Sección de Matemáticas
Universidad de La Laguna

Brenda Sofía Fernández Barabino

Un problema de rutas periódicas de vehículos con flexibilidad en las entregas

The Flexible Periodic Vehicle Routing Problem

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2023

Dirigido por
Inmaculada Rodríguez Martín

Inmaculada Rodríguez Martín
Matemáticas, Estadística e
Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Resumen · Abstract

Resumen

Esta memoria aborda el Problema Periódico y Flexible de Rutas de Vehículos (FPVRP), en el que un transportista tiene que establecer un plan de distribución para servir a sus clientes a lo largo de un horizonte temporal de planificación. Cada cliente tiene una demanda total que debe ser atendida en el horizonte temporal y un límite en la cantidad máxima que se puede entregar en cada visita. Se dispone de una flota homogénea de vehículos con capacidad limitada para realizar los servicios, y el objetivo es minimizar el costo total de las rutas. El FPVRP puede verse como una generalización del Problema Periódico de Rutas de Vehículos (PVRP) que, en cambio, tiene frecuencias y calendarios de servicio fijos y donde la cantidad a entregar a cada cliente en cada visita es fija. El PVRP es en sí mismo una generalización del clásico Problema de Rutas de Vehículos (VRP). Mostramos una formulación matemática para el FPVRP, junto con algunas desigualdades que sirven para fortalecerla. Codificamos el modelo en Python y presentamos algunos resultados computacionales.

Palabras clave: problemas de rutas periódicas, flexibilidad.

Abstract

This memory addresses the Flexible Periodic Vehicle Routing Problem (FPVRP), where a carrier has to establish a distribution plan to serve his customers over a planning horizon. Each customer has a total demand that must be served within the horizon and a limit on the maximum quantity that can be delivered at each visit. A fleet of homogeneous capacitated vehicles is available to perform the services, and the objective is to minimize the total routing cost. The FPVRP can be seen as a generalization of the Periodic Vehicle Routing Problem (PVRP), which instead has fixed sevice frequencies and schedules and where the quantity to be delivered to each customer in each visit is fixed. The PVRP is itself a generalization of the classical Vehicle Routing Problem (VRP). We show a mathematical formulation for the FPVRP, together with some inequalities that serve to strengthen it. We code the model in Python and present some computational results.

Key words: periodic vehicle routing, flexibility.

Contenido

Resumen/Abstract III

Introducción VII

1. Fundamentos teóricos 1
1.1. Teoría de grafos 1
1.1.1. Conceptos básicos 1
1.2. Programación matemática y optimización 3
1.2.1. Optimización 3
1.2.2. Programación matemática 3
1.2.3. Algoritmos para la resolución de problemas de Programación Lineal Entera o Mixta 5

2. Problemas clásicos de rutas 7
2.1. El Problema del Viajante de Comercio (TSP) 7
2.1.1. Planteamiento del problema para el caso asimétrico 9
2.2. El Problema de Rutas de Vehículos (VRP) 11
2.2.1. Planteamiento del problema para el caso asimétrico 11
2.3. Algunas variantes del Problema de Rutas de Vehículos 14
2.3.1. Problema Periódico de Rutas de Vehículos (PVRP) 14
2.3.2. Problema de Rutas de Vehículos con Inventario (IRP) 15
2.3.3. Flexibilidad en los problemas de rutas 16

3. El Problema Periódico y Flexible de Rutas de Vehículos 19
3.1. Descripción del problema 19
3.2. Modelo matemático 20
3.3. Restricciones que refuerzan el modelo 22

4. Experimentos computacionales 25
4.1. Material y herramientas de trabajo 25
4.2. Instancias 26

4.3. Ajustes 27
4.4. Comparación entre PVRP y FPVRP 28
4.5. Resultados completos 31
4.5.1. Consideraciones adicionales 34

A. Apéndice 37
A.1. Código Python del FPVRP 37

Bibliografía 41

Introducción

Los problemas de rutas de vehículos (VRP, por si siglas en inglés) tienen por objetivo diseñar un plan de distribución en el que se utiliza una flota de vehículos para entregar (o recoger) mercancías a un conjunto de clientes. En estos problemas se atiende a los clientes en un solo periodo, lo que significa que las operaciones de entrega comienzan al comienzo del periodo y deben terminar al final del mismo. Sin embargo, varias aplicaciones del mundo real conllevan dar servicio a clientes que tienen unas demandas periódicas a lo largo de un horizonte de tiempo determinado. Por ejemplo, puede aparecer periodicidad en servicios como la entrega de comestibles, la recolección de basuras, o la programación de visitas a domicilio a pacientes. En este contexto aparece el Problema Periódico de Rutas de Vehículos (PVRP), que es un problema de rutas de vehículos en el que el servicio a los clientes debe brindarse durante múltiples periodos. Se supone que los clientes deben ser atendidos con cierta frecuencia, eligiendo un calendario de visitas de entre un conjunto determinado, y deben recibir una cantidad fija en cada visita. El problema es elegir el calendario de visitas de cada cliente y organizar las rutas de los vehículos. El PVRP implica que las frecuencias de las visitas a los clientes, los calendarios de visitas a lo largo del horizonte temporal y la cantidad de mercancía entregada (o recogida) a cada cliente en cada visita, son fijos. Sin embargo, dado que uno de los criterios más importantes en este tipo de problemas es la minimización de los costos de transporte, la incorporación de políticas de flexibles de servicio puede generar ahorros significativos. Con el término "política flexible de servicio" nos referimos a políticas de servicio donde la frecuencia de las visitas a cada cliente así como las cantidades entregadas no están determinadas a priori. Por tanto, hay que decidir los periodos de tiempo en los que se atenderá a cada cliente y la cantidad a entregar en cada visita. Esta es la motivación del problema que estudiamos en esta memoria, el Problema Periódico y Flexible de Rutas de Vehículos (FPVRP), propuesto por Archeti et al. [3].

En el FPVRP, los clientes tienen una demanda total que se debe satisfacer a lo largo del horizonte de planificación. La cantidad entregada en cada visita no

debe exceder la capacidad del cliente, que suele ser inferior a la demanda total. Por lo tanto, se realizan múltiples visitas a cada cliente durante el horizonte de planificación. El FPVRP puede verse como una extensión del PVRP donde no se establece una frecuencia ni un calendario fijos de visitas. En cambio, si la capacidad del cliente corresponde al ratio entre la demanda total y la frecuencia de visitas establecida en el PVRP, entonces el FPVRP se convierte en una generalización del PVRP donde cada cliente debe ser visitado un número de veces que sea al menos igual a la frecuencia del PVRP. Además, se debe decidir la cantidad entregada en cada visita. Esto claramente aumenta la flexibilidad con respecto a la configuración del PVRP y puede producir ahorros de costos.

El objetivo del presente trabajo es estudiar el FPVRP. La memoria se organiza de la siguiente forma. En el Capítulo 1 se hace un repaso de conceptos básicos de teoría de grafos y programación matemática, que nos servirán luego para describir formalmente el problema. El capítulo 2 pretende poner en contexto el FPVRP dentro de los problemas de rutas. En el capítulo 3 está dedicado al FPVRP, para el que se presenta un modelo matemático de programación lineal entera mixta, así como algunas familias de restricciones que sirven para reforzar dicho modelo. Este modelo ha sido implementado en Python, y en el Capítulo 4 mostramos los resultados de nuestros experimentos computacionales. Acabamos con algunas conclusiones.

1

Fundamentos teóricos

Para comprender el problema que abordamos en este trabajo, necesitamos conocer algunas herramientas básicas, como pueden ser la optimización combinatoria y la de teoría de grafos.

1.1. Teoría de grafos

La teoría de grafos es una rama de las matemáticas y la informática que estudia las propiedades de los grafos. Los orígenes de esta teoría están con un estudio realizado por el matemático suizo Leonhard Euler en 1736. El desafío era encontrar una ruta eficiente para cruzar todos los puentes de la ciudad de Königsberg pasando una única vez por cada puente de forma que se llegara al punto de partida.

La teoría de grafos intenta representar visualmente conjuntos de datos abstractos en términos de nodos o vértices, y las uniones o relaciones entre éstos a través de aristas. En las matemáticas se utiliza para diversas aplicaciones, aunque nosotros nos centraremos en su uso para la solución de problemas de rutas de vehículos con restricciones.

Vamos a enunciar a continuación algunas definiciones.

1.1.1. Conceptos básicos

Un grafo dirigido (digrafo) es una pareja de conjuntos $G = (V, A)$, donde $V = \{1, \ldots, n\}$ es un conjunto de vértices o nodos y $A \subseteq \big\{(i, j) : (i, j) \in V, i \neq j\big\}$ es un conjunto de pares ordenados de vértices denominados arcos. Dado un arco $a = (i, j) \in A$, i es el origen (cola) y j es el destino (cabeza) del arco. Por ejemplo, sea

$$V = \big\{a, b, c, d\big\} \text{ y } A = \big\{(a, b), (b, c), (b, d), (d, b), (d, a)\big\}$$

el grafo resultante es

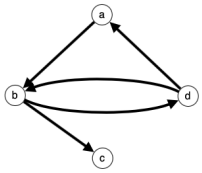


Figura 1.1: Grafo dirigido.

Para un grafo dirigido $G = (V, A)$, los conjuntos de arcos que salen y entran en un vértice i se denotan por $\delta^+(i)$ y $\delta^-(i)$ respectivamente. El grado exterior de i se define como $|\delta^+(i)|$, y grado interior como $|\delta^-(i)|$. Así mismo, dado un subconjunto de vértices $S \subset V$, definimos los conjuntos de arcos $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$, $\delta^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$ y $A(S) = \{(i,j) \in A : i \in S, j \in S\}$.

Un camino es una secuencia de arcos de A donde el vértice destino de un arco coincide con el vértice origen del siguiente arco. Por ejemplo, $\big\{(a,b),(b,d),(d,a)\big\}$. Un camino Hamiltoniano es un camino que visita todos los vértices de un grafo una vez. Si el primer y el último vértice visitado coinciden, se denomina ciclo Hamiltoniano.

Decimos que $G' = (V', A')$ es un subgrafo del grafo $G = (V, A)$ si $V' \subseteq V$, $A' \subseteq A$ y todos los arcos de A' tienen sus extremos en V'.

Un grafo no dirigido $G = (V, E)$ es una pareja de conjuntos, donde $V = \{1, \ldots, n\}$ es un conjunto finito de nodos y $E \subseteq \big\{\{i,j\} : i, j \in V, i < j\big\}$ es un conjunto no ordenado de pares de vértices denominados ejes. Es decir, los ejes no tienen dirección. Un grafo no dirigido es simple si exsite a lo sumo un eje entre cualquier par de vértices, y es completo si es simple y cada par de vértices está conectado por un eje. Un grafo no dirigido se puede transformar en uno dirigido sustituyendo cada eje por dos arcos con direcciones contrarias. Las definiciones que dimos anteriormente para grafos dirigidos se corresponden con otras análogas para grafos no dirigidos.

1.2. Programación matemática y optimización

En matemáticas, estadísticas, ciencias empíricas, ciencia de la computación o economía, la optimización (o programación matemática) es la selección del mejor elemento de un conjunto de elementos disponibles con respecto a algún criterio.

1.2.1. Optimización

La optimización es la acción de desarrollar una actividad lo más eficientemente posible, es decir, con la menor cantidad de recursos y en el menor tiempo posible. En las últimas décadas, el término optimización se ha vinculado al mundo de la informática. Sin embargo, es un concepto que también se utiliza en las matemáticas, en la gestión de procesos y la economía.

Desde el punto de vista de la matemática, optimizar significa elegir el mejor de los elementos que pertenecen a un conjunto. Es decir, encontrar el valor que deben tomar las variables de decisión (sujetas a restricciones) para que una función objetivo alcance un máximo o un mínimo dependiendo de nuestro propósito principal.

1.2.2. Programación matemática

La programación matemática constituye un campo amplio de estudio que incluye teoría, aplicaciones y técnicas computacionales para problemas de optimización, y forma parte del área de conocimiento de la Investigación Operativa. Tiene como objetivo principal encontrar una solución que minimice o maximice una función objetivo al tiempo que satisface una serie de limitaciones que se corresponden con restricciones formuladas como ecuaciones o inecuaciones.

La programación es una poderosa técnica de modelado utilizada en los procesos de toma de decisiones. Para resolver tales problemas podríamos decir que se siguen tres pasos:

1. Identificar las posibles decisiones que se pueden tomar, o, lo que es lo mismo, identificar las variables propias del problema. Por lo general, las variables son cuantitativas y se buscan los valores que optimizan el objetivo.
2. Determinar qué decisiones son aceptables, lo que se traduce en crear un conjunto de restricciones.
3. Calcular los valores asociados a cada decisión admisible. Esto determina los costos o beneficios de las variables en la función objetivo que debe ser minimizada o maximizada.

Con carácter general, este problema se puede representar de la siguiente manera:

$$min\left\{f(x) : x \in S\right\}$$

donde $S \subseteq \mathbb{R}^n$ y $f : S \to \mathbb{R}$, entendiendo S como el conjunto donde la función f alcanza el valor mínimo que se busca. De esta forma, f se considera la función objetivo, a S se le denomina región factible y cada elemento de esta región es una solución factible. El conjunto de todos estos elementos define nuestro problema de optimización.

Si estamos interesados en maximizar nuestra función objetivo, únicamente tenemos que tener en cuenta la siguiente igualdad:

$$max\left\{f(x) : x \in S\right\} = -min\left\{-f(x) : x \in S\right\}$$

Dependiendo de las características de las ecuaciones/inecuaciones, las variables y la función objetivo del modelo, la Programación Matemática se puede clasificar de las siguientes formas:

1. Desde el punto de vista de la linealidad de la función objetivo y de las ecuaciones/inecuaciones, se tratará de un problema de **Programación lineal** en el caso de estas ser lineales. En caso contrario, se considerará de **Programación no lineal**.
2. Considerando las características de las variables diferenciamos tres casos. En primer lugar, cuando todas las variables pueden tomar valores reales, se denomina **Programación continua**. En cambio, cuando las variables sólo toman valores enteros se trata de **Programación entera**. La **programación mixta** surge como una combinación de las dos anteriores, es decir, se da el caso del uso de variables continuas y enteras.

Por otro lado, también se puede clasificar los problemas según los resultados obtenidos:

1. **Problema óptimo:** se da cuando existe una solución $x^* \in S$ tal que $f(x^*) \leq f(x)\ \forall x \in S$. A x^* se la denomina solución óptima y se cumple que $min\{f(x) : x \in S\} = f(x^*)$.
2. **Problema no acotado:** esto se da cuando $\forall N \in \mathbb{R}$ existe un $x^* \in S$ tal que $f(x) < N$, con lo que $min\{f(x) : x \in S\} = -\infty$. Es decir, cuando se encuentran soluciones que hacen disminuir la función objetivo de forma infinita.
3. **Problema no factible:** se da cuando $S = \emptyset$, es decir, no existe ninguna solución factible para el problema.

Una rama importante de la programación matemática es la **Optimización Combinatoria**, que estudia problemas que tienen un número finito, aunque normalmente elevado, de soluciones factibles, es decir, problemas del tipo $min\{f(x) : x \in S\}$ con $|S| < \infty$.

1.2.3. Algoritmos para la resolución de problemas de Programación Lineal Entera o Mixta

Los algoritmos exactos que resuelven este tipo de problemas tratan de reducir el tamaño del espacio de soluciones para explorarlo de manera eficiente. A continuación describimos a grandes rasgos algunos de estos algoritmos.

1.2.3.1. Ramificación y acotación

Un algoritmo de ramificación y acotación (*branch and bound*) es un método de resolución de problemas de programación lineal entera que se basa en la creación de un árbol de búsqueda. El objetivo del algoritmo es encontrar la solución óptima del problema a través de la exploración de todas las posibles soluciones.

En el nodo raíz del árbol de búsqueda se sitúa el problema original. Se comienza por resolver la relajación lineal de este problema (es decir, el problema sin las restricciones de integralidad sobre las variables). Si la solución obtenida es entera, paramos; si no, se crean dos nuevos subproblemas hijos eligiendo una variable que tenga un valor fraccionario y añadiendo restricciones sobre el valor que puede tomar esa variable (ramificación). Cada nuevo subproblema es un nodo del árbol de búsqueda. Las soluciones de las relajaciones lineales de estos subproblemas proporcionan, en el caso de ser enteras, soluciones factible y cotas superiores para la solución del problema original (si es de minimizar); si no son enteras proporcionan cotas inferiores para el valor óptimo de cada subproblema. Si la cota inferior de un subproblema es mayor o igual que el valor de la mejor solución factible encontrada hasta el momento, ese nodo del árbol se poda, es decir no se ramifica más a partir de él. También se para la exploración de un nodo cuando el correspondiente problema es no factible.

El algoritmo termina cuando no quedan nodos por explorar, y con la garantía de que la mejor solución factible encontrada es la solución óptima del problema original. En resumen, un algoritmo de ramificación y acotación se basa en la exploración de todas las posibles soluciones de un problema de programación lineal entera a través de la creación de un árbol de búsqueda, y la utilización de técnicas de acotación para eliminar soluciones inviables y reducir el espacio de búsqueda

1.2.3.2. Hiperplanos de corte

Un algoritmo de hiperplanos de corte (*cutting planes*) en un primer paso relaja las condiciones de integralidad sobre las variables del problema entero, y resuelve el programa lineal resultante. Si el programa lineal es infactible, el programa entero también lo es. Si la solución óptima del programa lineal cumple las condiciones de integralidad, se ha encontrado un óptimo del problema entero.

En caso contrario, se busca identificar desigualdades lineales que sean violadas por la solución fraccionaria y sean válidas para las soluciones enteras factibles. A estas desigualdades se las denomina cortes, y al proceso de generarlas se le denomina problema de separación. Los cortes generados son añadidos al problema entero original. El proceso se repite hasta que se encuentra una solución óptima entera para el problema original. En cada iteración, se generan nuevos cortes y se agregan al problema lineal entero, lo que reduce el espacio de búsqueda de soluciones factibles.

El éxito del algoritmo depende en gran medida de la eficiencia de los métodos para encontrar desigualdades violadas que puedan ser agregadas a la formulación para separar las soluciones fraccionarias. Existen cortes generales aplicables a cualquier problema de programación lineal entera, como los cortes de Gomory, pero normalmente lo más eficiente es usar cortes deducidos de la estructura particular de cada problema.

1.2.3.3. Ramificación y corte

Un algoritmo ramificación y corte (*branch and cut*) es un algoritmo de ramificación y acotación en el cual se generan hiperplanos de corte en cada subproblema del árbol de búsqueda. El objetivo de esto es reducir significativamente el número de nodos del árbol. Generalmente, la cota producida en cada nodo del árbol de búsqueda es mejor que en un *branch and bound*, debido a las nuevas desigualdades agregadas a la formulación del correspondiente subproblema, y esto hace que el algoritmo sea más eficiente.

2

Problemas clásicos de rutas

En la gestión del transporte y distribución de mercancías, encontrar las mejores rutas para los vehículos es un aspecto de crucial relevancia. Una mala gestión puede generar altos incrementos en los costos operativos, y afectar a la competitividad de las empresas y la percepción de la calidad del servicio por parte de los clientes. Algunos de los costos más significativos en los que incurre una empresa incluyen los costos de transporte (principalmente generados por el precio del combustible) y los costos salariales asociados. El objetivo principal es minimizarlos. Algunas aplicaciones de este tipo de problemas surgen en empresas de distribución de productos, servicios de transporte, entregas periódicas, recolección de residuos, y muchos otros.

En este capítulo se definen y modelan dos problemas clásicos de optimización combinatoria que resuelven las anteriores necesidades: el Problema del Viajante de Comercio y el Problema de Rutas de Vehículos. Luego presentamos algunas variantes del segundo que aparecen cuando el diseño de las rutas tiene que abarcar varios periodos de tiempo. En cualquier caso, el objetivo principal es optimizar un conjunto de rutas de transporte que deben ser recorridas por uno o más vehículos que parten de un mismo depósito.

2.1. El Problema del Viajante de Comercio (TSP)

El Problema del Viajante de Comercio, en inglés *Travelling Salesman Problem* (TSP) es, si no el que más, uno de los problemas de rutas más estudiados dentro del campo de la optimización combinatoria.

Este problema trata de:

"Un viajante quiere visitar n ciudades una y sólo una vez cada una, empezando por una cualquiera de ellas y regresando al mismo lugar del que partió. Supongamos que conoce la distancia entre cualquier par de ciudades. ¿De qué forma debe hacer el recorrido si pretende minimizar la distancia total?" [25]

Los orígenes del problema se remontan a los años 30 del siglo XX, cuando el matemático británico Karl Menger planteó el problema de encontrar la ruta más corta para recorrer un conjunto de ciudades en una red ferroviaria. En la década de 1950, el problema se reformuló como el problema del vendedor ambulante o viajante de comercio, y se popularizó gracias a la obra de George Dantzig y otros investigadores en optimización combinatoria.

Desde entonces, el problema del viajante de comercio se ha convertido en un clásico en la teoría de la complejidad computacional y ha sido objeto de una intensa investigación en matemáticas, informática y ciencias de la administración, debido a su importancia en la optimización de rutas y la planificación de la logística en una variedad de campos, incluyendo la logística empresarial, el transporte y la planificación de rutas en la industria de la aviación y la planificación urbana. En la Tabla ?? podemos ver algunos de los récords históricos computacionales que se han conseguido.

Récords históricos computacionales		
Años	**Autores**	**Número de ciudades**
1954	Dantzig, Fulkerson, Johnson [14]	49
1971	Held, Karp [20]	64
1975	Camerini, Fratta, Maffioli	67
1977	Grötschel [17]	120
1980	Crowder, Padberg, Hong [13]	318
1987	Padberg, Rinaldi [22]	532
1987	Grötschel, Holland [18]	666
1987	Padberg, Rinaldi [23]	2.392
1994	Applegate, Bixby, Chvátal, Cook [1]	7.397
1998	Applegate, Bixby, Chvátal, Cook	13.509
2001	Applegate, Bixby, Chvátal, Cook	15.112
2004	Applegate, Bixby, Chvátal, Cook, Helsgaun	24.978
2005	Cook, Espinoza, Goycoolea [12]	33.810
2006	Applegate, Bixby, Chvátal, and Cook [2]	85.900

Tabla 2.1: Mayor tamaño de problemas resueltos del TSP

El Problema del Viajante de Comercio presenta dos variantes dependiendo de la naturaleza de la red, que puede ser un grafo simétrico o asimétrico. En el primer caso, nos encontramos ante un grafo no dirigido, pues existen caminos en ambas direcciones y la distancia entre un par de ciudades es la misma en ambos sentidos. Sin embargo, en el TSP asimétrico, que es el caso más general, puede que las distancias no sean las mismas en todos los casos en ambas direcciones o que directamente no existan caminos en ambas direcciones. En el libro de G. Gutin y A.P. Punnen [19], podemos encontrar más información, en el caso de querer seguir profundizando sobre este problema y sus variantes. Nosotros nos vamos a ceñir a modelizar el caso asimétrico.

2.1.1. Planteamiento del problema para el caso asimétrico

Recordemos el planteamiento general del problema:

Un viajante desea visitar un conjunto de ciudades, partiendo de una de ellas y regresando a la misma ciudad de partida de forma que la distancia recorrida (costo) sea la menor posible. Cada ciudad se visita exactamente una vez.

Se puede representar el problema sobre un grafo dirigido $G = (V, A)$, con $V = \{0\} \cup N$, siendo 0 el depósito u origen del circuito y $N = \{1, \ldots, n\}$ es el conjunto de las demás ciudades. $A \subseteq \{(i, j) : i, j \in V, i \neq j\}$ es el conjunto de arcos. Cada arco $a \in A$ tiene asociado un costo o distancia c_a.

Existen varias formulaciones matemáticas del problema. En todas las que presentamos aquí usamos las siguientes variables de decisión:

$$x_a := \begin{cases} 1, & \text{si el arco } a \in A \text{ forma parte de la ruta del vehículo,} \\ 0, & \text{en otro caso.} \end{cases}$$

Podemos formular el TSP de la siguiente forma:

$$\min \sum_{a \in A} c_a x_a \tag{2.1}$$

Con restricciones de subciclo

Sujeto a:

$$\sum_{a \in \delta^+(i)} x_a = \sum_{a \in \delta^-(i)} x_a = 1 \qquad i \in V \tag{2.2}$$

$$\sum_{a \in A(S)} x_a \leq |S| - 1 \qquad S \subset V \tag{2.3}$$

$$0 \leq x_a \leq 1 \tag{2.4}$$

$$x_a \in \mathbb{Z} \tag{2.5}$$

Con la restricción (2.2) forzamos a que en cada nodo el vehículo entre y salga una única vez. Esto se refuerza con las restricciones (2.4) y (2.5) que fuerzan a que las variables sólo tomen valores binarios. Por último, con la desigualdad (2.3) forzamos que en cualquier subconjunto de nodos contenido estrictamente en V el número de arcos recorridos por el vehículo sea menor que el número de nodos en el subconjunto. De esta forma obligamos a que de ese subconjunto salga el vehículo, evitando la aparición de subtours.

Con variables de potencial

Sujeto a:

$$\sum_{a\in\delta^+(i)} x_a = \sum_{a\in\delta^-(i)} x_a = 1 \qquad i \in V \tag{2.6}$$

$$u_j \geq u_i + x_{ij} - (n-2)(1-x_{ij}) + (n-3)x_{ji} \qquad i,j \in N, i \neq j \tag{2.7}$$

$$0 \leq x_a \leq 1 \tag{2.8}$$

$$x_a \in \mathbb{Z} \tag{2.9}$$

En este caso, hacemos uso de una variable entera u_j para determinar la posición del nodo j en la ruta a establecer.

Como podemos apreciar, respecto al modelo anterior, únicamente hemos intercambiado la restricción (2.3) por (2.7). Con estas segundas desigualdades forzamos a que el orden seguido para visitar las diferentes ciudades sea coherente y evitamos subtours.

Con variables de flujo

Sujeto a:

$$\sum_{a\in\delta^+(i)} x_a = \sum_{a\in\delta^-(i)} x_a = 1 \qquad i \in V \tag{2.10}$$

$$\sum_{j\in V, j\neq i} f_{ji} - \sum_{j\in V, j\neq i} f_{ij} = 1 \qquad i \in N \tag{2.11}$$

$$f_{0i} = nx_{0i} \qquad i \in N \tag{2.12}$$

$$f_{i0} = 0 \qquad i \in N \tag{2.13}$$

$$f_{ij} \leq nx_{ij} \qquad i,j \in N, i \neq j \tag{2.14}$$

$$0 \leq x_a \leq 1 \tag{2.15}$$

$$x_a \in \mathbb{Z} \tag{2.16}$$

Para este planteamiento, debemos introducir la variable continua de flujo f_{ij} que funciona de manera inversa a la de potencial. Es decir, en este caso vamos restándole una unidad cada vez que visitamos una ciudad.

Con la nueva restricción (2.11), conseguimos que se descargue una unidad de flujo cada vez que pasemos por un nodo. En cambio, con (2.12) y (2.14) obligamos a que del nodo de partida se salga con un flujo igual al número de ciudades a visitar y que solo exista flujo entre dos ciudades si existe la conexión entre estas. Por último, con (2.13), no permitimos que se regrese al nodo de partida con flujo.

2.2. El Problema de Rutas de Vehículos (VRP)

El término Problema de Rutas de Vehículos (VRP, por sus sigla en inglés) se utiliza para identificar una clase de problemas enfocados en el diseño de rutas óptimas para una flota de vehículos con una determinada capacidad que parten de un depósito y deben satisfacer la demanda de un conjunto de clientes. La minimización del coste total de transporte es uno de los objetivos más frecuentes en estos problemas. Las restricciones de VRP incluyen aquellas que modelan la suposición de que los vehículos tienen una capacidad limitada, de modo que la demanda total de los clientes visitados en una ruta por un vehículo no exceda la capacidad del mismo. Estos problemas fueron introducidos en 1959 por Dantzig y Ramser [15] en el trabajo *The Truck Dispatching Problem* en el que se buscaba un conjunto de rutas para una flota de vehículos despachadores de combustible, que debían viajar desde un depósito a varias estaciones de servicio para satisfacer su demanda.

El VRP con capacidades es un problema NP-difícil porque es una generalización de dos conocidos problemas combinatorios: el problema del viajante de comercio y el problema del embalaje en contenedores (*Bin Packing Problem*). Debido a la dificultad de encontrar soluciones de buena calidad en tiempos de cómputo pequeños para aplicaciones del mundo real, una cantidad considerable del trabajo actual en VRP trata sobre métodos aproximados para manejar instancias de gran tamaño.

2.2.1. Planteamiento del problema para el caso asimétrico

Formalmente, al igual que en el TSP, el VRP se define sobre un grafo dirigido $G = (V, A)$, con el conjunto de nodos $V = \{0\} \cup N$ donde 0 denota el depósito y $N = \{1, \ldots, n\}$ es el conjunto de clientes, y $A \subseteq \{(i, j) : i, j \in V, i \neq j\}$ es el conjunto de arcos, en el que cada uno tiene asignado un costo c_a. Recordemos que, al tratarse del caso asimétrico, $c_{ij} \neq c_{ji}\ \forall (i, j) \in A$. Además, cada cliente i, $i = 1, \ldots, n$, tiene asociada una demanda $d_i \geq 0$ que debe ser recogida. La demanda total de un conjunto de clientes $S \subseteq V$ se define como

$$d(S) = \sum_{i \in S} d_i.$$

Para el depósito $d_0 = 0$, es decir, se le asocia una demanda nula.

En el depósito se encuentra un conjunto $K = \{1, \ldots, k\}$ de vehículos iguales, todos con la misma capacidad C. Asumimos que $d_i \leq C$ para todo $i \in V$ y que $K \geq K_{\text{mín}}$, para asegurar la viabilidad del problema. K_{min} es el número mínimo de vehículos necesarios para satisfacer la demanda de todos los clientes, y se calcula de la siguiente manera

$$K_{min} = \lceil d(N)/C \rceil$$

A continuación, presentaremos diversas formulaciones para el problema del VRP teniendo en cuenta que, al igual que en el TSP, la variable de decisión asociada a las rutas se define de la siguiente manera:

$$x_a := \begin{cases} 1, & \text{si el arco } a \in A \text{ es atravesado por un vehículo,} \\ 0, & \text{en otro caso.} \end{cases}$$

Formulación de dos índices

El modelo se presenta como

$$\min \sum_{a \in A} c_a x_a$$

sujeto a:

$$\sum_{a \in \delta^+(i)} x_a = \sum_{a \in \delta^-(i)} x_a = 1 \qquad i \in N \qquad (2.17)$$

$$\sum_{a \in \delta^+(0)} x_a = \sum_{a \in \delta^-(0)} x_a = |K| \qquad (2.18)$$

$$\sum_{a \in \delta^+(S)} x_a \geq r(S) \qquad S \subseteq N, S \neq \emptyset \qquad (2.19)$$

$$x_a \in \{0, 1\} \qquad i, j \in V \qquad (2.20)$$

Con la restricción (2.17) imponemos que de cada cliente sólo entre y salga un vehículo. De manera similar, con (2.18) conseguimos que del depósito salgan y regresen $|K|$ vehículos. Por otro lado, con (2.19) evitamos subtours, pues determinamos que cada partición $(S, V \setminus S)$ debe ser atravesada por un número de arcos que no puede ser inferior a $r(S)$ (mínimo de vehículos necesarios para servir al conjunto S). Por último, con (2.20) indicamos que las variables de decisión son binarias.

El número de restricciones (2.19) crece exponencialmente con n, lo que hace prácticamente imposible resolver directamente, dando el modelo completo, para problemas grandes. Sin embargo, podemos evitar esto definiendo otra familia de restricciones con cardinal polinómico, como proponen en [26]. Para ello, definimos la variable continua $u_i, i \in N$ para representar la carga del vehículo después de visitar el cliente i y sustituimos (2.19) por las siguientes restricciones

$$u_j \geq u_i + d_j x_{ij} - C(1 - x_{ij}) \qquad i, j \in N \qquad (2.21)$$

$$d_i \leq u_i \leq C \qquad i \in N \qquad (2.22)$$

Estas nuevas desigualdades imponen las limitaciones de capacidad y conexión del problema.

Formulación de tres índices

La formulación de tres índices permite indicar explícitamente el vehículo que recorre cada arco. Esto permite solventar uno de los problemas que presenta la anterior formulación, que es el de la simetría.

Para este planteamiento usamos las siguientes variables de decisión:

$$x_a^k := \begin{cases} 1, & \text{si el arco } a \in A \text{ es atravesado por el vehículo, } k \\ 0, & \text{en otro caso.} \end{cases}$$

$$y_i^k := \begin{cases} 1, & \text{si el vértice } i \in V \text{ es visitado por el vehículo, } k \\ 0, & \text{en otro caso.} \end{cases}$$

Entonces, el modelo se define de la siguiente manera:

$$\min \sum_{k \in K} \sum_{a \in A} c_a x_a^k$$

sujeto a:

$$\sum_{k \in K} y_i^k = 1 \qquad i \in N \tag{2.23}$$

$$y_0^k = 1 \qquad k \in K \tag{2.24}$$

$$\sum_{a \in \delta^+(i)} x_a^k = \sum_{a \in \delta^-(i)} x_a^k = y_i^k \qquad i \in V, k \in K \tag{2.25}$$

$$\sum_{a \in \delta^+(S)} x_a^k \geq y_j^k \qquad S \subseteq N, j \in S, k \in K \tag{2.26}$$

$$\sum_{i \in V} d_i y_i^k \leq C \qquad k \in K \tag{2.27}$$

$$x_a^k \in \{0, 1\} \qquad a \in A, k \in K \tag{2.28}$$

$$y_i^k \in \{0, 1\} \qquad i \in V, k \in K \tag{2.29}$$

De forma similar a la formulación de dos índices, si definimos la variable continua $u_i^k, i \in N, k \in K$ para representar la carga del vehículo k después de visitar el cliente i, podemos reemplazar las ecuaciones (2.26) y las limitaciones de capacidad de (2.27) por la siguiente familia de restricciones:

$$u_j^k \geq u_i^k + d_j x_{ij}^k - C(1 - x_{ij}^k) \qquad i, j \in N, k \in K \tag{2.30}$$

$$d_i y_i^k \leq u_i^k \leq C y_i^k \qquad i \in N, k \in K \tag{2.31}$$

Formulación con variables de flujo

Para este último modelo, definimos las variables continuas de flujo

$$f_a = \text{carga del vehículo mientras recorre el arco } a \in A.$$

Por tanto, el problema queda de la siguiente manera:

$$\min \sum_{a \in A} c_a x_a$$

sujeto a:

$$\sum_{a \in \delta^+(i)} x_a = \sum_{a \in \delta^-(i)} x_a = 1 \qquad \forall i \in N \qquad (2.32)$$

$$\sum_{a \in \delta^+(0)} x_a = \sum_{a \in \delta^-(0)} x_a = |K| \qquad (2.33)$$

$$\sum_{a \in \delta^+(i)} f_a - \sum_{a \in \delta^-(i)} f_a = d_i \qquad \forall i \in N \qquad (2.34)$$

$$0 \leq f_a \leq C x_a \qquad \forall a \in A \qquad (2.35)$$

$$x_a \in \{0, 1\} \qquad \forall i, j \in V \qquad (2.36)$$

Como podemos observar, la formulación de este problema es muy parecida a la del modelo de dos índices que estudiamos anteriormente. En esta caso, aparece una nueva restricción relacionada con la variable que hemos incorporado: las ecuaciones de flujo de mercancía (2.34). Estas exigen que la diferencia de la carga del vehículo al llegar y salir de un vértice cualquiera sea exactamente la demanda de dicho vértice, d_i.

2.3. Algunas variantes del Problema de Rutas de Vehículos

En esta sección vamos a generalizar el modelo básico del Problema de Rutas de Vehículos para presentar la formulación del VRP periódico y el Problema de Rutas de Vehículos con Inventario.

2.3.1. Problema Periódico de Rutas de Vehículos (PVRP)

El Problema Periódico de Rutas de Vehículos (PVRP, por sus siglas en inglés) es un problema de rutas de vehículos en el que el servicio a los clientes debe prestarse a lo largo de múltiples periodos. Se supone que los clientes deben ser atendidos con una cierta frecuencia y de acuerdo con un calendario de visitas dado, y deben recibir una cantidad fija de producto en cada visita. El objetivo es elegir el calendario de visitas para cada cliente, de entre un conjunto de posibles

calendarios de visitas, y organizar las rutas de los vehículos en cada periodo de manera que el coste de las mismas sea el mínimo.

Como ya sabemos, el VRP fue introducido por Dantzig y Ramser en 1959. La variante del PVRP surgió más tarde (ver [6]), a medida que se hacía cada vez más importante planificar rutas dentro de un horizonte temporal de varios periodos. Se ha investigado ampliamente en la literatura sobre investigación operativa y logística, y se han propuesto varios enfoques para resolver el problema de manera eficiente. A pesar de su complejidad, el PVRP es un problema práctico y relevante que se encuentra en muchos escenarios de la vida real, y su estudio continuo es esencial para mejorar la eficiencia y la sostenibilidad de las operaciones de transporte y logística. Para resolverlo, se deben tener en cuenta las restricciones referentes a los calendarios de visitas de los distintos los clientes, y atender a la capacidad máxima del vehículo. Además, las rutas deben planificarse de manera que los vehículos regresen al punto de partida al final de cada periodo.

Definición del problema

El problema se define sobre un grafo dirigido $G = (V, A)$, con el conjunto de nodos $V = \{0\} \cup N$ donde 0 denota el depósito y $N = \{1, \ldots, n\}$ es el conjunto de clientes, y A es el conjunto de arcos con costo $c_a \geq 0$. Sea $T = \{1, \ldots, R\}$ un conjunto discreto de periodos de tiempo, cada cliente $i \in N$ tiene asociado una demanda d_i sobre T y una frecuencia de visitas F_i que es el número de veces que debe ser visitado a lo largo del horizonte temporal T. Además, cada cliente $i \in N$ tiene asociado un conjunto de posibles calendarios de visitas H_i. Por ejemplo, si el horizonte temporal son los días de lunes a viernes, un cliente que deba ser visitado dos veces ($F_i = 2$) puede tener asociado, por ejemplo, el conjunto de calendarios de visita {{lunes, miércoles}, {martes, jueves}, {lunes, viernes}}, y de ese conjunto hay que elegir uno. El cliente recibe la misma cantidad de producto d_i en cada visita. Para realizar el servicio disponemos de un conjunto K de vehículos con una capacidad homogénea C.

A la hora de resolver el problema hay que tomar tres decisiones: seleccionar un calendario de visitas para cada cliente, asignar los clientes a los vehículos y generar las rutas a realizar en cada periodo del horizonte temporal. Por tanto, en resumen, el PVRP es el problema de encontrar el conjunto de rutas que satisfacen la demanda periódica de los clientes, dentro del calendario seleccionado para cada cliente, con un coste total de rutas mínimo.

2.3.2. Problema de Rutas de Vehículos con Inventario (IRP)

El Problema de Rutas de Vehículos con Inventario (ver [4]), también conocido como IRP, por sus siglas en inglés, es un problema de optimización que involucra la entrega de productos desde un centro de distribución a un conjunto

de clientes, teniendo en cuenta el nivel de inventario en cada cliente. Esto es especialmente relevante en empresas de distribución y logística, donde se busca optimizar la asignación de recursos para maximizar la eficiencia de la operación.

En este problema, se deben determinar las rutas óptimas que minimicen la suma de los costos de transporte y de mantenimiento de inventario, garantizando que cada cliente reciba la cantidad necesaria de productos y que el inventario en cada cliente no exceda un nivel máximo predefinido.

Definición del problema

El IRP se define en el mismo grafo $G = (V, A)$ utilizado para definir el PVRP. El único cambio es que los clientes ya no tienen una demanda diaria constante. En su lugar, se define una demanda d_i^t para cada cliente $i \in N$ en cada periodo de tiempo $t \in T$. Además, a cada i se le asocia un nivel de inventario inicial I_i^0 y una capacidad de almacenamiento s_i. El plan de distribución tiene que ser tal que cada cliente pueda, en cada periodo de tiempo, satisfacer la demanda d_i^t, por lo que debe tener una cantidad suficiente en inventario I_i^t. Además, hay que tener en cuenta que la cantidad entregada en cada visita más el inventario disponible cuando se realiza la visita, no debe superar la capacidad s_i. Se supone que, en cada cliente, el nivel de inventario en el momento $t \in T$ es el nivel de inventario en el momento $t-1$ más la cantidad entregada en el momento t menos la cantidad consumida en t. Hay costes c_a asociados al paso de los vehículos por los arcos, y costes h_i de mantenimiento de inventario en cada nodo.

El objetivo del IRP es determinar la cantidad de producto a entregar a cada cliente y las correspondientes rutas de servicio de manera que se garantice que no hay escasez para ningún cliente en cada periodo de tiempo, y se minimice el coste total de las rutas y de mantenimiento de inventario.

2.3.3. Flexibilidad en los problemas de rutas

Observamos que el PVRP y el IRP comparten la siguiente característica: los clientes pueden ser visitados más de una vez en la solución óptima. En los dos problemas estas visitas múltiples se deben a la necesidad de visitarlo más de una vez a lo largo del horizonte de planificación considerado. Sin embargo, puede ser conveniente tener múltiples visitas a clientes también cuando se considera un único periodo ($|T| = 1$), como muestran los estudios sobre el *Split Delivery Vehicle Routing Problem* (SDVRP). El SDVRP es una generalización del VRP clásico en el que se permite que cada cliente sea visitado por varios vehículos, dividiendo así su demanda. En el caso de querer seguir profundizando sobre este problema, podemos encontrar más información en el artículo de Archetti y Speranza [5] .

Por otro lado, dentro de los problemas de rutas periódicos, encontramos flexibilidad en la definición del PVRP con Elección de Servicio (PVRP-SC, de su siglas en inglés). El PVRP-SC (ver [16]) es una variante del del PVRP en la que la frecuencia de visitas es una variable de decisión. En concreto, se puede visitar a los clientes con mayor frecuencia que la predefinida o, al menos, con la misma frecuencia. Sin embargo, la cantidad de producto a entregar a cada cliente en cada visita está establecida por el calendario de visitas seleccionado, que debe ser elegido entre un conjunto de opciones conocidas de antemano. La frecuencia de visitas elegida conlleva un beneficio, que afecta a la función objetivo. De esta forma, el objetivo del problema es minimizar el coste total de las rutas menos el beneficio total del servicio.

La siguiente tabla muestra las principales diferencias entre el PVRP, el PVRP-SC y el IRP.

Comparaciones entre PVRP, PVRP-SC e IRP

Problema	Periodicidad	Cantidad entregada	Objetivo
PVRP	Calendarios y frecuencias de visita predefinidos.	Misma cantidad en cada visita.	Minimizar el coste de las rutas.
PVRP-SC	Calendarios predefinidos. La frecuencia de las visitas es una variable de decisión.	Depende del la frecuencia y calendario de visitas elegido para cada cliente.	Minimizar el coste total de las rutas menos el beneficio del servicio.
IRP	Sin calendarios predefinidos y con un número de visitas ilimitado.	Se modela como una variable de decisión y depende la política de reposición.	Minimizar los costes de almacenamiento y rutas.

Como puede verse en la tabla, el PVRP-SC relaja la restricción de frecuencia de visitas fija del PVRP y, por tanto, permite un mayor grado de flexibilidad. En cuanto al IRP, el grado de flexibilidad es similar al del PVRP-SC, es decir, no hay frecuencias, calendarios de visitas o cantidades fijas de demanda a entregar. Sin embargo, hay que tener en cuenta las restricciones de inventario.

El Problema Periódico y Flexible de Rutas de Vehículos (FPVRP) es otra variante del PVRP que aumenta aún más el grado de flexibilidad al eliminar los calendarios de visitas predefinidos y la restricción de entregar una cantidad fija en cada visita, aunque cada cliente tiene una capacidad de almacenamiento de mercancía. Dedicaremos los capítulos siguientes de la memoria a este problema.

La relación entre el FPVRP y el PVRP se puede establecer fijando la capacidad de almacenamiento de cada cliente como el ratio entre su demanda total y la frecuencia de visitas F_i dada en el PVRP. De esta forma, en el FPVRP el cliente i tiene que ser visitado al menos F_i veces (la frecuencia dada en el PVRP). Además, hay que decidir la cantidad entregada en cada visita. Esto

aumenta claramente la flexibilidad con respecto a la configuración PVRP y puede producir ahorros de costes.

El FPVRP también se asemeja al IRP en que no se establece una frecuencia fija de visitas y la cantidad entregada es una variable de decisión. Sin embargo, a diferencia del IRP, no se tiene en cuenta el nivel y el costo de inventario. De nuevo, esto confiere al FPVRP una mayor flexibilidad con respecto al IRP, lo que puede suponer un ahorro de costes.

3

El Problema Periódico y Flexible de Rutas de Vehículos

El Problema Periódico y Flexible de Rutas de Vehículos (*Flexible Periodic Vehicle Routing Problem* o FPVRP) es una variante del Problema Periódico de Rutas de Vehículos propuesta y estudiada por Archetti et al. [3]. El objetivo del problema es establecer a mínimo coste una planificación de rutas de vehículos para servir a unos clientes a lo largo de un horizonte temporal de varios periodos. Cada cliente tiene una demanda total que debe ser satisfecha dentro del horizonte temporal, y un límite a la cantidad máxima que se le puede entregar en cada visita. El FPVRP permite una mayor flexibilidad que el PVRP en la planificación de las rutas ya que ni las frecuencias, ni los calendarios de visita, ni las cantidades a entregar a cada cliente en cada visita, están fijados de antemano.

3.1. Descripción del problema

Al igual que en otras versiones del PVRP, en el Problema Periódico y Flexible de Rutas de Vehículos se busca encontrar las rutas óptimas para un conjunto de vehículos que deben atender a un conjunto de clientes en varios periodos. Sin embargo, esta vez contamos con la flexibilidad de poder elegir los días en que visitar a los clientes dentro del horizonte de planificación dado y la cantidad a entregar a los clientes en cada visita. Por tanto, deben tomarse tres tipos de decisiones: qué clientes se visitan en cada periodo del horizonte temporal, qué rutas se deben realizar en cada periodo, y qué cantidad de producto se debe entregar a cada cliente en cada visita.

El problema se define formalmente sobre un grafo completo y dirigo $G = (V, A)$, con un conjunto de nodos $V = \{0\} \cup N$ donde 0 denota el depósito y $N = \{1, \ldots, n\}$ es el conjunto de clientes, y siendo A el conjunto de arcos. Sea $T = \{1, \ldots, R\}$ un conjunto discreto de periodos de tiempo. Cada cliente i ($i = 1, \ldots, n$) tiene asociado una demanda d_i sobre T y una capacidad de almacenamiento s_i. La cantidad entregada a un cliente i en cada visita no puede ser mayor que s_i, y la suma de las cantidades que se le entregan a lo largo de T debe ser igual a d_i. Para realizar el servicio disponemos de un conjunto K

de vehículos con una capacidad homogénea C. Un costo $c_{ij} \geq 0$ está asociado a cada arco $(i, j) \in A$ y se paga cada vez que un vehículo lo atraviesa. El FPVRP se define como el problema de encontrar el conjunto de rutas que minimice los costos totales del desplazamiento satisfaciendo las demandas de los clientes.

3.2. Modelo matemático

En esta sección mostramos una formulación MILP (programación lineal entera mixta, por sus siglas en inglés) para el FPVRP basada en variables de carga. Las formulaciones tradicionales para el VRP utilizan variables de decisión con un índice de vehículo que muestra qué vehículo atraviesa cada arco. Esto da lugar a un elevado número de variables de decisión, sobre todo en problemas en los que las decisiones deben tomarse para los distintos periodos de un horizonte temporal dado, como es el caso del FPVRP. Para mitigar esta dificultad, tal y como hicieron Archetti et al. [4] y Letchford y Salazar-González [21], vamos a usar variables de decisión que identifiquen los arcos utilizados en las soluciones sin explicitar los vehículos que los recorren. Además, usaremos un conjunto adicional de variables continuas de flujo de mercancías que garanticen la correcta definición de las rutas. De esta manera tendremos una formulación compacta para el problema, sin familias de restricciones de tamaño exponencial que requerirían el uso de procedimientos de separación. Este tipo de formulación compacta suele ser eficiente en la práctica, aunque su relajación es normalmente peor que la de los modelos basados sólo en variables de rutas.

Para modelizar el FPVRP vamos pues a usar dos conjuntos de variables de decisión binarias, que identifican los arcos que se recorren y los clientes que se visitan en cada periodo de tiempo respectivamente. Además, se define un conjunto de variables de decisión enteras que indican el número de vehículos que se utilizan en cada periodo de tiempo. Esto se puede calcular fácilmente contando el número de arcos que salen del depósito en cada periodo de tiempo. Finalmente, se utilizan dos conjuntos adicionales de variables continuas. El primero indica la carga de los vehículos cuando atraviesan los arcos, mientras que el segundo muestra la cantidad de producto entregado a cada cliente en cada visita. La definición de las variables es la siguiente:

$$x_a^t := \begin{cases} 1, & \text{si el arco } a \in A \text{ es atravesado por un vehículo en el periodo } t \in T \\ 0, & \text{en otro caso} \end{cases}$$

$$y_i^t := \begin{cases} 1, & \text{si el cliente } i \in N \text{ es visitado en el periodo } t \in T \\ 0, & \text{en otro caso} \end{cases}$$

y_0^t : número de vehículos usados en el periodo $t \in T$.

f_a^t : carga del vehículo que atraviesa el arco $a \in A$ en el periodo $t \in T$.

q_i^t : cantidad entregada al cliente $i \in N$ en el periodo $t \in T$.

Utilizando todo esto, la formulación del FPVRP queda de la siguiente manera:

$$\min \sum_{t \in T} \sum_{a \in A} c_a x_a^t \tag{3.1}$$

sujeto a:

$$q_i^t \leq s_i y_i^t \qquad \forall i \in N, t \in T \tag{3.2}$$

$$\sum_{i \in N} q_i^t \leq C y_0^t \qquad \forall t \in T \tag{3.3}$$

$$\sum_{a \in \delta^+(i)} x_a^t = y_i^t \qquad \forall i \in N, t \in T \tag{3.4}$$

$$\sum_{a \in \delta^+(i)} x_a^t = \sum_{a \in \delta^-(i)} x_a^t \qquad \forall i \in V, t \in T \tag{3.5}$$

$$\sum_{a \in \delta^+(i)} f_a^t - \sum_{a \in \delta^-(i)} f_a^t = \begin{cases} -q_i^t & i \in N \\ \sum_{j \in N} q_j^t, & i = 0 \end{cases} \qquad \forall i \in V, t \in T \tag{3.6}$$

$$f_a^t \leq C x_a^t \qquad \forall a \in A, t \in T \tag{3.7}$$

$$\sum_{a \in \delta^+(0)} x_a^t \leq |K| \qquad \forall t \in T \tag{3.8}$$

$$\sum_{a \in \delta^+(0)} x_a^t = y_0^t \qquad \forall t \in T \tag{3.9}$$

$$\sum_{t \in T} q_i^t = d_i \qquad \forall i \in N \tag{3.10}$$

$$q_i^t \geq 0 \qquad \forall i \in N, t \in T \tag{3.11}$$

$$y_0^t \in \mathbb{Z} \qquad \forall t \in T \tag{3.12}$$

$$y_i^t \in \{0, 1\} \qquad \forall i \in N, t \in T \tag{3.13}$$

$$x_a^t \in \{0, 1\}, f_a^t \geq 0 \qquad \forall a \in A, t \in T \tag{3.14}$$

Al igual que en los modelos que hemos estudiado en los capítulos anteriores, la función objetivo busca minimizar el costo total de las rutas. Ahora, vamos a estudiar lo que aporta cada restricción al modelo:

- (3.2) imponen que la cantidad entregada a cada cliente i en cada periodo no supere su capacidad de almacenamiento s_i.
- (3.3) establecen que la cantidad total de mercancía entregada en el periodo t no supere la capacidad total de los vehículos utilizados en ese periodo.

- (3.4) obligan a que salga exactamente un vehículo del cliente i si ese cliente es visitado en el periodo t, y a que no salga ningún vehículo si el cliente no se visita.
- (3.5) fuerzan a que el número de vehículos que llegan y que salen de un nodo en cada periodo sea el mismo. Es decir, son las restricciones de conservación de flujo.
- (3.6) son las restricciones de conservación de la carga en los nodos. En los clientes imponen que, en cada periodo, la carga de los vehículos que llegan menos la de los que salen del cliente sea igual a la cantidad entregada al cliente. En el depósito imponen que , en cada periodo, la carga transportada por los vehículos que salen menos la de los que entran sea igual a la cantidad total de mercancía a entregar a los clientes.
- (3.7) imponen que, en cada periodo t, la carga de un vehículo que atraviese un arco no supere su capacidad, y vinculan las variables x y f.
- (3.8) garantizan que el número de vehículos utilizados en cada periodo sea como máximo $|K|$.
- (3.9) aseguran que las variables y_0^t tomen el valor adecuado (el del número de vehículo usados ese periodo).
- (3.10) imponen que la cantidad total entregada a cada cliente al final del horizonte temporal sea igual a su demanda total d_i.

Por último, las restricciones (3.11)-(3.14) definen el dominio de las variables.

Esta formulación tiene $|T|(6|N|+2|A|+6)+|N|$ restricciones, con $|T|(|N|+|A|)$ variables binarias, $|T|$ enteras y $|T|(|N|+|A|)$ continuas.

3.3. Restricciones que refuerzan el modelo

En esta sección presentamos algunas restricciones que sirven para reforzar la formulación matemática del FPVRP e intentar mejorar los tiempos computacionales.

- **Restricción I1. Suma de las cargas finales:** Impone que todos los vehículos regresen vacíos al depósito, es decir, con cero carga, en todos los periodos. De hecho, esta desigualdad no es válida, ya que existen soluciones factibles para el FPVRP que no la satisfacen. En cambio, son cortes de optimalidad, ya que existe al menos una solución óptima que la satisface. Por lo tanto, se pueden utilizar para reducir el dominio de las soluciones que se exploran y reducir así el tiempo de cómputo.

$$\sum_{t\in T}\sum_{j\in N} f_{j0}^t = 0 \tag{3.15}$$

- **Restricciones I2. Ruptura de la simetría de las rutas:** Las siguientes desigualdades rompen parcialmente la simetría de las rutas cuando los costes de los arcos son simétricos. Gracias a esto sólo se considerarán las rutas con una orientación determinada (ya que la misma ruta con la orientación opuesta tendrá el mismo coste). Entre las dos orientaciones posibles para una ruta, elegimos la que parte del cliente con el índice más bajo. Para ello, imponemos

$$x_{i0}^t \leq \sum_{r \leq i} x_{0r}^t, \quad \forall i \in N, \forall t \in T \tag{3.16}$$

  Es decir, si el arco $(i, 0)$ entra en el depósito en el periodo de tiempo t, entonces debe haber un arco $(0, r)$ con $r \leq i$ que salga del depósito en este periodo. Las desigualdades que rompen la simetría son también cortes de optimalidad, pero no desigualdades válidas.
- **Restricciones I3. Relación entre las variables x e y:** Esta relación puede imponerse de varias maneras, en nuestra formulación se hace en dos pasos. Por un lado, las restricciones de conservación de carga (3.6) relacionan las variables de carga f con las q, que, a su vez, activan las y. Por otro lado, las restricciones (3.7) relacionan las variables de carga f con las variables de ruta x . No obstante, la relación entre las x e y puede establecerse de una forma más directa. En particular, ningún arco x_{ij} puede ser utilizado en el periodo de tiempo t a menos que los clientes i y j sean visitados en el mismo periodo de tiempo. Por lo tanto, las siguientes desigualdades son válidas:

$$x_{ij}^t \leq \frac{y_i^t + y_j^t}{2}, \quad \forall i, j \in N, \forall t \in T \tag{3.17}$$

  Además, teniendo en cuenta que ningún arco se recorrerá en ambas direcciones en el mismo periodo de tiempo, las desigualdades anteriores pueden reforzarse a:

$$x_{ij}^t + x_{ji}^t \leq \frac{y_i^t + y_j^t}{2}, \quad \forall i \leq j \in N, \forall t \in T \tag{3.18}$$

4

Experimentos computacionales

En este capítulo presentamos los resultados que obtuvimos al implementar el modelo matemático para el FPVRP. También realizamos pruebas para evaluar la efectividad de las restricciones que presentamos en la Sección 3.3, y comparamos las soluciones del PVRP y el FPVRP.

4.1. Material y herramientas de trabajo

Google Colaboratory, o **Google Colab** para abreviar, es un producto de Google Research. Es un servicio alojado de Jupyter Notebook que no requiere configuración y que ofrece acceso gratuito a recursos informáticos con el que se puede escribir y ejecutar código en Python. Está especialmente diseñado para proyectos de ciencia de datos y aprendizaje automático. Proporciona acceso gratuito a recursos de cómputo de Google, como CPU, GPU y TPU, lo que permite realizar cálculos intensivos. Además, permite la colaboración en tiempo real y la integración con Google Drive para la gestión de proyectos.

Google Colab viene con una amplia gama de librerías y frameworks preinstalados, incluyendo la librería **OR-Tools**, una potente librería de optimización desarrollada por Google que proporciona herramientas y algoritmos para resolver problemas de optimización lineal, entera y combinatoria. Con esta librería, los usuarios pueden abordar desafiantes problemas de programación lineal, como los de rutas más cortas, asignación de tareas y muchos otros problemas de optimización.

Inicialmente, teníamos la idea implementar nuestros códigos en Python usando Google Colab y el resolvedor **CBC**. *COIN-OR branch and cut* (CBC) es un solucionador de programación entera mixta de código abierto escrito en C++. Puede utilizarse como ejecutable independiente y como biblioteca invocable. Sin embargo, durante las primeras pruebas, notamos que el tiempo de ejecución era demasiado alto, lo que nos llevó a buscar una alternativa más eficiente. Así pasamos a usar **Gurobi** como resolvedor. Gurobi es un software comercial especializado en optimización matemática que permite modelizar y

resolver diferentes tipos de problemas (LP, MIP, Convexos, No Convexos, etc.). Gurobi tiene un tiempo de ejecución menor que CBC debido a sus algoritmos y técnicas avanzadas de optimización. Además, está diseñado para aprovechar al máximo los recursos de cálculo disponibles, como CPU multinúcleo y GPUs. Esto significa que puede realizar cálculos paralelos y distribuir la carga de trabajo de manera eficiente, lo que resulta en un procesamiento más rápido de los problemas de optimización.

Debido a problemas par ejecutar Gurobi en el entorno del Google Colab, optamos por ejecutar los códigos directamente en un portátil. Como entorno de desarrollo en el portátil usamos **Spyder**. Este es un entorno de desarrollo integrado popular para el análisis de datos y el desarrollo de algoritmos. Proporciona una interfaz intuitiva y amigable que facilita la escritura, ejecución y depuración de código Python.

4.2. Instancias

Las instancias usadas para llevar a cabo nuestros experimentos computacionales son instancias del PVRP provenientes de [24] y que tienen las siguientes características. El número de nodos varía entre 11 o 41. Las coordenadas de los clientes fueron generadas aleatoriamente en $[0, 100] \times [0, 100]$ y el depósito está situado en $(50, 50)$. Las demandas d_i de los cliente varían entre 1 y 15. Los costes c_{ij} se computaron como las distancias euclídeas entre i y j. El número de periodos $|T|$ y el número de vehículos disponibles $|K|$ varían entre 2 y 4. La capacidad de los vehículos se generó aleatoriamente entre 50 y 100. Para cada combinación de número de nodos, número de vehículos y número de periodos, tenemos tres instancias, que denotamos por las letras a, b y c. En las instancias aparecen, asociados a cada cliente i, una frecuencias de visitas F_i y unos posibles calendarios de visitas, que no son necesarios en el FPVRP. Sin embargo, no aparecen las capacidades s_i que deben tener los clientes en el FPVRP, y que nosotros definimos como un número aleatorio entre $\lceil d_i/|T| \rceil$ y d_i.

Para entender mejor esto, vamos a representar gráficamente en la Figura 4.1 estas características para la instancia 11-2-2-a (es decir, la instancia con 11 nodos, $|T| = 2$, $|K| = 2$, de nombre a). En la Figura 4.1, se puede observar que el depósito se encuentra en el centro y se distingue por su color azul. Este depósito está rodeado por 10 clientes que están distribuidos en un área de 100×100. Cada cliente se encuentra acompañado por una dupla, donde la primera componente sirve para identificarlo y la segunda indica su demanda. En esta instancia la capacidad de los vehículos es $C = 51$.

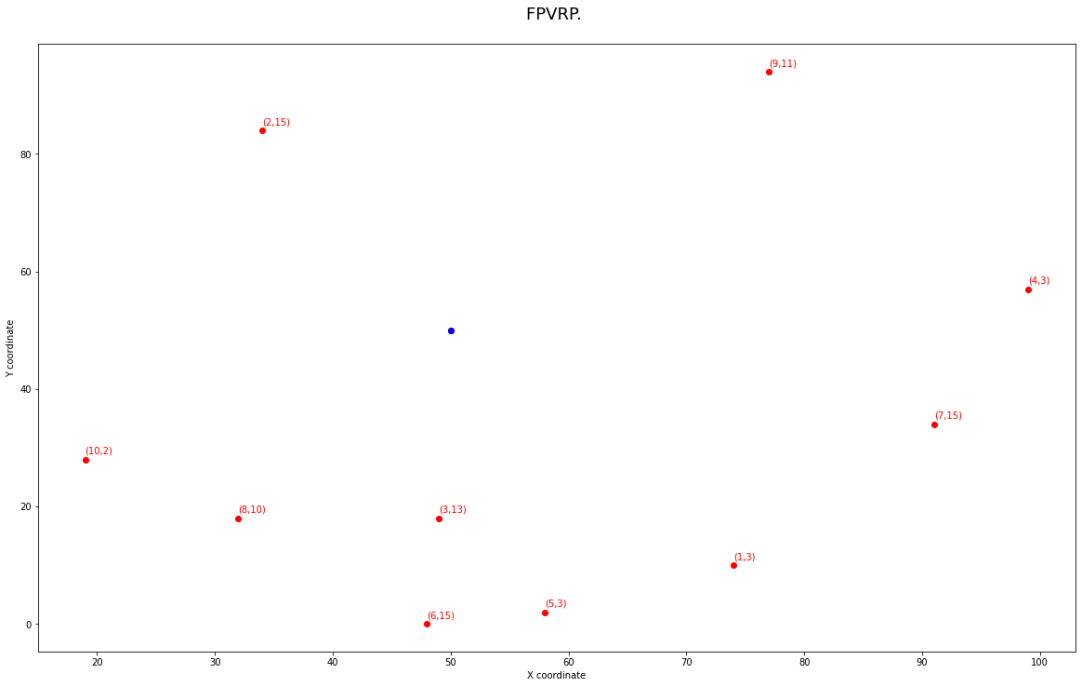


Figura 4.1: Representación instancia 11-2-2-a.

4.3. Ajustes

La Tabla 4.1 muestra los tiempos de computación necesarios para resolver las instancias con 11 nodos, tanto usando el resolvedor CBC como el Gurobi. En el primer caso, el código se ejecutó dentro del Google Colab, mientras que el segundo caso el código se ejecutó en un portátil. Se observa que el Gurobi resuelve todos estos problemas en menos de un segundo, mientras que con CBC se llaga a tardar 153 segundos para resolver la instancia 11-2-3-b. Los tiempos de cómputo no son directamente comparables porque se obtuvieron sobre plataformas diferentes (la web Google Colab en el caso del CBC, y un portátil en el caso del Gurobi), sin embargo, por experiencias de otras personas, queda claro que el Gurobi es mucho más eficiente que el CBC.

A la vista de estos resultados, decidimos usar sólo Gurobi para realizar el resto de los experimentos computacionales.

La siguiente prueba que se realizó consistió en evaluar la conveniencia de reforzar el modelo (3.1)-(3.14) con las restricciones I1 (3.15), I2 (3.16) e I3 (3.18) descritas en la Sección 3.3. Los resultados para las instancias de 21 nodos se pueden ver en la Tabla 4.2. La columna *Base* muestra los tiempos obtenidos resolviendo el modelo (3.1)-(3.14), y las demás columnas muestran los tiempos obtenidos cuando se amplía el modelo base con una o varias de las familias de restricciones adicionales. Para cada instancia se señala en negrita el tiempo

Segundos para conseguir la solución					
$\|V\|$	$\|T\|$	$\|K\|$	**Nom.**	**CBC**	**Gurobi**
11	2	2	a	0.5024	0.0628
11	2	2	b	98.8409	0.1276
11	2	2	c	8.5767	0.0759
11	2	3	a	0.5851	0.0649
11	2	3	b	153.8186	0.0632
11	2	3	c	8.2719	0.1876

Tabla 4.1

mínimo de cómputo. Los resultados muestran que en 10 de los 18 casos los mejores tiempos los da el modelo base ampliado con las restricciones I2 e I3. Esa configuración es la que usaremos a partir de ahora.

4.4. Comparación entre PVRP y FPVRP

El FPVRP puede considerarse una versión del PVRP en la que no están fijados ni la frecuencia ni el calendario de visitas de los clientes. Además, hay que decidir la cantidad entregada a cada cliente en cada visita. Esto aumenta claramente la flexibilidad con respecto a la configuración PVRP y puede producir ahorros de costes. En esta sección vamos a analizar la ventaja potencial del FPVRP en relación con el PVRP. Dado que los dos modelos tienen por objetivo minimizar el coste total de las rutas a lo largo del horizonte temporal, las ventajas potenciales se pueden cuantificar en términos de reducción relativa porcentual del valor de la función objetivo de los modelos comparados. En concreto, utilizaremos el valor

$$\%mej = (Z_{PVRP} - Z_{FPVRP})/Z_{PVRP} \cdot 100,$$

siendo Z_{PVRP} y Z_{FPVRP} los valores óptimos del PVRP y del FPVRP, respetivamete, para medir el porcentaje de ahorro del FPVRP frente al PVRP.

Para poder realizar la comparación entre los dos problemas de manera adecuada, a la hora de resolver el FPVRP definimos la demanda de cada cliente como $d'_i = d_i F_i$ (es decir, la demanda de un cliente en el FPVRP es igual a lo que hay que entregar e ese cliente en el PVRP en cada visita multiplicado por su número de visitas), y la capacidad de almacenamiento como $s'_i = d'_i$ (es decir, no se impone restricción de almacenamiento en el FPVRP). De esta manera nos aseguramos de que el FPVRP y el PVRP se resuelven sobre instancias equivalentes.

Segundos para conseguir la solución con las restricciones										
$\|V\|$	$\|T\|$	$\|K\|$	**Nom.**	**Base**	**I1**	**I2**	**I3**	**I1+I2**	**I1+I3**	**I2+I3**
21	2	2	a	0.7221	0.6830	0.6329	0.6733	0.9240	0.5054	**0.3947**
21	2	2	b	**0.4418**	0.7402	0.6957	0.8617	0.7598	0.6286	0.5001
21	2	2	c	46.5361	40.7294	16.9173	15.5494	**8.1532**	38.7910	12.2003
21	2	3	a	0.6246	0.8474	0.7416	0.8836	0.6411	0.5173	**0.4717**
21	2	3	b	0.6911	0.8332	1.2535	**0.4917**	0.7285	0.6255	0.5582
21	2	3	c	42.1774	31.8405	16.1441	35.4054	13.4739	**13.3159**	18.1787
21	2	4	a	3.1414	4.5094	6.0333	7.1792	4.7040	2.4929	**0.4634**
21	2	4	b	4.4632	4.6665	7.0649	4.4825	7.5836	3.3240	**0.6499**
21	2	4	c	111.6211	33.6303	69.8668	69.2460	49.4534	65.7224	**12.5683**
21	3	2	a	24.0334	3.1205	5.5046	2.4453	2.2486	**2.1419**	2.2493
21	3	2	b	4.3392	8.6564	3.4743	3.7240	4.1094	4.4283	**3.3775**
21	3	2	c	5.5213	6.3529	3.8825	**2.8804**	3.3758	6.8122	4.5773
21	3	3	a	3.1483	2.5305	2.7230	5.8298	**1.9448**	2.4744	2.9092
21	3	3	b	12.2760	6.5965	6.9177	7.7786	5.5409	4.0073	**3.3988**
21	3	3	c	7.3157	3.6782	4.4382	**2.8005**	10.7437	5.0072	4.5521
21	3	4	a	25.1340	19.1496	20.7347	14.6246	18.6907	6.7380	**3.1838**
21	3	4	b	50.1363	52.6326	49.8400	39.5144	58.8208	19.8955	**3.6836**
21	3	4	c	25.0029	24.5398	30.7903	14.2304	53.5062	11.6061	**2.7432**

Tabla 4.2

La Tabla 4.3 muestra el valor óptimo de los problemas PVRP y FVRP para las instancias de 11 y 21 nodos, así como el porcentaje de reducción de costos que se obtiene al *flexibilizar* el PVRP.

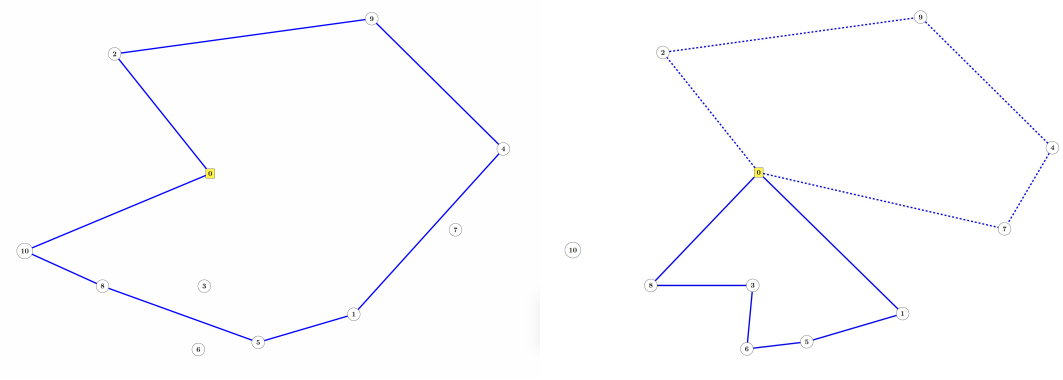
El porcentaje de ahorro siempre es positivo, es decir, existe una mejora en los costos con el FPVRP frente al PVRP. De hecho, al pasar del PVRP al FPVRP se consigue un ahorro que varía entre el 26.87 % y el 46.49 % en el caso de las instancias con 11 nodos, y entre el el 24.62 % y el 39.83 % para las instancias con 21 nodos. Para visualizar esto, vamos a representar gráficamente las soluciones óptimas del PVRP y del FPVRP para la instancia 11-2-2-a. En esta instancia la capacidad de los vehículos es $C = 51$, y las demandas y frecuencias de visita de los clientes en el problema PVRP son $d = \{3, 15, 13, 3, 3, 15, 15, 10, 11, 2\}$ y $F = \{2, 2, 1, 2, 2, 1, 1, 2, 2, 1\}$, y por tanto las demandas a la hora de resolver en FPRVP son $d' = \{6, 30, 13, 6, 6, 15, 15, 20, 22, 2\}$. En el PVRP a los clientes 3 y 6 se les debe visitar una sola vez, en alguno de los dos periodos, al cliente 7 se le puede visitar sólo en el segundo periodo, al cliente 10 sólo en el primero, y a

Comparaciones valor objetivo						
$\|V\|$	$\|T\|$	$\|K\|$	**Nom.**	**FPVRP**	**PVRP**	**%mej**
11	2	2	a	449	614	26.87 %
11	2	2	b	397	547	27.42 %
11	2	2	c	360	574	37.28 %
11	2	3	a	449	614	26.87 %
11	2	3	b	397	547	27.42 %
11	2	3	c	360	574	37.28 %
11	3	2	a	502	827	39.30 %
11	3	2	b	493	923	46.59 %
11	3	2	c	360	666	45.95 %
11	3	3	a	502	827	39.30 %
11	3	3	b	493	923	46.59 %
11	3	3	c	360	666	45.95 %
21	2	2	a	591	818	27.75 %
21	2	2	b	No factible	No factible	
21	2	2	c	529	794	33.38 %
21	2	3	a	591	818	27.75 %
21	2	3	b	591	784	24.62 %
21	2	3	c	529	794	33.38 %
21	3	2	a	653	1073	39.14 %
21	3	2	b	No factible	No factible	
21	3	2	c	571	949	39.83 %
21	3	3	a	653	1073	39.14 %
21	3	3	b	733	1193	38.56 %
21	3	3	c	571	949	39.83 %

Tabla 4.3: Tabla comparación entre FPVRP y PVRP

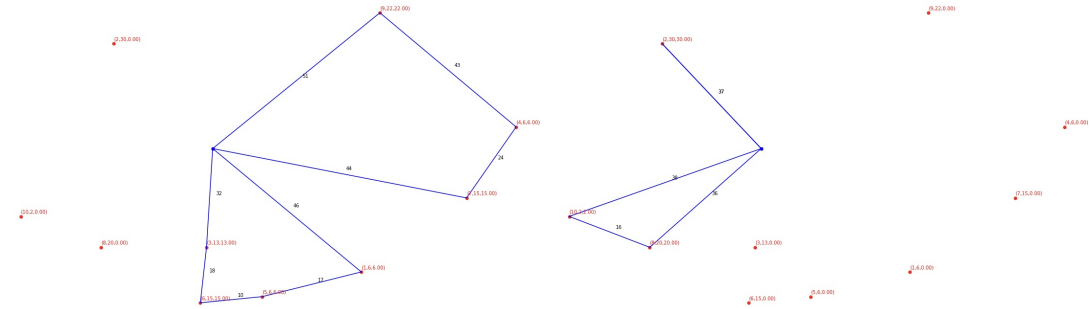
los demás clientes se les debe visitar en ambos periodos. Las soluciones pueden verse en las Figuras 4.2 y 4.3.

Se puede observar que existen diferencias significativas entre las soluciones de los problemas FPVRP y PVRP. El FPVRP presenta ventajas en términos de eficiencia y ahorro debido a su capacidad para elegir el momento y el enfoque óptimo para visitar a los clientes. Un ejemplo concreto que ilustra esta diferencia se encuentra en el cliente $i = 2$. Este cliente tiene, en el PVRP, una demanda de $d_i = 15$ que se debe satisfacer en cada visita, y debe ser visitado en los



(a) $t = 1$ (b) $t = 2$

Figura 4.2: Solución óptima del PVRP para la instancia 11-2-2-a (costo: 614)



(a) $t = 1$ (b) $t = 2$

Figura 4.3: Solución óptima FPVRP para la instancia 11-2-2-a (costo: 449)

dos periodos ($F_i = 2$). En el FPVRP le asociamos una demanda $d'_i = 30$, un $s_i = d'_i$, y relajamos las condiciones sobre las visitas. Como resultado, en la solución óptima del FPVRP, se decide visitar a este cliente sólo en el segundo periodo, en el que se le suministra el total de su demanda. Esto demuestra que el FPVRP proporciona la capacidad de optimizar la decisión de elegir el momento más conveniente para atender a los clientes, lo que a su vez conduce a un ahorro significativo en términos de costos.

4.5. Resultados completos

Las Tablas 4.4 a 4.7 muestran los resultados obtenidos al resolver el FVRPV en el conjunto completo de instancias, que tienen desde 11 hasta 41 nodos. Cada instancia se ejecutó con un límite de 10 minutos de tiempo de cómputo. Cuando se alcanza ese límite se indica con la palabra *superado* en la tabla. Vemos que ese límite sólo se alcanza en algunas instancias de 41 nodos. Podemos ver que el

problema se hace más difícil a medida que aumenta el número de nodos, y para un $|V|$ fijado, normalmente cuando el número de vehículos aumenta.

Tiempo y valor óptimo de cada problema

$\|T\|$	nom.	$k = 2$		$k = 3$		$k = 4$	
		seg.	sol.	seg.	sol.	seg.	sol.
2	a	0.0232	543	0.0225	543		
2	b	0.0877	640	0.1418	640		
2	c	0.0221	557	0.0274	557		
3	a	0.1153	739	0.1171	739		
3	b	0.2231	840	0.2910	840		
3	c	0.3500	740	0.3034	740		

Tabla 4.4: Resultados para $|V| = 11$

Tiempo y valor óptimo de cada problema

$\|T\|$	nom.	$\|K\| = 2$		$\|K\| = 3$		$\|K\| = 4$	
		seg.	sol.	seg.	sol.	seg.	sol.
2	a	0.3947	845	0.4717	845	0.46341	845
2	b	0.5001	754	0.55816	754	0.64985	754
2	c	12.2003	709	18.17871	709	12.56826	709
3	a	2.24929	922	2.90919	922	3.18384	922
3	b	3.37750	891	3.39882	891	3.68362	891
3	c	4.57733	975	4.55213	975	2.74321	975

Tabla 4.5: Resultados para $|V| = 21$

En la Tabla 4.7, que presenta las soluciones para un conjunto de tamaño $|V| = 41$, se puede observar que algunas soluciones están marcadas con un asterisco (*) para indicar que son las mejores soluciones encontradas hasta el momento en relación con el tiempo límite establecido. Sin embargo, es importante destacar que no podemos garantizar que estas soluciones sean óptimas en términos de la función objetivo.

Tiempo y valor óptimo de cada problema							
$\|T\|$	nom.	$\|K\| = 2$		$\|K\| = 3$		$\|K\| = 4$	
		seg.	sol.	seg.	sol.	seg.	sol.
2	a	8.11121	866	12.54313	866	9.93228	866
2	b	22.15322	889	19.27195	889	27.87004	889
2	c	6.03276	841	6.16281	841	3.09311	841
3	a	51.86567	1123	56.35082	1123	71.60467	1123
3	b	267.45780	1127	173.53248	1127	193.65309	1127
3	c	27.11441	974	33.06176	974	12.05045	974
4	a	162.06559	1289	166.20754	1289	316.30279	1289
4	b	40.95535	1269	49.46831	1269	32.31731	1269
4	c	114.55474	1159	232.66822	1159	178.33716	1159

Tabla 4.6: Resultados para $|V| = 31$

Tiempo y valor óptimo de cada problema							
$\|T\|$	nom.	$\|K\| = 2$		$\|K\| = 3$		$\|K\| = 4$	
		seg.	sol.	seg.	sol.	seg.	sol.
2	a	No fact.		48.3481	900	56.0508	900
2	b	33.6592	1045	34.3680	1045	61.5759	1045
2	c	18.2841	1036	68.5621	1036	51.7262	1036
3	a	77.3141	1143	129.8837	1143	151.3503	1143
3	b	superado	1371*	superado	1371*	superado	1371*
3	c	260.8484	1260	295.6339	1260	314.3663	1260
4	a	superado	1295*	superado	1295*	superado	1295*
4	b	362.2173	1376	324.1620	1376	315.4357	1376
4	c	superado	1429*	superado	1436*	superado	1421*

Tabla 4.7: Resultados para $|V| = 41$

4.5.1. Consideraciones adicionales

En las Tablas 4.4 a 4.7 se observa que, dada una instancia con un tamaño y un número de periodos determinados, el valor óptimo del problema es el mismo independientemente del número de vehículos disponibles $|K|$. Esto es así porque, en estas instancias, la capacidad de los vehículos resulta grande en comparación con las demandas de los clientes, y como consecuencia en la solución óptima siempre se utiliza el menor número de vehículos posible, aunque se disponga de más. En todo caso, la situación cambia si imponemos que se usen siempre todos los vehículos disponibles, como se ve en la Tabla 4.8. Por ejemplo, el valor óptimo para la instancia 11-2-2-a es 621 si imponemos que se usen los dos vehículos en cada periodo, y 543 si no lo exigimos. Lo mismo ocurre en las demás instancias.

Comparaciones valor objetivo

$\|V\|$	$\|T\|$	$\|K\|$	**Nom.**	$<= \|K\|$	$== \|K\|$
11	2	2	a	543	621
11	2	2	b	640	711
11	2	2	c	557	593
11	2	3	a	543	716
11	2	3	b	640	787
11	2	3	c	557	651
11	3	2	a	739	839
11	3	2	b	840	930
11	3	2	c	740	818
11	3	3	a	739	991
11	3	3	b	840	1042
11	3	3	c	740	908

Tabla 4.8

Conclusiones

En este trabajo se ha abordado el problema de rutas periódicas de vehículos con flexibilidad en las entregas (FPVRP), en el que un transportista tiene que establecer un plan de distribución para servir a sus clientes durante un horizonte de planificación. Es una versión de uno de los problemas más relevantes y ampliamente estudiados en el campo de la optimización combinatoria: el problema de rutas periódicas de vehículos (PVRP). Además,el FPVRP comparte algunas características comunes con otros VRPs como el problema de rutas de inventario (IRP), en el que se consideran los niveles de inventario en cada periodo de tiempo y, normalmente, se incluye un coste de inventario en la función objetivo.

Durante el desarrollo de este proyecto, hemos adquirido conocimientos significativos en cuanto a las distintas variantes de los problemas de rutas de vehículos y las diferentes formulaciones utilizadas para su resolución. También hemos explorado el uso software como OR-tools o Gurobi para la resolución de los mismos. La familiarización con estas herramientas nos ha permitido ser capaces de formular el modelo matemático y encontrar soluciones óptimas o aproximadas para el problema estudiado.

En términos de experiencia computacional, hemos evaluado tanto la formulación matemática básica del FPVRP como algunas de las restricciones que sirven para reforzar el modelo e intentar mejorar los tiempos computacionales. Nuestro estudio ha demostrado que la combinación de las restricciones I2 y I3 ha mostrado un rendimiento destacado en términos de tiempos computacionales en el contexto del FPVRP. Sin embargo, es importante tener en cuenta que los resultados pueden variar según el problema y los datos específicos, por lo que se recomienda una evaluación cuidadosa y personalizada de las restricciones en cada caso. Además, se ha observado que los métodos propuestos encuentran la solución óptima al problema, pero a medida que aumenta el número de nodos y vehículos, se requiere un mayor esfuerzo computacional para encontrar soluciones óptimas. Esto hace que los enfoques propuestos sean adecuados para problemas de pequeñas dimensiones pero no para aquellos de mayor tamaño.

Por otro lado, los resultados de los experimentos computacionales realizados han demostrado que el FPVRP permite disminuir los costes en comparación con el PVRP debido a la flexibilidad en la asignación de las visitas a los clientes. En el PVRP, se establece la obligación de visitar a cada cliente en ciertos períodos de tiempo, lo que puede conducir a rutas ineficientes y mayores costos operativos. En el FPVRP, los vehículos tienen la libertad de decidir la forma más eficiente de realizar las visitas a los clientes, lo que permite optimizar las rutas y disminuir los costos totales. Al flexibilizar el número y los periodos de visita, además de la cantidad a entregar a los clientes en cada visita, se pueden planificar rutas más eficientes y adaptadas a las necesidades específicas de cada cliente.

A

Apéndice

A.1. Código Python del FPVRP

```
import math
import random
import numpy as np
import time
import gurobipy as gp

# Desactivar la salida de registro de Gurobi
gp.setParam('OutputFlag', 0)

#lectura del .txt del problema
with open("test11-p2-m2-a.txt", "r") as archivo:
    contenido = archivo.read()
    lineas = contenido.split("\n") # dividir el archivo en lineas
    datos = []
    for linea in lineas:
        numeros = list(map(float, linea.split()))
        datos.append(numeros)
        #agregar sublista a la lista principal
datos

#Definimos la dimension del problema
m = datos[0][1] #numero vehiculos
n = datos[0][2] +1 #clientes + deposito
t = datos[0][3] #periodos
EPS = 0.001

info = datos[(int(t))+1: -1]
V = range(len(info))
N = range(1, len(info))
T = range(int(t))

#Guardamos los puntos en el plano de cada localizacion
points = [(info[i][1], info[i][2] ) for i in V]

#Asignamos la misma capacidad para todos los vehiculos
C = datos[1][1]

# Asignamos la demanda de cada cliente
d = [info[i][4] for i in V] #FPVRP
#d = [info[i][4]*info[i][5] for i in V] #Para comparar con el PVRP

# Capacidad de almacenamiento de cada cliente
random.seed(1)
```

```
s = [random.randint(math.ceil(d[i]/t), d[i]) for i in V]
#s = [d[i] for i in V] #Para comparar con el PVRP

# costos
c = np.zeros((int(n), int(n)))
# matriz de costos
for i in V :
  for j in V:
    if j != i:
      dx = points[i][0] - points[j][0]
      dy = points[i][1] - points[j][1]
      # Cost is equal to rounded Euclidean distance
      c[i,j] = math.floor(math.sqrt(dx*dx + dy*dy))

# Funcion para dibujar
import matplotlib.pyplot as plt
#draws a given route as a list of arcs
def dibujaIN(selected):
  #Creamos un dibujo mas grande
  plt.figure(figsize=(20,12))
  plt.xlabel("X coordinate", fontsize='10')
  plt.ylabel("Y coordinate", fontsize='10')
  plt.title('FPVRP. \n', fontsize='18')

  # Dibujamos los nodos con el deposito en azul
  plt.plot([p[0] for p in points], [p[1] for p in points], 'ro')
  plt.plot(points[0][0], points[0][1], 'bo')
  for i in N :
    plt.annotate('(%i,%i)'%(i, d[i]), (points[i][0],points[i][1]+1),fontsize='10',color='red')

  for (i,j) in selected:
    plt.plot([points[i][0],points[j][0]], [points[i][1],points[j][1]], 'b-')
    plt.annotate('%i'%c[i,j], (((points[i][0]+points[j][0])/2)+1,((points[i][1]+points[j][1])/2)+1),
    fontsize='10')

  plt.show()
dibujaIN({})

def dibuja(selected,t):
  # Creamos un dibujo mas grande
  plt.figure(figsize=(20,12))
  plt.xlabel("X coordinate", fontsize='10')
  plt.ylabel("Y coordinate", fontsize='10')
  plt.title('FPVRP. \n', fontsize='18')

# Dibujamos los nodos con el deposito en azul
  plt.plot([p[0] for p in points],[p[1] for p in points], 'ro')
  plt.plot(points[0][0],points[0][1], 'bo')
  for i in N :
    plt.annotate('(%i,%i,%.2f)'%(i,d[i],q[i, t].x), (points[i][0], points[i][1] + 1), fontsize='10',
    color='red')

  for (i,j) in selected:
    plt.plot([points[i][0],points[j][0]], [points[i][1],points[j][1]], 'b-')
    plt.annotate('%i'%c[i,j], (((points[i][0]+points[j][0])/2)+1,((points[i][1]+points[j][1])/2)+1),
    fontsize='10')

  plt.show()

# Modelo
model = gp.Model('FPVRP')

selected = {}
```

```
# Variables
x = {(i,j,t): model.addVar(vtype=gp.GRB.BINARY, name='x[%i,%i,%i]' % (i,j,t)) for i in V for j
in V for t in T if i!=j}
f = {(i,j,t): model.addVar(lb=0.0, ub=gp.GRB.INFINITY, name='f[%i,%i,%i]' % (i,j,t)) for i in V
for j in V for t in T if i!=j}
y = {(i,t): model.addVar(vtype=gp.GRB.BINARY, name='y[%i,%i]' % (i,t)) for i in N for t in T}
q = {(i,t): model.addVar(lb=0.0, ub=gp.GRB.INFINITY, name='q[%i,%i]' % (i,t)) for i in N for t
in T}
y0 = {t: model.addVar(lb=0.0, ub=gp.GRB.INFINITY, name='y0[%i]' % t) for t in T}
# Variable entera

# Funcion objetivo
model.setObjective(gp.quicksum(c[i,j]*x[i,j,t] for i in V for j in V for t in T if i!=j),
gp.GRB.MINIMIZE)

# Restricciones
[model.addConstr(q[i,t]<=s[i]*y[i,t]) for i in N for t in T]
[model.addConstr(gp.quicksum(q[i,t] for i in N)<=C*y0[t]) for t in T]
[model.addConstr(gp.quicksum(x[i,j,t] for j in V if i != j)==y[i,t]) for i in N for t in T]
[model.addConstr(gp.quicksum(x[i,j,t] for j in V if i != j)==gp.quicksum(x[j,i,t] for j in V
if i!=j)) for i in V for t in T]
[model.addConstr(gp.quicksum(f[i,j,t]-f[j,i,t] for j in V if j != i) == -q[i,t]) for i in N
for t in T]
[model.addConstr(gp.quicksum(f[0,j,t]-f[j,0,t] for j in V if j != 0) == gp.quicksum(q[l,t] for l
in N)) for t in T]
[model.addConstr(f[i,j,t] <= C*x[i,j,t]) for i in V for j in V for t in T if j != i]
[model.addConstr(gp.quicksum(x[0,j,t] for j in N)<= m) for t in T]
[model.addConstr(gp.quicksum(x[j,0,t] for j in N)==y0[t]) for t in T]
[model.addConstr(gp.quicksum(q[i,t] for t in T)==d[i]) for i in N]


#Restricciones para mejorar el modelo

#Restriccion 1
[model.addConstr(gp.quicksum(gp.quicksum(f[j,0,t] for j in N) for t in T) == 0)]
#Restriccion2
[model.addConstr(x[i,0,t] <= gp.quicksum(x[0,r,t] for r in N if r<= i))for i in N for t in T]
#Restriccion 3
[model.addConstr((x[i,j,t] + x[j,i,t]) <= ((y[j,t]+y[i,t])/2)) for i in N for j in N
for t in T if i <= j if j!=i ]


# Solucionamos
# Registro de tiempo de inicio
start_time = time.time()
# Establecer limite de tiempo en segundos
model.setParam('TimeLimit', 600) #10 minutos como limite

model.optimize()

# Registro de tiempo de finalizacion
end_time = time.time()

if model.status == gp.GRB.OPTIMAL:
    print('Valor  ptimo  =', model.objVal)
    print("Tiempo:", end_time - start_time, "segundos")
    print( '\n')
    for t in T:
        selected = [(i,j) for i in V for j in V if i != j and x[i,j,t].x > EPS]
        dibuja(selected,t)
elif model.status == gp.GRB.TIME_LIMIT:
    print('Se ha superado el limite de tiempo.')
```

```
    print('Valor =', model.objVal)
    print( '\n')
else:
    print('El problema no tiene solucion optima.')
    print( '\n')
```

Bibliografía

[1] Apllegate, D., Bixby, R., Chvátal, V., and Cook, W. *Finding cuts in the TSP (a preliminary report)*. DIMACS technical report, 95-05, 1995.

[2] Applegate, D., Bixby, R., Chvátal, V., and Cook, W. *The travelling salesman problem: a computational study*. Princeton University Press, 2006.

[3] Archetti, C., Fernández, E., Huerta-Muñoz, D.L., *The flexible periodic vehicle routing problem*. Computers and Operations Research, vol. 85, 58-70, 2017.

[4] Archetti, C., Bianchessi, N., Irnich, S., Speranza, M.G., *Formulations for an inventory routing problem*. International Transactions in Operational Research, vol. 21 (3), 353-374, 2014.

[5] Archetti, C., Speranza, M. *Vehicle routing problems with split deliveries*. International Transactions in Operational Research, vol.39, 3-22, 2012.

[6] Beltrami, E.J., Bodin, L.D. Networks and vehicle routing for municipal waste collection. *Networks* vol. 4, 65-94, 1974.

[7] Bertazzi, L., Speranza, M., *Inventory routing problems: an introduction*. EURO Journal on Transportation and Logistics, vol. 1, 307-326, 2012.

[8] Bertazzi, L., Speranza, M., *Inventory routing problems with multiple customers*. EURO Journal on Transportation and Logistics, vol. 2, 255-275, 2013.

[9] Coelho, L.C., Cordeau, J.-F., Laporte, G. *Thirty years of inventory routing*, Transportation Science. vol. 48, 1-19, 2013.

[10] Christofides, N., Beasley, J.E. *The period routing problem*. Networks, vol. 14, 237–256, 1984.

[11] Clarke, G., Wright, J. *Scheduling of vehicles from a central depot to a number of delivery points*. Operations Research, vol. 12, 568–581, 1964.

[12] Cook, W., Espinoza, D., Goycoolea, M. *Integer programming and combinatorial optimization*. Lecture Notes in Computer Science, vol. 3509, 452-467, 2005.

[13] Crowder, H., Padberg, M. *Solving large scale symmetric travelling salesman problems to optimality*. Management Science, vol. 26, 485-509, 1980.

[14] Dantzig, G., Fulkerson, R., Johnson, S. *Solution of a large-scale traveling salesman problem.* Journal of the Operations Research Society of America, vol. 2, 393–410, 1954

[15] Dantzig, G., Ramser, J. H. *The truck dispatching problem.* Management Science, vol. 6, 80-91, 1959.

[16] Francis, P., Smilowitz, K., Michal, T. *The periodic vehicle routing problem with service choice*. Transportation Sciende, vol. 40, 439–454, 2006.

[17] Grötschel, M. *Polyedrische Charakterisierunge Kombinatorischer Optimierungsprobleme*. Hain, Meisenheim am Glan, 1977.

[18] Grötschel, M., Holland , O. *A cutting plane algorithm for minimum perfect 2-matchings*. Computing, vol. 39, 327-344, 1987.

[19] Gutin, G., Punnen, A. P. (eds.) *The traveling salesman problem and its variations*. Kluwer Academic Publishers, 2002.

[20] Held, M., Karp, R. M. *The travelling salesman problem and minimum spanning trees: part II*. Mathematical Programming, vol. 1, 6-25, 1971.

[21] Letchford, A. N., Salazar-González, J.J. *Stronger multi-commodity flow formulations of the capacitated vehicle routing problem*. European Journal of Operational Research, vol. 244, 730-738, 2015.

[22] Padberg, M., Rinaldi, G. *Optimization of a 532-city symmetric traveling salesman problem by branch and cut*. Operations Research Letters, vol. 6, 1-7, 1987.

[23] Padberg, M., Rinaldi, G. *A branch and cut algorithm for the resolution of large-scale symmetric travelling salesman problems*. SIAM Review, vol. 33, 60-100, 1991.

[24] Rodríguez-Martín, I., Salazar-González, J.J., Yaman, H. *The periodic vehicle routing problem with driver consistency*. European Journal of Operational Research, vol. 273, 575-584, 2019.

[25] Stockdale, M. L. *El problema del viajante: un algoritmo heurístico y una aplicación.* PhD thesis, 2011.

[26] Toth, P., Vigo, D. (eds.) *The vehicle routing problem*. MOS-SIAM Series on Optimization, Philadelphia, 2002.