de inserción de TSP modificada, y se hacen procedimientos de post-optimización. La heurística propuesta superó los métodos previos para las instancias planteadas por los mismos autores en su trabajo de 1989 [9].

Posteriormente, Toth y Vigo (1996) propusieron una forma mejorada de la heurística extendida anteriormente, la cual comienza desde una solución infactible obtenida por una relajación lagrangiana del problema e intenta hacerla factible a través de intercambios entre los arcos que componen las rutas, de manera intra-rutas e inter-rutas.

A pesar de su poca popularidad, otro tipo de heurística utilizado para este problema es el de *Population search*. Desarrollado por Vidal et al.(2014) para diversas variantes de VRP incluyendo VRPB, se propone un algoritmo genético unificado, utilizando diferentes operadores de búsqueda local y mecanismos diversificadores independientes del problema. Para mejorar la calidad del espacio de búsqueda local, los autores proponen una metodología de evaluación de rutas unificada. El algoritmo evalúa los movimientos como una concetenación de sub secuencias, preprocesando su información en un proceso posterior de optimización, y junto a otros mecanismos avanzados, logro varias nuevas mejores soluciones para las instancias de Goetschalckx y Jacobs-Blecha [7].

Ghaziri y Osman (2006) describieron un algoritmo de mapas de características autoorganizado, basado en redes neuronales competitivas no-supervisadas. Definiendo cada solución como una cantidad de nodos interconectados que consolida la arquitectura de los nodos, se utiliza el principio de *winner-take-all* para presentar un cliente al algoritmo y seleccionar la neurona más cercana en términos de distancia. Luego, se identifica la manera en que las neuronas adaptan sus estados o posiciones para finalmente optimizar la solución aplicando un procedimiento 2-opt. Segun los autores, muy buenos resultados fueron obtenidos, donde para las instancias de Toth y Vigo se obtuvieron 14 nuevas mejores soluciones [7].

3.1.2. Metaheurísticas

Algunas metaheurísticas locales han sido implementadas para resolver el problema. Osman y Wassan (2002) detallan como resolver el problema basándose en la heurística *reactive tabu search*, la cual se propone desde dos heurísticas de construcción de rutas: *saving-insertion* y *saving-assignment*, para rápidamente generar soluciones iniciales. El concepto *reactivo* de la heurística es utilizado para acelerar las diferentes estructuras de vecindad utilizadas en las fases de intensificación y diversificación del algoritmo. Además, el método combina varias estructuras especializadas para administrar de forma eficiente las búsquedas. Este método fue reportado con buen desempeño en las instancias de Goetschalckx y obtuvo nuevas mejores soluciones para las instancias de mayor tamaño propuestas por Toth y Vigo (1996). El algoritmo fue posteriormente combinado con un esquema de programación de memoria adaptativa por Wassan (2007) para balancear las fases de intensificación y diversificación, aportando mayor robustez al proceso de

búsqueda y dando como resultado una convergencia temprana. Gracias a esta mejora, se obtuvieron 45 nuevas mejores soluciones en las instancias utilizadas como benchmark.[7]

Otro algoritmo basado en la heurística de *tabu search* fue desarrollado por Brandão (2006). Este algoritmo propone iniciar la búsqueda desde una cota pseudo-inferior, obtenida a través de dos heuristícas constructivas. La primera utiliza la solución para dos problemas VRP abiertos, basado en el hecho de que una ruta VRPB está constituida por dos caminos hamiltonianos unidos uno después del otro, donde los vehículos no cumplen con la restricción de volver al inicio de cada ruta. El segundo método utiliza una cota inferior para el VRP llamada *K-tree initial solution*, ya que utilizan *K-trees* para el cálculo de esta cota inferior [10].

Más recientemente se han propuesto varias heurísticas de búsqueda local para el VRPB estándar. De esta forma, Ropke y Pisinger (2006) presentaron una heurística de búsqueda unificada adaptativa para grandes vecindades, la cual es capaz de resolver el VRPB estándar y 5 variantes de este: VRPMB, VRPBTW, Multiple Depot Mixed VRPB, Mixed VRPBTW y el VRPB con entregas y retiros simultáneos, además de permitir el modelamiento de otros problemas de enrutamiento. Este método arrojó resultados de vanguardia en 338 instancias, incluyendo 227 nuevas mejores soluciones [7].

Otros métodos también se han implementado. Gajpal y Abad (2009) desarrollaron un algoritmo basándose en un sistema de multi-colonias de hormigas (MACS, por sus siglas en inglés) para resolver el VRPB. *Ant colony system* (ACS) es un enfoque algorítmico inspirado en el comportamiento de hormigas reales, donde se utilizan *hormigas artificiales* para construir una solución a través de *información feromónica* de soluciones previamente generadas [11]. Este método es un procedimiento de *cluster-first, route-second*, que utiliza dos tipos de hormigas para la solución. La primera asigna clientes a los vehículos mientras que la segunda construye una ruta para un vehículo dados los clientes asignados. Para esta heurística, se lograron buenos resultados de desempeño en promedio y se obtuvieron 5 nuevas mejores soluciones [7]. Zachariadis y Kiranoudis (2012) desarrollaron una heurística basada en la implementación eficiente de la *variable length bone exchange local search*. Este algoritmo fue aplicado en 62 benchmarks conocidas para VRPB y arrojó las mejores soluciones conocidas para cada uno de ellos. Finalmente, se propuso un algoritmo basado en la *iterated local search*, por Cuervo et al. (2014). El algoritmo explora una estructura de vecindad amplia en cada iteración y utiliza una estructura de datos para almacenar información sobre el conjunto de soluciones vecinas. Para controlar la factibilidad, un ajuste dinámico sobre la penalización de costos es aplicado a las soluciones infactibles. Gracias a esto, muchas soluciones fueron mejoradas en comparación con las heurísticas clásicas, pero el método requiere mucho mayor tiempo de computación para obtener las soluciones.

3.1.3. Métodos exactos

Varios algoritmos exactos han sido desarrollados para resolver de manera eficiente, pero se ha probado que la mayoría de ellos son óptimos solamente en instancias de tamaño pequeño y mediano.

El primer método exacto descrito en la literatura, fue propuesto por Yano et al. (1987). Los autores describen un procedimiento que utiliza el bien conocido método de *set-partitioning* para encontrar un conjunto de rutas óptimo. Se ha probado que el algoritmo descrito es óptimo para pequeñas rutas, considerando un promedio de 15 puntos de entrega y no más de 15 puntos de retiro [3], con rutas compuestas por un máximo de 4 clientes cada una, siendo un método difícil de implementar en problemas de tamaño real.

Toth y Vigo (1997) desarrollaron el primer método exacto basado un algoritmo de *branch-and-bound*, utilizando una formulación para programación lineal entera la cual computa cotas inferiores a través de una relajación lagrangiana de las variables. Esta relajación es combinada con otra obtenida al dejar de tener en cuenta las restricciones de capacidad del modelo, produciendo un procedimiento de doble cota, el cual es utilizado en un algoritmo de *branch-and-bound* para resolver el VRPB a optimalidad [4][12]. El algoritmo exacto fué capaz de resolver las instancias propuestas por Goetschalckx y Jacobs-Blecha (1989) considerando un conjunto de 25 a 68 clientes, y generó nuevas instancias para un conjunto entre 21 a 100 clientes [7].

Mingozzi et al. (1999) proponen una formulación basada en *set-partitioning* haciendo uso de variables para *caminos elementales* sobre dos subgrafos inducidos por los clientes de tipo backhaul y linehaul, respectivamente. En este artículo, los autores combinan dos heurísticas para resolver el problema dual, donde a través de los límites resultantes de la computación de cotas inferiores mediante una relajación de tipo LP, se reduce el número de caminos (variables) del modelo sin pérdida de optimalidad [12]. Resulta importante destacar que este método es capaz de resolver todas las instancias propuestas por Goetschalckx y Jacobs-Blecha (1989) y Toth y Vigo (1997) [7].

Queiroga et al. (2019) proponen dos algoritmos de tipo *branch-cut-and-price* (BCP) para resolver de manera exacta el VRPB. El primero siguiendo el enfoque tradicional de un subproblema de fijación de precios, y el segundo explotando la condición de particiones del conjunto de clientes en linehaul/backhaul, definiendo dos subproblemas de fijación de precios. Estos métodos, incorporan elementos del estado del arte de BCP como *ng-routes/paths, rounded capacity cuts, limited-memory rank-1 cuts, strong branching, route enumeration* y *arc elimination*, utilizando costes reducidos y *dual stabilization*. Según los autores, se demostró a través de experimentos computacionales que los algoritmos propuestos son capaces de obtener soluciones óptimas para todas las instancias propuestas a modo de *benchmark* con hasta 200 clientes, varias de ellas por primera vez. Junto con esto, nuevas instancias fueron propuestas con hasta 1000 clientes, utilizando las instancias propuestas para CVRP por Uchoa et al. (2017) y siguiendo el esquema propuesto por Toth y Vigo(1997), además de mostrar buenos resultados para las variantes del problema VRPBTW y HFFVTPB [12].

3.2. Mejores algoritmos y Tendencia actual

Por los resultados anteriormente descritos, y las formas de agrupar los métodos presentados, se separarán las comparaciones algorítmicas de la misma manera que en este documento.

Para probar computacionalmente el desempeño de los algoritmos, los autores de cada método utilizan los conjuntos de instancias propuestos por Goetschalckx y Jacobs-Blecha (GJ89), junto con los de Toth y Vigo (TV97). En el primer conjunto de instancias, la desviación promedio de todas las heurísticas propuestas es menor a 0.5 %, y en términos de calidad de solución, Vidal et al.(2014), junto con Gajpal y Abad (2009) son los mejores. Para el segundo, se obtienen desviaciones estándares menores a 0.75 % para cada heurística. Respecto a la calidad de las soluciones, Gajpal y Abad (2009) junto con Brandão (2006) son los favoritos[7].

Por otro lado, los algoritmos exactos son muy escasos, existiendo a la fecha sólo 4 de ellos, por consecuencia, el mejor algoritmo a la fecha conocido es el último propuesto [12], dado que a diferencia de los otros algoritmos exactos, éste resuelve todas las instancias de GJ89 y TV97, encontrando incluso algunas soluciones óptimas nuevas nunca antes reportadas y mejorando otras en ambos conjuntos. Además, demuestra la eficiencia obtenida al utilizar variables separadas para linehaul y backhaul, reportando los resultados de las pruebas realizadas bajo el conjunto

de instancias propuesto por los mismos autores.

Dada la cantidad de métodos heurísticos y meta-heurísticos aplicados a encontrar soluciones para el VRPB estándar, y considerando su orden cronológico se ha notado que existe una tendencia creciente a desarrollar algoritmos que implementen heurísticas, siendo las de tipo constructivas las más comunes, seguidas de aquellas que utilizan la técnica de *tabu search*.

4. Modelo Matemático

A continuación, se presenta el modelo planteado por Toth y Vigo (1997) [12] [4] . Dado un grafo dirigido $G = (V, A)$, donde $V = 0 \cup L \cup B$, con $\{0\}$ el vértice que representa el depósito, L los clientes linehaul con $L = \{1, 2, ..., n\}$ y los clientes backhaul $B = \{n+1, n+2, ..., n+m\}$. Por otro lado, sea $A' = A_1 \cup A_2 \cup A_3$, $L_0 := L \cup \{0\}$ y $B_0 := B \cup \{0\}$, donde

$$A_1 := \{(i,j) \in A : i \in L_0, j \in L\},$$
$$A_2 := \{(i,j) \in A : i \in B, j \in B_0\},$$
$$A_3 := \{(i,j) \in A : i \in L, j \in B_0\}$$

En otras palabras, el conjunto de arcos A' puede ser particionado en tres subconjuntos disjuntos. El primero, A_1, contiene todos los arcos desde el depósito y vértices linehaul, a vértices linehaul. El segundo conjunto, A_2, contiene todos los arcos desde vértices backhaul a vértices del mismo conjunto y el depósito. El tercero, A_3, contiene los arcos que conectan a los vértices linehaul con los backhaul, o el depósito. Es muy importante notar que A' no contiene arcos que no puedan pertenecer a una solución factible, dado que no existen arcos desde el backhaul hacia el linehaul, o desde el depósito a un vértice backhaul.

Sea $\mathcal{F} = \mathcal{L} \cup \mathcal{B}$ donde $\mathcal{L}$ y $\mathcal{B}$ son las familias de todos los subconjuntos de vértices L o B, respectivamente. Para cada $S \subseteq \mathcal{F}$, se define $r(S) = \lceil \sum_{i \in S} d_i / Q \rceil$ como una cota inferior del número de vehículos necesario para cumplir con las demandas de todos los clientes en S. Para cada $i \in V$ se define $\Delta_i^+ = \{j : (i,j) \in A'\}$ y $\Delta_i^- = \{j : (j,i) \in A'\}$.

4.1. Constantes

L : Conjunto de clientes linehaul
B : Conjunto de clientes backhaul
Q : Capacidad de carga de un vehículo
c_{ij} : Coste asociado al arco $(i,j) \in A$
d_i : Demanda asociada al vertice $i \in V$

Cabe destacar que en este documento, el coste asociado a los arcos se considera como la distancia euclidiana entre dos vértices.

4.2. Variables

$$x_{ij} = \begin{cases} 1 & \text{si un arco } (i,j) \in A' \text{ es atravesado por un vehículo} \\ 0 & \text{e.o.c.} \end{cases} \tag{1}$$

4.3. Función objetivo

El objetivo principal de la función es minimizar la distancia total recorrida por la flota de vehículos.

$$Min \sum_{(i,j)\in A'} c_{ij} x_{ij} \tag{2}$$

4.4. Restricciones

$$\sum_{i\in\Delta_j^-} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{3}$$

$$\sum_{j\in\Delta_i^+} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{4}$$

$$\sum_{i\in\Delta_0^-} x_{i0} = Q, \tag{5}$$

$$\sum_{j\in\Delta_0^+} x_{0j} = Q, \tag{6}$$

$$\sum_{j\in S} \sum_{i\in\Delta_j^-\setminus S} x_{ij} \geq r(S) \qquad \forall S \in \mathcal{F}, \tag{7}$$

$$\sum_{i\in S} \sum_{j\in\Delta_i^+\setminus S} x_{ij} \geq r(S) \qquad \forall S \in \mathcal{F}, \tag{8}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A', \tag{9}$$

Las restricciones (3) y (4) aseguran que un cliente sea visitado exáctamente una vez, mientras que las restricciones (5) y (6) imponen que Q vehículos deban entrar y salir del depósito. Por otro lado, las restricciones (7) y (8), conocidas como las *rounded capacity constriants (RCC)* imponen las condiciones de conectividad y capacidad, garantizando la eliminación de *subtours*. Finalmente, la restricción (9) define el dominio de las variables.

5. Representación

A continuación se detalla la representación utilizada en el desarrollo del código fuente, para encontrar una solución al problema VRPB utilizando backtracking.

5.1. Formato de entrada

Para ilustrar el procedimiento, se utilizará un ejemplo reducido de las instancias entregadas para facilitar la comprensión del lector, conservando los formatos, y comenzando por la entrada.

5			
0	1	120.0	160.0
1	2	215.0	248.0
1	3	15.0	31.0
2	4	185.0	20.0
2	5	209.0	220.0
2	1000.0		
2	230.0		
3	326.0		
4	105.0		
5	237.0		

Cuadro 1: Formato de Entrada

La primera línea del formato de entrada [Tabla 1] representa una cantidad N de nodos a considerar en la instancia. Las siguientes N líneas detallan (en orden): Tipo de Nodo (0 si es depósito, 1 si es Linehaul, 2 si es Backhaul), identificador, coordenada en x, coordenada en y. Luego, se detalla la cantidad Q de vehículos, junto con su capacidad máxima. Finalmente, se listan sin considerar el depósito, los N-1 nodos con sus respectivas demandas.

Esta información en la implementación se almacena de la siguiente manera: los nodos son almacenados en una estructura que empaqueta identificador, tipo, coordenadas en x, coordenadas en y, y demanda del nodo. Además, todos los nodos excepto el depósito son agrupados en dos arreglos: uno para Linehauls y otro para Backhauls, de tal manera que permitan su fácil acceso y distribución al momento de la creación de las Q rutas factibles utilizando backtracking.

$$deposito = n_0 \qquad L = [n_{L1}, n_{L2}, ..., n_{Ln}] \qquad B = [n_{B1}, n_{B2}, ..., n_{Bn}] \tag{10}$$

Los vehículos son agrupados en un arreglo de tamaño Q, y representados en una estructura que agrupa los siguientes atributos: identificador, capacidad, distancia recorrida, demanda Linehaul, demanda Backhaul y una ruta, la cual es la representación escogida para una solución candidata.

$$Vehiculos = [V_1, V_2, ..., V_Q] \qquad \forall j \in 0, Q \tag{11}$$

Una ruta es un arreglo de nodos donde cada valor representa el objeto del i-ésimo nodo, y tiene la particularidad de comenzar y terminar en el nodo depósito (n_1), tal de obligar al algoritmo a efectuar ciclos cerrados para siempre cumplir con las restricciones 6 y 7 planteadas en la sección 4.

$$ruta = [n_1, n_2, ..., n_N, n_1] \qquad \forall i \in 0, N \tag{12}$$

Una solución candidata, es aquella que considera un conjunto de rutas que no superan la capacidad de los vehículos, donde los nodos (o clientes) son visitados una única vez por cada

vehículo. Para cumplir con la representación deseada, se implementa una colección de Q rutas a través de un arreglo, la cual recibe rutas determinadas por los procesos de backtracking y eliminación de los nodos del EDB descritos en la siguiente sección.

$$rutas = [ruta_1, ruta_2, ..., ruta_Q] \qquad \forall j \in 0, Q \tag{13}$$

Se ha elegido esta representación, ya que permite almacenar todas las variables necesarias, encapsulándolas en un tipo de dato que permite preservar el orden de instanciación, facilitando los cálculos necesarios para cumplir con las restricciones de capacidad y la necesidad de minimizar la distancia total de la ruta, logrando representar satisfactoriamente cualquier solución del espacio de soluciones factible de forma eficiente.

La estructura por si sola no reduce el EDB, pero permite efectuar todas las operaciones necesarias sobre el espacio para reducirlo, por ejemplo, al eliminar de la lista de nodos todos aquellos pertenecientes a una ruta solución.

6. Descripción del algoritmo

La idea detrás del algoritmo es simple: hacer backtracking sobre todos los nodos y obtener una ruta factible que cumpla con las restricciones, para luego descartar los nodos que compongan esta ruta del espacio de búsqueda, e iterar el proceso hasta que no queden nodos por cubrir.

Algorithm 1 Proceso General

```
d : nododepósito
L : nodosLinehaul
B : nodosBackhaul
i ← 0
Q ≥ 0
rutas ← []
while i ≤ Q and L is not empty do
    V_i : Vehiculo i
    ruta ← [d]
    rutaL ← Backtracking(L, d, V_i, ruta)
    L \ rutaL
    if B is not empty then
        rutaFinal ← Backtracking(B, d, V_i, rutaL)
        B \ rutaFinal
    else
        rutaFinal ← rutaL
    end if
    rutas ← rutaFinal
end while
return rutas
```

Primeramente, se itera sobre el conjunto de nodos linehaul buscando el primer nodo que al ser instanciado en la ruta actual no viole la restricción de capacidad impuesta por el vehículo, hasta que no exista un nodo con demanda suficientemente pequeña como para ser añadido. De esta manera, el algoritmo asegura seleccionar solamente rutas factibles.

Una vez completada la asignación de nodos tipo linehaul, en caso de existir nodos backhaul disponibles, se instancian iterativamente en el orden entregado aplicando el procedimiento anteriormente descrito, buscando los nodos backhaul hasta completar la capacidad de carga o que no existan nodos backhaul. Finalmente, se agrega el nodo depósito a la ruta para completar el ciclo cerrado.

Una vez encontradas todas las rutas factibles (o soluciones candidatas) mediante backtracking, se procede a buscar la ruta de menor distancia dentro del conjunto de rutas obtenido y se asigna a un vehículo de forma definitiva, eliminando los nodos utilizados de los conjuntos L y B, para posteriormente repetir el proceso de backtracking hasta que no queden vehículos con rutas por asignar ni nodos disponibles.

Al reducir los conjuntos L y B se reduce iterativamente el tamaño del espacio de búsqueda, convergiendo a una solución candidata de forma más rápida a medida que se van asignando rutas a los diferentes vehículos, y cumpliendo el requisito de unicidad de vehículos por nodo.

Se decidió relajar la condición de que solo pueden ser visitados una sola vez por un vehículo hasta el final de la creación de las rutas factibles, dado que el algoritmo puede demorar mucho tiempo en obtener todas las soluciones (al estar basado en una técnica completa) e incluir este paso dentro del backtracking solo dificulta más la situación. Para evitar esta situación, se implementa un contador de tiempo que permite detener la ejecución en un tiempo T deseado.

La idea de todos estos pasos descritos anteriormente, es progresivamente ir reduciendo el espacio de búsqueda; filtrando aquellas soluciones que incumplan la capacidad de carga de los vehículos, filtrando por la distancia total de las rutas y finalmente por la restricción de unicidad de rutas para cada nodo.

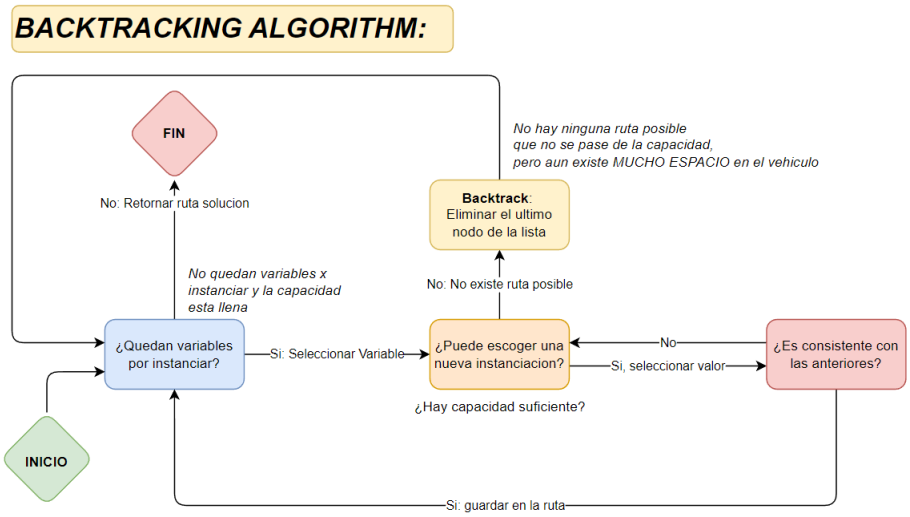

Figura 1: Algoritmo de Backtracking

7. Experimentos

En esta sección, se busca poner a prueba la representación y el algoritmo descrito anteriormente. Luego de una búsqueda sin éxito en la literatura sobre algún algoritmo que implemente la técnica estudiada en este documento para utilizar sus resultados a modo de benchmark, se propone implementar una variación en la selección de variables a instanciar para estimar cuanto puede afectar en el comportamiento de este.

La variación propuesta consiste en modificar la forma en como se seleccionan los valores a instanciar en las rutas; en vez de instanciar el primer nodo que no exceda la capacidad máxima del vehículo al añadirlo a la ruta, se busca el nodo que minimice la distancia de todas las posibles instanciaciones a asignar, calculando la distancia euclidiana entre el nodo previamente instanciado (en un principio, el depósito) y el nodo actual, conservando el valor de distancia mínima en cada nivel del árbol de búsqueda, hasta que no exista un nodo con demanda suficientemente pequeña como para ser añadido sin superar la carga máxima. De esta manera, el algoritmo asegura seleccionar el nodo más cercano al paso anterior, sin romper la restricción de capacidad.

Algorithm 2 Backtracking Iterativo

```
L_actual ← 0
alto ← 1
Lista                                                    ▷ Lista Linehaul/Backhaul
MaxIteraciones ← 9999
min_distancia ← 1e + 08
while alto ≥ 1 do
    if MaxIteraciones then
        break
    end if
    if L_actual < Lista.size() then
        Actual ← Lista[L_actual]
        Anterior ← ruta.back()
        if no supera capacidad then
            if no está en ruta posible then
                distancia = dist(Anterior, Actual)
                if min_distancia > distancia then                          ▷ Variación
                    min_distancia ← distancia
                    NodoInstanciar ← Actual
                end if
            end if
            L_actual + +
        else
            L_actual + +
        end if
    else                                          ▷ Instanciar nodo seleccionado
        if no supera capacidad & no está en ruta posible then
            ruta ← NodoInstanciar
            alto + +
            L_actual ← 0
            min_distancia ← 1e + 08
        else                                                  ▷ Capacidad maxima
            rutasPosibles ← ruta
            alto − −
            L_actual ← 0
            min_distancia ← 1e + 08
        end if
    end if
end while
return ruta ← Seleccionar.mindist(rutas)
```

El algoritmo descrito representa la variación explicada anteriormente. Para evitar esta variación, solo hace falta eliminar la condición de mínima distancia detallada en el pseudocódigo. Cabe destacar que ambas implementaciones seleccionan únicamente nodos del vecindario de nodos no pertenecientes a soluciones candidatas similares a la ruta actualmente instanciada.

Para comparar ambos algoritmos, se seleccionaron 5 instancias; GA1 y GA2 las cuales se considera representan un EDB pequeño, GG1 de tamaño mediano con 57 clientes y GJ1/GN6, dos instancias que representan los espacios de búsqueda más grandes de los encontrados en el conjunto de instancias entregados para hacer pruebas.

Para medir la calidad de las soluciones, se ha efectuado un cálculo simple en cada solución para obtener la distancia total recorrida por todas las rutas, sumando la distancia recorrida de cada ruta perteneciente a la solución. Además, se considera el tiempo de ejecución de cada algoritmo. Es importante mencionar que se tiene en cuenta el tiempo de ejecución para hacer comparaciones entre los órdenes de magnitud que se obtienen, y no como un factor determinante en la calidad de la solución, dado que el algoritmo está diseñado para terminar su ejecución con una cantidad máxima de iteraciones, por lo que no resulta un indicador muy preciso.

De esta forma, se determina que variación del algoritmo produce mejores resultados, los cuales se encuentran en la siguiente sección.

8. Resultados

A continuacion, se presentan los resultados de los experimentos para las instancias seleccionadas:

		Original		**Variacion**	
Instancia	*Nodos*	*Distancia*	*Tiempo [s]*	*Distancia*	*Tiempo [s]*
GA1	25	377075	7.11406	281886	5.43553
GA2	25	385535	3.81948	287930	7.09214
GG1	57	780054	8.35902	444832	8.2739
GJ1	94	1.25819e+06	7.42522	531898	8.12193
GN6	150	2.08909e+06	3.48865	546221	3.96939

Cuadro 2: Resultados Algoritmo original vs variación

En la tabla se presentan los resultados para las instancias seleccionadas, donde la columna *Distancia* corresponde a la distancia total recorrida por todos los vehículos correspondientes a la solución obtenida, y la columna *Tiempo[s]* considera el tiempo empleado en la resolución.

Basándose en los resultados expuestos en el Cuadro 2, se puede notar que al seleccionar el mejor vecino dentro de los nodos no pertenecientes a soluciones candidatas en este punto, se obtienen mejores soluciones en tiempos similares para cada una de las instancias.

Se considera el tiempo de ejecución para mostrar que ambos algoritmos resuelven el problema en tiempos de orden de magnitud similar, dado que ambos están acotados por un máximo de iteraciones impuesto como decisión de diseño.

Estos resultados demuestran que al alterar el criterio de selección, sin modificar el orden de instanciación ni el procedimiento general de Backtracking, se logra obtener resultados de mayor calidad en tiempos similares, respondiendo al objetivo planteado en la sección 7.

9. Conclusiones

Como se hizo notar en la sección 3 existen técnicas que resuelven diferentes variantes del problema, pero todas las estudiadas en este documento al menos incluyen la resolución del VRPB estándar, resolviendo todas el mismo problema, utilizando las mismas instancias de benchmark. Se puede apreciar también que la mayoría de las técnicas estudiadas corresponden a heurísticas constructivas y de búsqueda local, dado que VRPB es un problema NP-hard [4]. Por otro lado, el tamaño de las instancias es influyente en los resultados de las técnicas, siendo las de tipo exacto las más afectadas por este parámetro, obteniendo peores resultados en general.[7]

Respecto a la propuesta de solución implementada en este documento, cabe destacar que la forma de almacenar los datos entregados por las instancias y la propuesta de solución candidata representan correctamente el problema, logrando encontrar solución a un total de 41 instancias de las 62 entregadas para este problema gracias a la implementación del algoritmo descrito en la sección 6, obteniendo para cada instancia un conjunto de rutas que cumplen con todas las restricciones del problema.

Como *Backtracking* es una técnica completa, fue necesario acotar la busqueda de soluciones implementando una cota maxima de iteraciones, para asi asegurar un tiempo de ejecucion total aceptable.

Por los resultados obtenidos en la seccion de Experimentos, se puede concluir que alterar el criterio de seleccion de las variables a instanciar puede resultar beneficioso para obtener soluciones de mayor calidad en tiempos de ejecucion similares, condiciendo el supuesto inicial.

Estudios posteriores deberían enfocarse en combinar heurísticas que han resultado beneficiosas para el estudio de VRPB como *population search*, *adaptive large neighborhood search* o *iterated local search*, dado que a la fecha no existen investigaciones al respecto. Además, no se han reportado estudios respecto a un modelo de aproximación contínua para VRPB, ni métodos estocásticos que resuelvan el problema. Respecto a las metodologías exactas, a día de hoy no se ha propuesto ningún algoritmo exacto que resuelva la extensión de intervalos temporales de VRPB, VRPBTW.

Resulta apropiado decir que el campo de los VRPBs aún tiene mucho por desarrollar, dados los avances en métodos heurísticos y exactos para VRP, así como los que vemos día a día en la tecnología.

Referencias

[1] INE, "Envíos de encomiendas aumentan 81,2 % en comparación con el mismo mes del año pasado," *(visitado el 29.09.2022)*, Junio 2021. [Online]. Available: https://www.ine.cl/prensa/2021/08/02/junio-2021-env%C3%ADos-de-encomiendas-aumentan-81-2-en-comparaci%C3%B3n-con-el-mismo-mes-del-a%C3%B1o-pasado

[2] J. H. Dantzig, G B; Ramser, "The truck dispatching problem," *Management science*, vol. 6, pp. 80–91, 1959.

[3] C. J.-B. Marc Goetschalckx, "The vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. volume 42, pp. 39–51, 1989.

[4] P. Toth and D. Vigo, "An overview of vehicle routing problems," *The vehicle routing problem*, pp. 1–26, 2002.

[5] S. Wade, A. C.; Salhi, "An investigation into a new class of vehicle routing problem with backhauls." *Omega*, vol. 30, pp. 479–487, 2002.

[6] G. Salhi, S.; Nagy, "A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling." *Journal of the Operational Research Society*, vol. 50, pp. 1034–1042, 1999.

[7] Ç. Koç and G. Laporte, "Vehicle routing with backhauls: Review and research perspectives," *Computers & Operations Research*, vol. 91, pp. 79–91, 2018.

[8] L. Deif, I.; Bodin, "Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling." *Proceedings of the Babson conference on software uses in transportation and logistics management*, pp. 75–96, 1984.

[9] C. Jacobs-Blecha and M. Goetschalckx, "The vehicle routing problem with backhauls: properties and solution algorithms," *Atlanta: Georgia Tech Research Corporation*, 1992.

[10] J. Brandao, "A new tabu search algorithm for the vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. 173, no. 2, pp. 540–555, 2006.

[11] Y. Gajpal and P. L. Abad, "Multi-ant colony system (macs) for a vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. 196, no. 1, pp. 102–117, 2009.

[12] E. Queiroga, Y. Frota, R. Sadykov, A. Subramanian, E. Uchoa, and T. Vidal, "On the exact solution of vehicle routing problems with backhauls," *European Journal of Operational Research*, vol. 287, no. 1, pp. 76–89, 2020.