Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

GRADO EN INGENIERÍA DE ORGANIZACIÓN INDUSTRIAL

ÁREA DE ORGANIZACIÓN DE EMPRESAS

PLANIFICACIÓN DEL REPARTO DE DESPERDICIOS ALIMENTARIOS APTOS PARA LA GANADERÍA

D. MARTÍN HERRERO, Nicolás
TUTOR: Dña. CALZADA INFANTE, Laura

FECHA: JULIO 2024

ÍNDICE

1.- Objetivo y alcance

El objetivo principal de este trabajo es desarrollar un modelo de optimización para la planificación y distribución eficiente de la magaya, desde los llagares hasta distintas ganaderías en Asturias. La magaya es un subproducto de la producción de sidra que se puede aprovechar para alimentar al ganado. No obstante, muchos ganaderos no pueden disponer de ella por el elevado coste de transporte. Este modelo tiene como fin maximizar la eficiencia logística, minimizando los costes asociados al transporte y distribución, y asegurando el aprovechamiento máximo de la magaya como recurso alimenticio para el ganado vacuno. A través de la implementación de un modelo basado en la programación lineal, se pretende proporcionar una solución robusta y sostenible que satisfaga las necesidades de las ganaderías y contribuya a la sostenibilidad del sistema agroalimentario regional.

El alcance de este trabajo abarca la recolección y análisis de datos sobre la producción de magaya en los llagares de sidra y la demanda de magaya en las ganaderías utilizando diferentes fuentes. Se desarrollará un modelo de optimización que incluirá múltiples variables y restricciones, como la capacidad de los vehículos, los costes de transporte, la demanda y oferta de magaya. Además, se realizarán simulaciones con datos reales para validar la efectividad del modelo, identificando posibles mejoras y ajustes necesarios. Se evaluarán los costes de implementación del modelo, incluyendo, por ejemplo, el alquiler de vehículos. Finalmente, se desarrollará una propuesta detallada para la implementación del modelo en la práctica, con recomendaciones para la gestión de la flota de vehículos y la planificación de rutas.

Este trabajo busca no solo optimizar la distribución de magaya, sino también contribuir a la sostenibilidad y eficiencia del sistema agroalimentario en Asturias, promoviendo el aprovechamiento de subproductos y reduciendo el desperdicio alimentario.

2.- Introducción y antecedentes

Actualmente, la población mundial ha alcanzado los 8.000 millones de personas, lo que ha generado una demanda sin precedentes de recursos naturales. Según un informe de 2023 sobre los Objetivos de Desarrollo Sostenible de las Naciones Unidas, cada año se necesitan 1.75 planetas para satisfacer las necesidades de la humanidad. Además, se desperdician aproximadamente 931 millones de toneladas de alimentos anualmente. Esta situación subraya la importancia de la meta 12.3 de los ODS, que busca reducir a la mitad el desperdicio mundial de alimentos per cápita en los niveles de venta minorista y de los consumidores, así como reducir las pérdidas de alimentos en las cadenas de producción y suministro, incluidas las pérdidas posteriores a la cosecha. (*The Sustainable Development Goals Report 2023*, 2023)

En este contexto, la gestión de los desechos alimentarios se ha convertido en una prioridad global. El aprovechamiento de estos desechos, especialmente en la ganadería, no solo contribuye a la sostenibilidad ambiental, sino que también ofrece una oportunidad para reducir costes y mejorar la eficiencia de los sistemas de producción. La reutilización de subproductos como la magaya, los restos de manzana generados durante la producción de sidra puede ser una solución efectiva para abordar estos desafíos globales.

En Asturias, los indicadores sectoriales de producción de carne, pescado y leche han mostrado una tendencia negativa en los últimos años (Hispalink-Asturias, 2024). Por otra parte, a nivel global, en los últimos años se ha observado un incremento en los costes en la cadena alimentaria debido al incremento de los precios de las materias primas, como los cereales por la guerra de Ucrania, generando una presión significativa en los productores locales. Como consecuencia, el precio de la cesta de la compra ha aumentado, afectando tanto a los consumidores finales como a los productores y distribuidores de alimentos.

El aumento de los precios de la cesta de la compra y de las materias primas genera una presión considerable en la cadena alimentaria. Este fenómeno afecta a los consumidores finales y a los productores y distribuidores de alimentos. La inflación de los precios puede

atribuirse a diversos factores, incluidos cambios en la oferta y demanda, variaciones en los costes de producción y fluctuaciones en los mercados internacionales.

Ante este panorama, es crucial que las empresas, especialmente las ganaderas y productoras de leche, encuentren maneras de reducir sus costes de producción. Una estrategia efectiva es el aprovechamiento de los desperdicios alimentarios como fuente de alimento para el ganado. Sin embargo, estos desperdicios, como la magaya, a menudo no se aprovechan debido al desconocimiento de la demanda o al incremento en los costes de transporte.

La planificación eficiente y la optimización de los recursos disponibles son esenciales para que las empresas puedan ahorrar costes y aumentar su competitividad. En el caso específico de las ganaderías, la implementación de modelos de optimización puede ayudar a diseñar rutas de distribución eficientes y a asignar recursos de manera que se minimicen los costes operativos. Al reducir los costes de transporte y mejorar la gestión de los desperdicios alimentarios, las ganaderías pueden asegurar un suministro constante y accesible de alimentos para el ganado, mejorar su sostenibilidad y contribuir a la reducción del desperdicio alimentario.

2.1.- ESTRUCTURA

El presente documento se ha organizado en varios capítulos y secciones con el objetivo de proporcionar una visión clara y comprensible del proyecto de optimización para la distribución de magaya en Asturias. A continuación, se describe brevemente la estructura del documento:

1. **Objetivo y Alcance:** En este capítulo se define el propósito principal del proyecto, los objetivos específicos que se pretenden alcanzar y el alcance del estudio. Se detalla la problemática abordada y se justifica la importancia del proyecto en el contexto de la industria ganadera y la producción de sidra.

2. **Introducción y Antecedentes:** Este capítulo proporciona un marco contextual para el proyecto.

3. **Descripción General del Servicio, Requisitos del Mismo y Criterios para la Evaluación del Cumplimiento de los Mismos:** En este capítulo se presenta una visión general del servicio de planificación desarrollado, junto con los requisitos que debe cumplir y los criterios para evaluar su éxito.

4. **Tecnologías Empleadas:** Este capítulo proporciona un marco teórico y explora las tecnologías y métodos utilizados en el desarrollo del modelo de optimización, además de la definición del mismo.

5. **Resultados**: En este capítulo se presentan y analizan los resultados obtenidos tras la implementación del modelo.

6. **Conclusiones**: En este capítulo se presentan las conclusiones del estudio, destacando los resultados más importantes y su relevancia.

7. **Anexos:** En el Anexo A se presentan las tablas relacionadas con las empresas usadas en este trabajo, mientras que en el Anexo B aparece el código de Python usado para la realización del proyecto.

3.- Descripción general del servicio, requisitos del mismo y criterios para la evaluación del cumplimiento de los mismos

3.1.- DESCRIPCIÓN DEL SERVICIO

El servicio que se detalla en este trabajo se centra en la creación de un modelo para la optimización y planificación de la distribución de magaya desde los distintos llagares hasta las ganaderías en Asturias. La distribución eficiente de la magaya plantea desafíos logísticos significativos debido a la dispersión geográfica de los llagares y las ganaderías, así como a la variabilidad en la oferta y demanda a lo largo del tiempo.

Para abordar estos desafíos, se ha utilizado un modelo de optimización basado en programación lineal y adaptado del Problema de Enrutamiento de Vehículos (VRP). Este modelo tiene en cuenta múltiples tipos de vehículos, cada uno con diferentes capacidades y costes, y optimiza la asignación y las rutas de estos vehículos para minimizar los costes totales del sistema. El servicio no solo mejora la eficiencia logística, sino que también contribuye a la sostenibilidad ambiental al maximizar el aprovechamiento de la magaya y reducir los desperdicios.

3.2.- REQUISITOS DEL SERVICIO

El servicio de planificación propuesto debe cumplir con los siguientes requisitos:

- **Eficiencia en la distribución**: Optimizar las rutas de transporte para minimizar los costes y el tiempo de entrega.
- **Satisfacción del cliente**: Asegurar que las ganaderías reciban la cantidad suficiente de magaya en los tiempos establecidos.

- **Cumplimiento de regulaciones**: Garantizar que todas las operaciones cumplan con las normativas ambientales y de transporte vigentes.

3.3.- CRITERIOS DE EVALUACIÓN

Para evaluar el cumplimiento de los requisitos establecidos para el servicio de planificación y distribución de magaya, sería conveniente considerar los siguientes criterios. Aunque estos criterios no se implementarán en la práctica, su inclusión en el análisis proporciona un marco teórico útil para futuras evaluaciones y mejoras del sistema:

- **Reducción de Costes**: Realizar una comparación detallada de los costes antes y después de la implementación del modelo de optimización. Esto permitiría identificar las áreas donde se han logrado ahorros significativos y ajustar las estrategias en consecuencia. La reducción de costes es un indicador clave de la eficiencia y efectividad del modelo implementado.

- **Satisfacción del Cliente**: Llevar a cabo encuestas periódicas a las ganaderías para evaluar su nivel de satisfacción con la calidad del servicio recibido. Las encuestas podrían incluir preguntas sobre la puntualidad de las entregas, la calidad de la magaya entregada y la flexibilidad del servicio. La satisfacción del cliente es crucial para asegurar la viabilidad a largo plazo del servicio y la mejora continua.

- **Eficiencia en la Distribución**: Analizar las rutas y tiempos de entrega optimizados para evaluar la eficiencia del modelo. Esto podría incluir la medición del tiempo total de entrega, la distancia recorrida y el número de entregas realizadas por vehículo. La eficiencia en la distribución es esencial para minimizar los costes operativos y maximizar la utilización de los recursos.

- **Cumplimiento Normativo**: Verificar el cumplimiento de todas las regulaciones aplicables relacionadas con el transporte de productos agroalimentarios y la gestión de desechos. Esto incluye la revisión de normativas ambientales, de seguridad y de

transporte. Asegurar el cumplimiento normativo es fundamental para evitar sanciones y mantener la integridad del servicio.

4.- Tecnologías empleadas

4.1.- INTRODUCCIÓN A LA OPTIMIZACIÓN COMBINATORIA

La optimización combinatoria es una rama de la matemática y la informática que se enfoca en encontrar la mejor solución dentro de un conjunto finito de posibles soluciones. Estos problemas son comunes en diversas áreas, incluyendo la logística, la producción y la planificación de recursos. Los problemas de optimización combinatoria son generalmente difíciles de resolver debido al gran número de combinaciones posibles.

4.1.1.- Programación lineal

La programación lineal es una técnica matemática utilizada para resolver problemas de optimización en los que la función objetivo y las restricciones son lineales. En términos simples, se trata de maximizar o minimizar una cantidad (como el coste o el tiempo) sujeta a una serie de restricciones (como la capacidad de los vehículos o las demandas de los clientes). Este método se utiliza ampliamente debido a su capacidad para encontrar soluciones óptimas de manera eficiente en problemas donde las relaciones entre las variables son lineales. Los problemas de programación lineal dan una solución óptima exacta y se pueden resolver de varias maneras. Por ejemplo (Robere, 2012; Li et al., 2023):

- ***Simplex***: Un método iterativo que avanza a través de los vértices de un poliedro factible para encontrar la solución óptima. En cada iteración, el método *Simplex* evalúa las posibles soluciones adyacentes y se mueve hacia la mejor opción disponible hasta que no se pueda mejorar más la solución.
- ***Branch and Bound***: Este algoritmo divide el problema en subproblemas más pequeños (ramificación) y calcula límites superiores e inferiores para estos subproblemas (acotación). Si el límite inferior de un subproblema es mayor que el límite superior de otro, ese subproblema se descarta. Este método es particularmente útil para problemas de programación entera y mixta.

- ***Branch and Cut***: Una extensión del método de *Branch and Bound*, que además de ramificar y acotar, agrega cortes (restricciones adicionales) para eliminar fracciones de la región factible que no contienen soluciones enteras viables. Este método es utilizado por *solvers* avanzados como GUROBI, y es eficiente para resolver problemas de gran escala en programación entera y mixta. Los cortes permiten reducir el tamaño del problema más rápidamente, mejorando el tiempo de cómputo necesario para encontrar la solución óptima.

4.2.- ALGORITMOS DE ENRUTAMIENTO

Los algoritmos de enrutamiento son técnicas utilizadas para encontrar las rutas óptimas en redes de transporte. En el contexto de la logística y la distribución, estos algoritmos se utilizan para planificar rutas eficientes que minimicen los costos de transporte y el tiempo de viaje. Entre los problemas más comunes abordados por estos algoritmos y los problemas en los que se basa el modelo estudiado en este trabajo se encuentran el Problema del Viajante (Travelling Salesman Problem, TSP), el Problema de Enrutamiento de Vehículos (Vehicle Routing Problem, VRP) y sus variantes.

4.2.1.- Problema del viajante (TSP)

El Problema del Viajante es uno de los problemas más estudiados en la teoría de la optimización combinatoria. El TSP consiste en encontrar la ruta más corta que permite a un viajante visitar un conjunto de ciudades exactamente una vez y regresar al punto de partida, como se puede observar en la Figura 4.1. A pesar de su simplicidad aparente, el TSP es un problema muy difícil de resolver, ya que no existe un algoritmo eficiente conocido que pueda encontrar la solución óptima para todas las posibles configuraciones de ciudades.

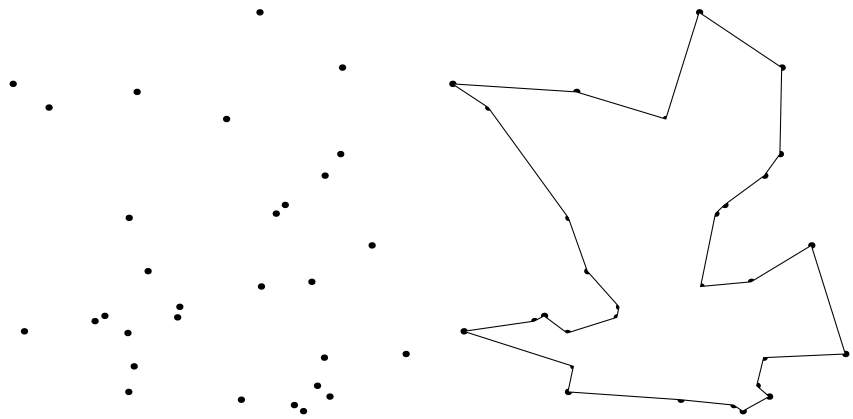

Figura 4.1.- Ejemplo TSP (Fuente: (Weisstein, s. f.))

4.2.2.- Variantes del TSP

El TSP tiene varias variantes que se adaptan a diferentes contextos y restricciones adicionales. A continuación, se describen algunas de las más comunes (Wahyuningsih et al., 2016):

- **TSP asimétrico (ATSP)**: Las distancias entre las ciudades no son necesariamente iguales en ambas direcciones. Es decir, la distancia de la ciudad A a la ciudad B puede ser diferente de la distancia de la ciudad B a la ciudad A. Esta variante es útil en situaciones donde los costos de viaje dependen de la dirección, como en redes de tráfico urbano.

- **TSP con ventanas de tiempo (TSPTW)**: En esta variante, cada ciudad debe ser visitada dentro de un intervalo de tiempo específico. Esta restricción es relevante en la logística de entrega, donde las entregas deben realizarse dentro de ciertos horarios para cumplir con los requisitos del cliente.

- **TSP de múltiples viajantes (mTSP)**: Aquí, múltiples viajantes deben visitar un conjunto de ciudades, cada uno empezando y terminando en su propia base. Esta variante es común en problemas de distribución donde se utilizan múltiples vehículos.

- **TSP con restricciones de capacidad (CTSP)**: El viajante tiene una capacidad limitada para transportar bienes. Es una mezcla del TSP con el problema de la mochila, donde se debe optimizar tanto la ruta como la carga transportada.

4.2.3.- Problema de enrutamiento de vehículos (VRP)

El VRP extiende el TSP al considerar múltiples vehículos que deben realizar entregas a un conjunto de clientes, como se puede ver en la Figura 4.2. El objetivo es minimizar el costo total de las rutas mientras se satisfacen las restricciones de capacidad de los vehículos y las demandas de los clientes. Este problema es aún más complejo que el TSP debido a la necesidad de coordinar múltiples vehículos y optimizar no solo las rutas, sino también la asignación de vehículos a clientes.

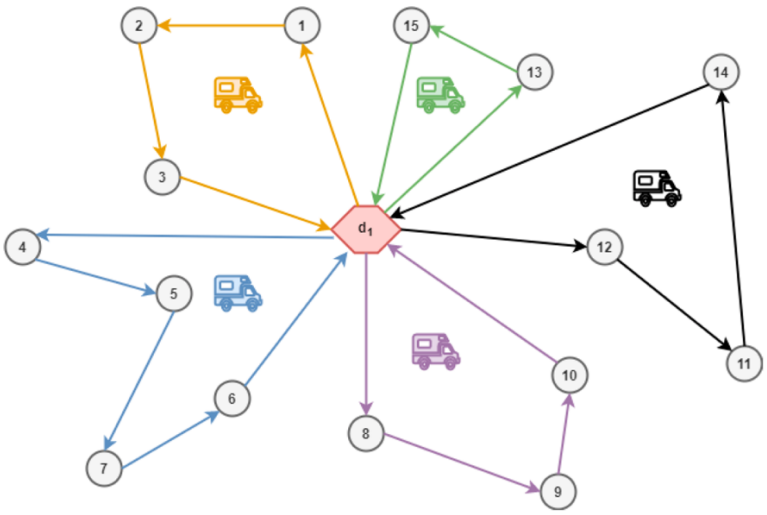

Figura 4.2.- Ejemplo VRP (Fuente: (Çetinyamaç, 2024))

4.2.4.- Variantes del VRP

El VRP también tiene múltiples variantes que incorporan diferentes restricciones y objetivos. Algunas de ellas muy similares a las del TSP. Veamos algún ejemplo (Nanda Kumar & Panneerselvam, 2012):

- **VRP con restricciones de capacidad (CVRP)**: En esta variante, cada vehículo tiene una capacidad limitada para transportar bienes. El objetivo es diseñar rutas que

minimicen el costo total mientras se respetan las restricciones de capacidad de los vehículos.

- **VRP con ventanas de tiempo (VRPTW)**: Similar al TSPTW, esta variante requiere que cada cliente sea atendido dentro de un intervalo de tiempo específico. Esta variante es crucial en la logística de entrega Just-In-Time.

- **VRP de múltiples depósitos (MDVRP)**: En esta variante, los vehículos pueden partir de varios depósitos diferentes. El desafío es coordinar las rutas de los vehículos de manera eficiente considerando múltiples puntos de partida y regreso.

- **VRP con recogidas y entregas (VRPPD)**: Aquí, los vehículos deben recoger y entregar bienes en diferentes ubicaciones. Esta variante es común en problemas de logística inversa y reciclaje.

- **VRP de flota heterogénea (HVRP)**: En esta variante, la flota de vehículos es heterogénea, es decir, los vehículos tienen diferentes capacidades y costos operativos. El objetivo es asignar las rutas de manera que se minimice el costo total considerando las diferencias entre los vehículos.

4.3.- COMPLEJIDAD COMPUTACIONAL

La complejidad computacional es una rama de la informática teórica que se ocupa de clasificar los problemas según la cantidad de recursos necesarios para resolverlos. Estos recursos pueden incluir tiempo, espacio (memoria) y otros factores como energía. La teoría de la complejidad computacional es importante para entender la viabilidad de los algoritmos y la optimización de problemas en contextos prácticos.

4.3.1.- Tipos de complejidad

A continuación, se introducen los diferentes tipos de complejidad computacional para comprender mejor el problema (Papadimitriou & Steiglitz, 1998).

- **P (Problemas Polinomiales)**: Son problemas que pueden ser resueltos en tiempo polinomial, es decir, el tiempo de ejecución del algoritmo crece de manera polinómica con respecto al tamaño de la entrada. Estos problemas son considerados "fáciles" de resolver desde un punto de vista computacional.

- **NP (Problemas No Determinísticos Polinomiales)**: Son problemas para los cuales una solución propuesta puede ser verificada en tiempo polinomial. No se conoce si todos los problemas NP pueden ser resueltos en tiempo polinomial.

- **NP-hard**: Son problemas que, al menos, son tan difíciles como los problemas más difíciles en NP. No se requiere que estos problemas estén en NP (es decir, que sus soluciones puedan ser verificadas en tiempo polinomial).

- **NP-complete**: Son problemas que son tanto NP como NP-hard. Si se encontrara un algoritmo en tiempo polinomial para resolver un problema NP-completo, todos los problemas en NP podrían ser resueltos en tiempo polinomial.

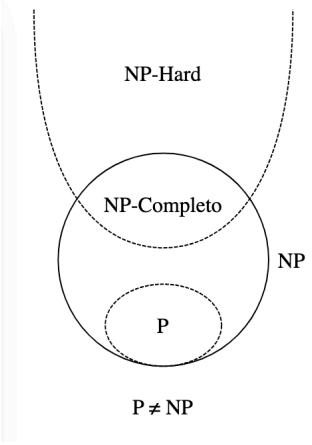

*Figura 4.3.- Clases de complejidades (Fuente: (*Complexity Classes*, s. f.))*

El TSP y el VRP, al ser problemas NP-hard, presentan desafíos significativos en términos de complejidad computacional. La dificultad radica en la explosión combinatoria del número de posibles soluciones a medida que el tamaño del problema aumenta. Esto implica que, para grandes instancias del problema, los métodos exactos pueden ser impracticables debido al tiempo de cómputo excesivo. Por esta razón, se emplean heurísticas y metaheurísticas para encontrar soluciones aproximadas en un tiempo razonable, aunque no garanticen la solución óptima.

4.4.- SOFTWARE Y HERRAMIENTAS

Para el desarrollo del modelo de optimización, se ha utilizado el software Python, junto con la librería *PuLP* para la programación lineal. Python es un lenguaje de programación ampliamente utilizado en el ámbito de la ciencia de datos y la optimización debido a la gran cantidad de librerías disponibles. *PuLP* es una librería de Python que permite la formulación y resolución de problemas de programación lineal. Desde *PuLP* se ha llamado al solver GUROBI, que como antes se ha mencionado, es un solver que usa el método Brunch and Cut para la resolución de modelos.

Además, se han utilizado herramientas de análisis de datos para procesar y gestionar la información obtenida de las bases de datos. Estas herramientas incluyen Excel para la obtención y organización de datos y la librería de Python *pandas* para la manipulación de estos.

El entorno de desarrollo y pruebas se realizó en un ordenador con las siguientes especificaciones:

- **Procesador**: Intel Core i9-9900K a 3.6 GHz
- **Memoria RAM**: 64 GB DDR4

4.5.- DATOS Y BASES DE DATOS

La información utilizada para el desarrollo del modelo proviene de diversas fuentes, incluyendo la base de datos SABI (Sistema de Análisis de Balances Ibéricos). Esta base de datos proporciona información detallada sobre las ubicaciones y los ingresos de los llagares y ganaderías en Asturias. Estos ingresos se han utilizado para estimar la capacidad de producción de magaya y las necesidades de alimentación de las ganaderías.

Además, se han utilizado datos del censo de ganado obtenidos por las encuestas ganaderas (*Ministerio de Agricultura, Pesca y Alimentación*, 2023) para calcular el número de ganado bovino en Asturias y realizar un reparto proporcional entre las ganaderías según los ingresos de explotación de estas. Esta información es crucial para determinar la demanda de magaya y diseñar un sistema de distribución que satisfaga las necesidades de todas las ganaderías de manera eficiente.

Con el fin de realizar los cálculo de los residuos generados aprovechables para la ganadería se ha partido de la cantidad media anual de magaya que se genera en Asturias (*La magaya de la actividad sidrera: fuente de compuestos bioactivos de elevado interés. Ácidos triterpénicos.*, 2018) y se ha repartido este número de manera proporcional según los ingresos de explotación a cada llagar de nuestra base de datos.

4.5.1.- Llagares y Ganaderías

Los datos sobre los llagares y las ganaderías utilizadas en este proyecto han sido obtenidos de la base de datos (SABI, 2024). Esta base de datos proporciona información detallada sobre la ubicación y los ingresos. Para este estudio, se seleccionaron los llagares productores de sidra y las ganaderías que podrían beneficiarse del uso de magaya como alimento para el ganado. La cantidad de empresas usadas en este estudio es de 59, incluyendo 33 llagares y 26 ganaderías. Las ubicaciones totales ascienden a 60, al incluir un depósito para los vehículos que se comentará más adelante.

Para una referencia completa, se incluye una tabla detallada con todos los llagares y ganaderías consideradas en este estudio en el Anexo A del documento (Tabla A.1).

4.5.2.- Ganado Bovino y Magaya

La cantidad de ganado bovino (cabezas de ganado) presente en Asturias a fecha de noviembre de 2023 es de 358.477 (*Resultados de las encuestas de ganado bovino noviembre 2023*, 2023). Esta es la cantidad de ganado con la que se va a trabajar en este trabajo. Se asignará de manera proporcional a los ingresos de cada ganadería un número de ganado.

Según el SERIDA (*La magaya de la actividad sidrera: fuente de compuestos bioactivos de elevado interés. Ácidos triterpénicos.*, 2018), la magaya producida por los llagares en Asturias está entre 9.000 y 12.000 toneladas al año. Se usará un valor medio de 10.500 toneladas para realizar una estimación de la cantidad de magaya que genera cada llagar. De nuevo, de manera proporcional a los ingresos de cada uno de estos.

4.5.3.- Flota de Vehículos

La flota que se contempla para realizar la distribución de la magaya incluye cinco tipos diferentes de vehículos, cada uno con características específicas en términos de capacidad de carga, consumo de combustible y costes asociados. A continuación, se describen los detalles de cada tipo de vehículo, incluyendo estimaciones de consumo, capacidad y precios del alquiler.

Tipo 1: Camión Grande

- Capacidad de Carga: 20 toneladas
- Consumo de Combustible: 30 litros/100 km
- Precio de Alquiler: 90 €/periodo

Tipo 2: Camión Mediano

- Capacidad de Carga: 10 toneladas
- Consumo de Combustible: 25 litros/100 km
- Precio de Alquiler: 50 €/periodo

Tipo 3: Furgoneta Grande

- Capacidad de Carga: 3 toneladas
- Consumo de Combustible: 15 litros/100 km
- Precie de Alquiler: 20 €/periodo

Tipo 4: Furgoneta Mediana

- Capacidad de Carga: 1.5 toneladas
- Consumo de Combustible: 10 litros/100 km
- Precio de Alquiler: 10 €/periodo

Tipo 5: Furgoneta Pequeña

- Capacidad de Carga: 0.75 toneladas
- Consumo de Combustible: 7 litros/100 km
- Precio de Alquiler: 6 €/periodo

4.6.- MODELO

El objetivo principal del modelo desarrollado en este proyecto es planificar de manera eficiente la distribución de magaya desde los llagares hasta las ganaderías en Asturias. Para ello, se ha diseñado un VRP que tiene en cuenta múltiples factores relevantes, como la cantidad de magaya disponible en los llagares, la demanda de las ganaderías, las capacidades de los vehículos, y las rutas de transporte.

El modelo considera un depósito central desde el cual los vehículos salen al inicio de cada periodo y al cual regresan al final de este. Este enfoque garantiza que los vehículos comiencen y terminen sus rutas en un punto centralizado, facilitando la gestión y el control de la flota. Este depósito no tiene almacén de producto, por lo que los vehículos tienen que entrar y salir de él vacíos.

El modelo matemático desarrollado utiliza programación lineal para optimizar las rutas de los vehículos, asegurando que la distribución de magaya sea lo más eficiente y económica posible.

El modelo una variación del VRP, adaptado para la distribución de magaya desde los llagares hasta las ganaderías. Este modelo considera múltiples tipos de vehículos, cada uno con capacidades y costes de ruta específicos, y busca optimizar la asignación y las rutas de estos vehículos para minimizar los costes totales del sistema.

Además, se ha basado en el punto 7.5, Problema de dimensionamiento de flota, del libro “Formulación de Modelos de Programación Matemática” (Díaz et al., 2021). Este libro proporciona un marco teórico y práctico fundamental para la formulación de modelos de optimización, lo cual ha sido adaptado y aplicado a la problemática específica de la distribución de magaya en Asturias.

Como resultado, el modelo es una mezcla de varias variantes anteriormente mencionadas, como el **CVRP**, ya que cada vehículo tiene una capacidad limitada para transportar la magaya; el **VRPPD**, porque tenemos nodos con oferta (llagares) y nodos con demanda (ganaderías) y el **HVRP**, al tener cada tipo de vehículo distintos costes de adquisición, mantenimiento, retirada y costes de ruta. Modelamos el problema matemáticamente en el siguiente apartado.

4.6.1.- Modelo matemático

CONJUNTOS

- K: Conjunto de vehículos.
- N: Conjunto de nodos.
- A: Conjunto de arcos, $A = \{(i,j): \text{i,j} \in N, i \neq j\}$
- δ_i^+: Subconjunto de arcos con origen en i, $\delta_i^+ = \{(i,j) \in A, i \neq j\}$
- δ_i^-: Subconjunto de arcos con destino en i, $\delta_i^- = \{(j,i) \in A, i \neq j\}$

PARÁMETROS

- T: Número de periodos en el horizonte de planificación (para esta simulación, un periodo es un día).
- a_k: Coste de alquiler diario (por periodo) de un vehículo tipo *k* (€/vehículo$_K$).
- c_{ijk}: Coste del viaje del arco *(i, j)* en el vehículo *k* (€/vehículo$_K$).
- $d_{(i,j)}$: Tiempo de viaje en el arco *(i, j)* (horas/vehículo$_K$).
- Q_{nt}: Demanda de producto del nodo *n* en el periodo *t*.
- S_{nt}: Oferta de producto del nodo *n* en el periodo *t*.
- C_k: Capacidad del vehículo tipo *k*.
- γ_n: Coste de penalización por no completar la entrega de cada kg de demanda en el nodo *n* (ficticio).
- Γ_k: Número total de horas que puede operar un vehículo tipo *k* en un periodo cualquiera.
- h_n: Coste por cada unidad (kg) del inventario del nodo *n* (ficticio).

VARIABLES DE DECISIÓN

- q_{nt}: Demanda de producto satisfecha del nodo *n* en el periodo *t*.
- Q_{nt}^{inc}: Demanda de producto insatisfecha del nodo *n* en el periodo *t*.
- f_{nt}: Oferta de producto recogida del nodo *n* en el periodo *t*.

- F_{nt}^{inc}: Oferta de producto no recogida del nodo *n* en el periodo *t*.
- x_{ijkt}: Número de vehículos del tipo *k* que transitan el arco *(i, j)* en el periodo *t*.
- I_{nt}: Inventario disponible dentro del nodo *n* en el periodo *t*.
- b_{ijkt}: Cantidad de producto trasportada en el arco *(i, j)* por vehículos tipo *k* en el periodo *t*.
- z_{kt}: El vehículo tipo *k* se usa en el periodo *t*.
- U_{ikt}: Variable entera necesaria para evitar la creación de subciclos no deseados y establecer el orden en el que se visita el nodo *n* por el vehículo *k* en el periodo *t*.
- Z_{ijkt}: Variable binaria que formará un circuito cerrado de al menos un vehículo para evitar bucles.
- P_{ikt}: Variable entera necesaria para las restricciones de prevención de subciclos no deseados. Si es positiva y distinta de 0 indica que el vehículo *k* en el periodo *t* pasa por el nodo *i*.

FUNCIÓN OBJETIVO

La función objetivo recoge varios sumatorios que intentará minimizar. El primero de ellos se refiere a la cantidad de vehículos a contratar, el segundo a los costes de transitar rutas y el tercero tratará de minimizar el inventario de cada nodo.

$$min \sum_{k=1}^{K} \sum_{t=1}^{T} (a_k \cdot z_{kt}) + \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} (c_{ijk} \cdot x_{ijkt}) + \sum_{n=1}^{N} \sum_{t=1}^{T} (\gamma_n \cdot q_{nt}^{inc} + h_n \cdot I_{nt})$$

RESTRICCIONES

1. La demanda en cada nodo y periodo puede ser satisfecha o incumplida.

$$q_{nt} + Q_{nt}^{inc} = Q_{nt} \qquad \forall n \in N, t \in T \tag{4.1}$$

2. La oferta en cada nodo y periodo puede ser satisfecha o incumplida.

$$f_{nt} + F_{nt}^{inc} = S_{nt} \qquad \forall n \in N, t \in T \tag{4.2}$$

3. Los flujos de vehículos deben equilibrarse en cada nodo dentro de cada periodo.

$$\sum_{j \in \delta_i^+} x_{ijkt} - \sum_{j \in \delta_i^-} x_{jikt} = 0 \qquad \forall i, j \in N, k \in K, t \in T \tag{4.3}$$

4. Se debe asegurar que las horas-vehículo de uso para cada tipo de vehículo *k* acumuladas durante el periodo de tiempo sea menor o igual al número total de horas-vehículo útiles que puede proporcionar la flora en el periodo de tiempo *t*.

$$\sum_{(i,j) \in A} d_{(i,j)} \cdot x_{ijkt} \leq \Gamma_k \cdot z_{kt} \qquad \forall k \in K, t \in T \tag{4.4}$$

5. En el periodo ficticio inicial y para cada tipo de vehículo, no hay vehículos en el sistema.

$$z_{k1} = 0, \forall k \in K \tag{4.5}$$

6. Todos los vehículos tienen que salir desde el nodo inicial.

$$\sum_{j \in \delta_1^+} x_{1jkt} = z_{kt} \qquad \forall t \in T, k \in K \tag{4.6}$$

7. Todos los vehículos tienen que llegar al nodo inicial de vuelta.

$$\sum_{j \in \delta_1^-} x_{j1kt} = z_{kt} \qquad \forall t \in T, k \in K \tag{4.7}$$

8. La cantidad transportada que sale del nodo inicial y llega a él es 0 para cada arco y para cada tipo de vehículo.

$$b_{1jkt} = b_{j1kt} = 0 \qquad \forall j \in N, k \in K, t \in T \tag{4.8}$$

9. La cantidad de producto transportada en el arco *(i, j)* con el vehículo tipo *k* en el periodo *t* no puede ser mayor a la capacidad de dicho vehículo.

$$b_{ijkt} \leq c_k \cdot x_{ijkt} \qquad \forall (i,j) \in A, k \in K, t \in T \tag{4.9}$$

10. En el periodo ficticio inicial, para cada nodo, el inventario está vacío.

$$I_{n1} = 0 \qquad \forall n \in N \tag{4.10}$$

11. Tiene que haber equilibrio de flujo en los inventarios. El inventario en el periodo *t* el igual al que había en el periodo anterior más lo que se recoge en ese nodo, y lo que llega de otros nodos. menos lo que demanda ese nodo y lo que se envía a otros nodos:

$$I_{it} = I_{i(t-1)} + f_{it} + \sum_{k=1}^{K} \sum_{j \in \delta_i^-} b_{ijkt} - q_{it} - \sum_{k=1}^{K} \sum_{j \in \delta_i^+} b_{ijkt} \qquad \forall t \in T, i \in N \tag{4.11}$$

12. Para evitar subciclos no deseados, usamos las restricciones anti bucles de Miller-Tucker-Zemlin (MTZ). Estas restricciones hacen que solo haya flujo en un único sentido para cada arco.

$$U_{1kt} = 0 \qquad \forall t \in T, k \in K \tag{4.12.1}$$

$$1 \leq U_{ikt} \leq N \qquad \forall i \in N, k \in K, t \in T \tag{4.12.2}$$

$$U_{jkt} \geq U_{ikt} + M \cdot (z_{ijkt} - 1) \qquad \forall k \in K, t \in T, i, j \in N, (i,j) \in A, j \neq 1 \tag{4.12.3}$$

13. Restricciones necesarias para permitir la circulación de más de un vehículo por cada tipo y la creación de bucles evitando los subciclos no deseados:

    a. La variable P_{ikt} será distinta de 0 y positiva si forma parte del camino generado por la restricción 12:

$$P_{ikt} = \sum_{j \in \delta_i^+} z_{ijkt} + \sum_{j \in \delta_i^-} z_{jikt} \qquad \forall i \in N, k \in K, t \in T \tag{4.13.1}$$

    b. Se permite el flujo de vehículos siempre que salgan desde un nodo *i* con una $P_{ikt} > 0$:

$$x_{ijkt} \leq M \cdot P_{ikt} \qquad \forall (i,j) \in A, k \in K, t \in T \tag{4.13.2}$$

    c. La ruta ficticia generada en la restricción 12 debe ser recorrida al menos una vez para evitar la formación de subciclos no deseados, pudiendo ser recorrida por varios vehículos:

$$x_{ijkt} \geq z_{ijkt} \qquad \forall (i,j) \in A, k \in K, t \in T \tag{4.13.3}$$

14. Se definen las variables:

$$q_{nt}, Q_{nt}^{inc}, f_{nt}, F_{nt}^{inc}, I_{nt} \in \mathbb{N} \qquad \forall n \in N, t \in T \tag{4.14.1}$$

$$P_{ikt}, U_{ikt,} \in \mathbb{N} \qquad i \in N, k \in K, t \in T \tag{4.14.2}$$

$$z_{kt} \in \{0,1\} \qquad \forall k \in K, t \in T \tag{4.14.3}$$

$$b_{ijkt} \in \mathbb{N} \qquad \forall (i,j) \in A, k \in K, t \in T \tag{4.14.4}$$

$$x_{ijkt} \in \mathbb{N} \qquad \forall (i,j) \in A, k \in K, t \in T \tag{4.14.5}$$

4.6.2.- Datos relevantes del modelo

Vamos a abordar algún detalle importante para la mejor comprensión del modelo.

- **Variables de oferta y demanda:** El modelo incluye variables para la demanda satisfecha y la demanda (u oferta) insatisfecha, usadas en las restricciones 4.1 y 4.2. Esta distinción es necesaria porque en la práctica puede no ser posible satisfacer toda la demanda debido a limitaciones de capacidad, disponibilidad de vehículos o restricciones temporales. La inclusión de una variable de demanda insatisfecha permite al modelo penalizar estas situaciones mediante un coste asociado a la demanda no cumplida, incentivando así la minimización de esta variable en la función objetivo.

- **Equilibrio de flujos de vehículos:** El equilibrio de flujos de vehículos es una restricción fundamental en el modelo (ver restricción 4.3), representada por la necesidad de que los flujos que entran y salen de un nodo se equilibren. Esto asegura que los vehículos que entran a un nodo también salen de él, lo que es esencial para mantener la coherencia en la planificación de rutas y evitar situaciones inviables donde los vehículos “desaparecen” o “aparecen” sin justificación.

- **Capacidad de vehículos:** Las restricciones de capacidad de los vehículos son críticas para reflejar las limitaciones físicas de transporte (restricción 4.8). Cada vehículo tiene una capacidad máxima de producto que puede transportar, y el modelo debe asegurarse de que esta capacidad no se exceda en ninguna ruta. Esta restricción garantiza que las soluciones propuestas sean factibles en términos operativos.

- **Horas-Vehículo disponibles:** La restricción 4.4 de horas-vehículo disponibles asegura que los vehículos no operen más allá de su capacidad temporal. Esto es importante para reflejar las limitaciones reales de operación, como las horas de trabajo permitidas y la disponibilidad de los vehículos. Al imponer esta restricción,

el modelo garantiza que las rutas planificadas sean viables dentro de los límites de tiempo operativos.

- **Inventario, equilibrio de Inventarios y coste de inventarios:** El modelo incluye variables de inventario para cada nodo y periodo, así como restricciones para mantener el equilibrio de inventarios. Estos inventarios están disponibles en todos los nodos. Si en un periodo no se recoge toda la magaya de un nodo con oferta, esa magaya se perdería y se desecharía, a no ser que el modelo crea necesario reservar parte de esa magaya para un periodo posterior, por lo que se le podría solicitar al llagar no desechar toda la magaya y reservar parte de ella en un "inventario" para recogerla en un periodo posterior. De la misma forma, se puede recoger el producto de un nodo con oferta y entregar en una ganadería más magaya de la solicitada. De esta manera, la ganadería guardaría esa magaya en su inventario, y la consumiría en el siguiente periodo. Es necesario incluir un coste relacionado con este inventario para incluirlo en la función objetivo y así que el modelo intente minimizar en la medida de lo posible la cantidad de producto que hay en los inventarios.

- **Restricciones anti subciclos no deseados:** Para evitar la creación de subciclos imposibles a nivel físico, se utilizan restricciones anti subciclos de Miller-Tucker-Zemlin (MTZ) (restricciones 4.12) en combinación con las restricciones 4.13. Las restricciones MTZ garantizan que los vehículos no formen bucles en su recorrido, lo que es crucial para asegurar que las rutas planificadas sean eficientes y no innecesariamente repetitivas. Además, se evitan subciclos infactibles desconectados de la ruta que puede seguir un vehículo en el mundo real. Con estas restricciones se asegura que, que los nodos se visitan en un determinado orden (U). Esto impide que un mismo vehículo pueda visitar el mismo nodo más de una vez. Esta restricción combinada con la de conservación de flujo consigue evitar la formación de estos bucles. Sin embargo, esta restricción tiene sus limitaciones. Al no poder pasar por el mismo nodo más de una vez, se restringe de manera agresiva la libertad de movimiento de un vehículo, obligando a necesitar más vehículos de los necesarios. Además, para el correcto funcionamiento de esta restricción se tiene

que usar una variable binaria, lo que forzaría al modelo a solo poder elegir un único vehículo por cada tipo. Gracias al grupo de restricciones 4.13, se evita en gran parte estos problemas. La variable z_{ijkt} usada para las restricciones MTZ es ahora una variable auxiliar. Se usa la ruta generada por esta restricción para asignarle un valor positivo y distinto de 0 a la variable P_{ikt}, siendo positiva en los nodos por los que haya pasado la ruta. Se permite el flujo libre de vehículos entre estos nodos, imponiendo únicamente que la ruta generada por la restricción 12 sea recorrida por al menos una vez durante ese periodo. Esta es una manera más eficiente y que no restringe prácticamente la libertad del modelo ni aleja la solución óptima.

- **Restricciones para la salida y llegada al depósito:** En este modelo, como veremos más adelante, consideramos un depósito de vehículos. Este depósito no puede tener producto, es únicamente un depósito para guardar los vehículos contratados. Por lo tanto, es necesario incluir las restricciones 5 a 8 (4.5 – 4.8). Las restricciones 4.6 y 4.7 aseguran que todos los vehículos contratados salgan y lleguen al nodo inicial (depósito) y la restricción 4.8 asegura que la cantidad de producto que pueden transportar en la salida o llegada a este depósito tiene que ser 0.

4.7.- UBICACIÓN DEL DEPÓSITO DE VEHÍCULOS

Usando los datos de ubicación de llagares y ganaderías y con la ayuda de Python se ha calculado la ubicación del depósito para los vehículos. El programa realizado se basa en calcular un centro de gravedad entre todos los puntos. Este programa le da el doble de peso a los llagares, porque los vehículos tendrán que pasar primero a recoger la magaya antes de entregarla a las ganaderías. Además, los pesos de cada llagar y de cada ganadería son también proporcionales a los ingresos.

4.8.- ELECCIÓN DE OFERTA Y DEMANDA

La magaya es prácticamente fibra en su totalidad (Bedriñana et al., 2021), por lo que puede usarse como forraje para la alimentación del ganado. El ganado vacuno necesita cada día

entre un 7 y un 10 % de forraje (fibra) (Arronis, 2006). Por lo tanto, suponiendo un forraje exclusivo de magaya, la demanda anual sería mucho mayor a la oferta disponible.

Las ganaderías que hay en la región son muchas más a las que se ha podido acceder en la base de datos usada para este trabajo. En realidad, el número de ganado total de las empresas vistas va a ser mucho menor al número total de ganado de la comunidad. Es por eso por lo que, para la simulación, se va a suponer una oferta y demanda equilibrada. Se repartirá la cantidad de magaya disponible entre las ganaderías, respetando la proporcionalidad de ingresos de cada una de ellas, y sobre ese valor se trabajará la demanda.

Tanto la oferta como la demanda variarán de manera aleatoria hasta en $\pm$ 15 %. Este es un valor arbitrario asignado para representar una situación más realista, en la que pudiera haber mayor o menor producción de magaya en distintos periodos, además de diferente demanda por diversas razones. La oferta y demanda media de cada nodo se puede consultar en el Anexo A (Tabla A.2)

4.9.- ELECCIÓN DEL RESTO DE PARÁMETROS

A continuación, se detallan los valores del resto de parámetros del modelo y la justificación de su elección basada en datos actuales y prácticas comunes:

- Se ha supuesto que la velocidad promedio de los viajes en Asturias es de aproximadamente 67 km/h. Esta estimación se basa en la consideración de las características geográficas y la infraestructura vial de la región. Asturias cuenta con una combinación de carreteras rurales y autopistas, lo que permite mantener una velocidad relativamente constante en la mayoría de las rutas.

- El coste del diésel se ha estimado en 1.5 €/litro, basado en los precios promedio de los últimos meses. (Confederación Española de Transporte de Mercancías, 2024)

- Se ha asumido que cada vehículo operará durante 16 horas al día, distribuidas en dos turnos de 8 horas con dos conductores. Esta configuración permite maximizar la utilización de los vehículos y mejorar la eficiencia operativa. La práctica de utilizar dos conductores por vehículo es común en la industria del transporte para asegurar que los vehículos puedan operar durante largos periodos sin infringir las regulaciones laborales sobre el tiempo de conducción y descanso obligatorio.

5.- Resultados

5.1.- RESULTADOS DEL MODELO

El objetivo de este capítulo es presentar y analizar los resultados obtenidos tras la implementación del modelo de optimización. Los resultados se derivan de la simulación del modelo utilizando los datos reales mencionados en el apartado 4.5.

Una vez obtenidos los resultados mediante el solver GUROBI, se aprecia que la simulación del modelo determinó la necesidad de usar únicamente dos tipos de vehículos de los cinco disponibles, K1 y K2 (correspondientes a los camiones más grandes), para cada periodo de distribución, como se puede ver en la Tabla 5.1. Con los datos usados en el modelo, coincide que para cada periodo se requiere únicamente un vehículo de cada tipo para cumplir con las demandas de las ganaderías y la capacidad de producción de los llagares.

Periodo	K1	K2	K3	K4	K5
1	1	1	0	0	0
2	1	1	0	0	0

Tabla 5.1.- Número de vehículos utilizados

Para ver de una manera gráfica los resultados, se muestran a continuación sobre un mapa los recorridos de cada vehículo a lo largo del primer periodo:

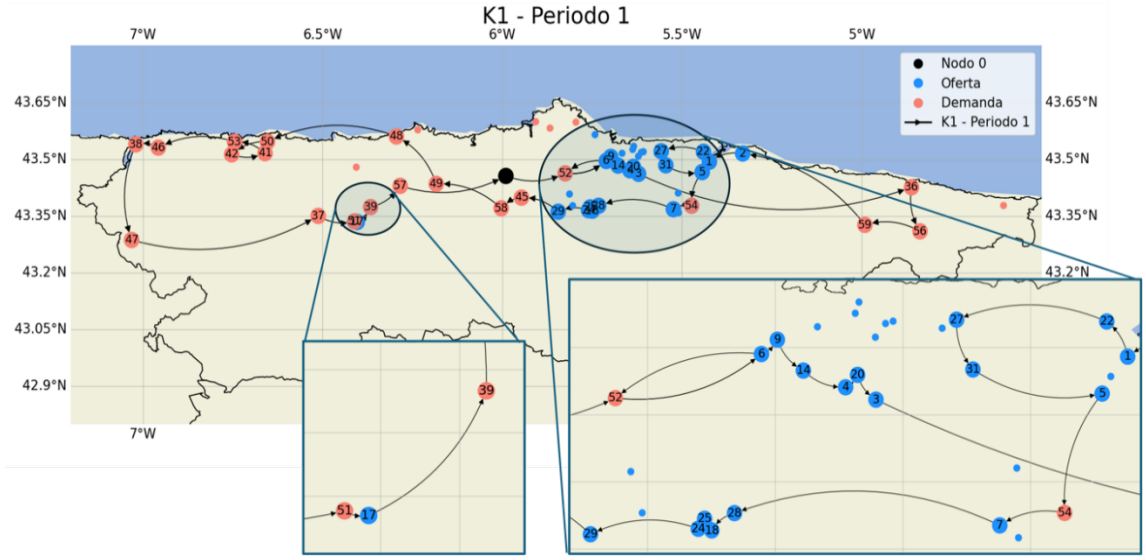

Figura 5.1.- Recorrido del vehículo K1 durante el periodo 1.

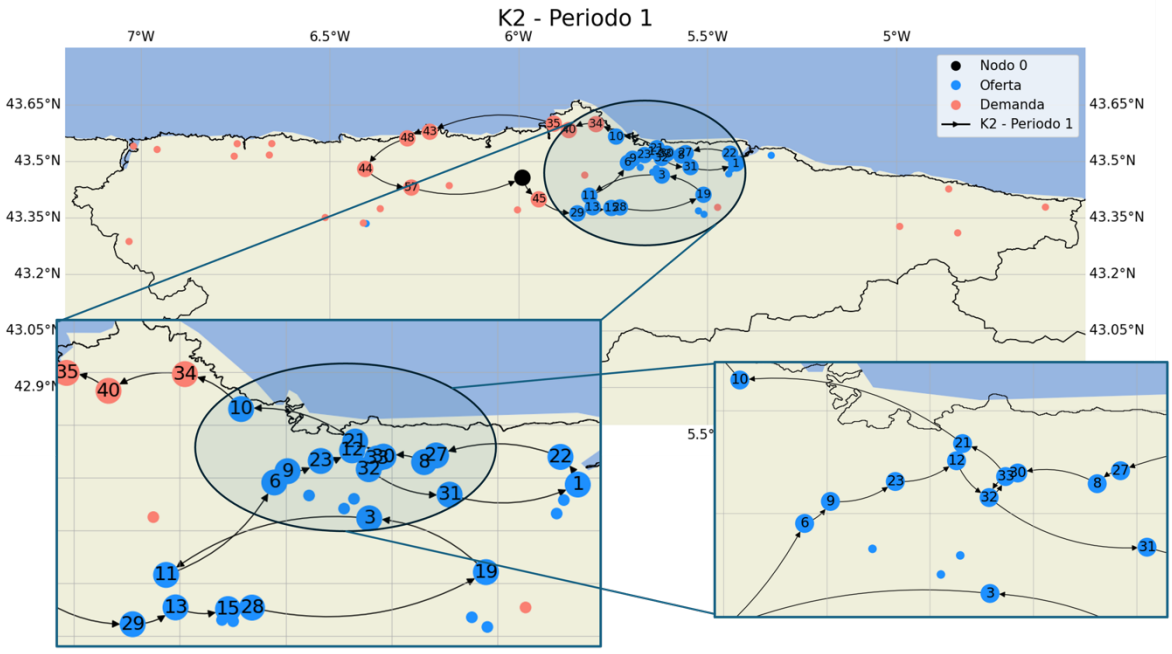

Figura 5.2.- Recorrido del vehículo K2 durante el periodo 1.

Cabe destacar que, en esta simulación, la solución óptima no utiliza ningún inventario. Es decir, no entrega más demanda de la que se solicita en ese periodo a ningún nodo para que sea consumida en el siguiente periodo. De la misma manera, no le pide a ningún llagar que mantenga parte o la totalidad la oferta no recogida en ese periodo para su recogida en el siguiente periodo. Esto se debe a la penalización puesta en la función objetivo para cada kg de magaya que se ubicase en un inventario. Lo ideal es

que los inventarios sean siempre 0. Se puede observar en la Tabla 5.2 la cantidad de producto no entregado a las ganaderías en cada periodo

Periodo	Oferta Total	Oferta No Recogida	Demanda Total	Demanda no servida
1	28799	486	28.961	648
2	29683	680	29.003	0

Tabla 5.2.- Resultados de oferta y demanda.

En el segundo periodo, la demanda es satisfecha en su totalidad. En el primer periodo, sin embargo, esto no es así. Hay dos nodos que no consiguen recibir toda la demanda. Uno de ellos recibe parte, pero el vehículo no tiene suficiente capacidad. Al ver el recorrido mostrado por el vehículo K2 en la Figura 5.2, destaca que, para hacer una ruta lógica, sin viajes extra y con ello un aumento del coste, tiene que pasar por 7 nodos con demanda de manera consecutiva, por los que tiene que repartir la magaya recogida anteriormente.

En Asturias, como se puede ver en las gráficas, los llagares están bastante concentrados en una misma zona geográfica. Es seguramente por esto por lo que el modelo decide que lo óptimo es usar vehículos grandes. Se necesita mucha capacidad, ya que, para recorrer una ruta eficiente de un recorrido mínimo, el vehículo tendrá que pasar por muchos nodos con demanda seguidos. Los camiones necesitan tener una capacidad lo suficientemente grande para recoger de golpe prácticamente toda la oferta de los llagares que se encuentran concentrados para luego repartir el producto a las ganaderías. En esta simulación, los vehículos pequeños recorrerían demasiados kilómetros.

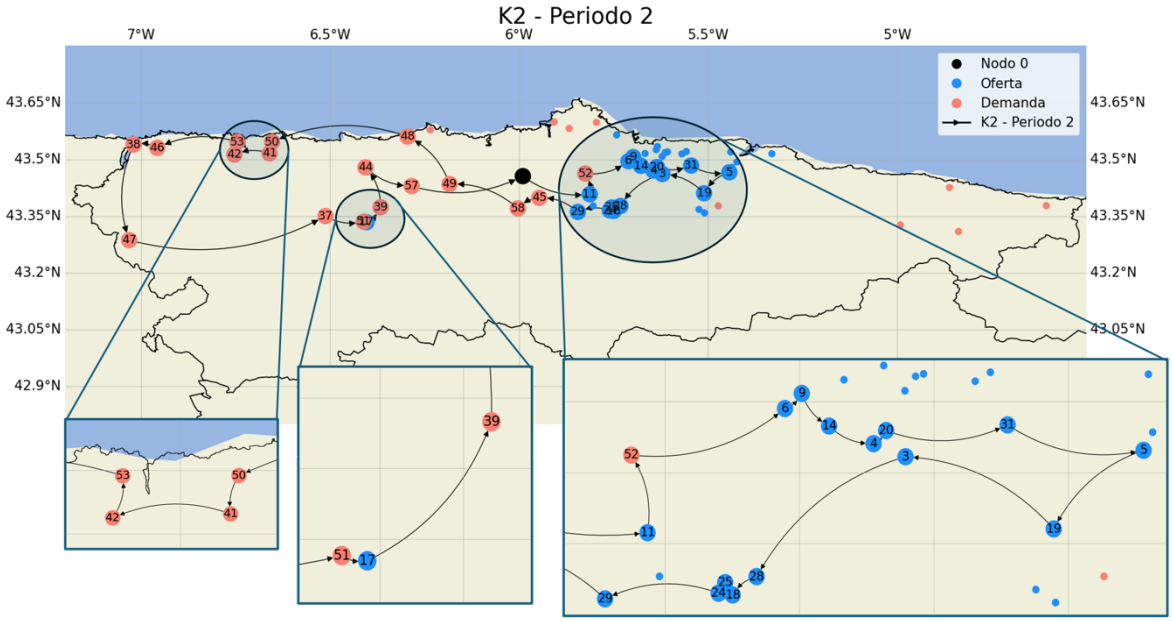

Figura 5.3.- Recorrido del vehículo K2 durante el periodo 2.

Esto se ve aún más claro en el segundo periodo. El vehículo K2 (Figura 5.3), recorre un único nodo con oferta de los últimos 17 nodos por los que pasa.

Para cada periodo, el vehículo K1, que es más grande y costoso, recorre los nodos con mayor cantidad de oferta y demanda, mientras que K2 hace viajes más largos donde recorre el resto de los nodos con menor cantidad de ambas.

El valor final de la función objetivo es de 64.800.865,86. La penalización impuesta en el modelo por cada kg de magaya no entregada a un nodo con demanda es de 100.000. Este número se ha considerado de manera arbitraria con el fin de forzar al modelo a que intente entregar siempre toda la demanda. Pasa lo mismo con el coste asignado a usar algún inventario, aunque en esta simulación no se ha usado ninguno. Teniendo esto en cuenta, restándole a la función objetivo esa penalización por cada kg de demanda no servida en la simulación (648 kg, como se puede ver en la Tabla 5.2), el valor real, en euros, para los dos días simulados es de 865,86 €.

5.2.- COSTE DE IMPLEMENTACIÓN

Para desarrollar e implementar el modelo tratado en este trabajo, se ha considerado un presupuesto que refleja los costes asociados tanto a la mano de obra como a los materiales necesarios. A continuación, se detalla el presupuesto considerando los distintos elementos del proyecto.

PRESUPUESTO MANO DE OBRA

1. Recopilación y Análisis de Datos

Actividad	Importe/Unidad	Unidad	Importe
1.1 Recopilación de datos de producción de magaya	30 €/h	40 h	1200,00 €
1.2 Análisis de datos de producción de magaya	25 €/h	50 h	1250,00 €
1.3 Recopilación de datos de demanda de ganaderías	30 €/h	20 h	600,00 €
1.4 Análisis de datos sobre demanda de ganaderías	25 €/h	25 h	625,00 €
1.5 Extracción y limpieza de datos	20 €/h	30 h	600,00 €
		Subtotal	**4.275,00 €**

2. Desarrollo del Modelo

Actividad	Importe/Unidad	Unidad	Importe
2.1 Formulación del modelo de programación lineal	25 €/h	60 h	1.500,00 €
2.2 Implementación en Python y PuLP	30 €/h	80 h	2.400,00 €
2.3 Validación y ajuste del modelo	25 €/h	40 h	1000,00 €
		Subtotal	**4.900,00 €**

3. Simulación y Pruebas

Actividad	Importe/Unidad	Unidad	Importe
3.1 Simulación con datos reales	30 €/h	50 h	1500,00 €
3.2 Análisis de resultados y optimización	40 €/h	60 h	2400,00 €
3.3 Documentación y reporte de resultados	20 €/h	30 h	600,00 €
		Subtotal	**4.500,00 €**

TOTAL MANO DE OBRA: 13.675,00 €

PRESUPUESTO DE EJECUCIÓN MATERIAL

1. Materiales de Oficina y Software

Material	Importe/Unidad	Unidad	Importe
Licencia de software GUROBI	0 €	1 ud	0,00 €
Equipos informáticos	2.000 €	1 ud	2.000,00 €
Material de oficina	200 €	1 ud	200,00 €
		Subtotal	**2.200,00 €**

TOTAL EJECUCIÓN MATERIAL: 2.200,00 €

Presupuesto de Ejecución por Contrata

	Importe
Presupuesto de Mano de Obra	13.675,00 €
Presupuesto de Ejecución Material	2.200,00 €
Gastos Generales (13%)	2.063,75 €
Beneficio Industrial (10%)	1.587,50 €
Total	**19.526,25 €**
IVA (21%)	4.100,52 €
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	**23.626,77 €**

El presupuesto total de ejecución por contrata asciende a la cantidad de VEINTITRÉS MIL SEISCIENTOS VEINTISÉIS EUROS CON SETENTA Y SIETE CÉNTIMOS (23.626,77 €).

5.3.- PLANIFICACIÓN TEMPORAL

La implementación del modelo de optimización requiere una planificación detallada para asegurar que todas las actividades se realicen de manera eficiente y dentro del plazo previsto. A continuación, se presenta un cronograma del proyecto, dividido en varias fases clave, con una estimación del tiempo necesario para cada tarea.

- Recopilación y análisis de datos: En esta primera fase del proyecto se obtienen y preparan los datos necesarios para la implementación del modelo. En total, se estima una duración de 15 días para completarse. Esta fase está formada por las siguientes tareas:
    - Recopilación de datos de producción de magaya (40 horas)
    - Análisis de datos de producción de magaya (50 horas)
    - Recopilación de datos de demanda de ganaderías (20 horas)

  - Análisis de datos de demanda de ganaderías (25 horas)
  - Extracción y limpieza de datos (30 horas)

- Desarrollo del modelo: En la segunda fase del proyecto, se desarrollará y validará el modelo de optimización. Esta fase dura alrededor de 15 días. Las tareas que la forman son las siguientes:
  - Formulación del modelo de programación lineal (60 horas)
  - Implementación en Python y PuLP (80 horas)
  - Validación y ajuste del modelo (40 horas)

- Simulación y pruebas: En esta última fase, que toma unos 11 días, se simulan el modelo con datos reales y se analizan y validan los resultados. Las tareas para realizar son las siguientes:
  - Simulación con datos reales (50 horas)
  - Análisis de resultados y optimización (60 horas)
  - Documentación y reporte de resultados (30 horas)

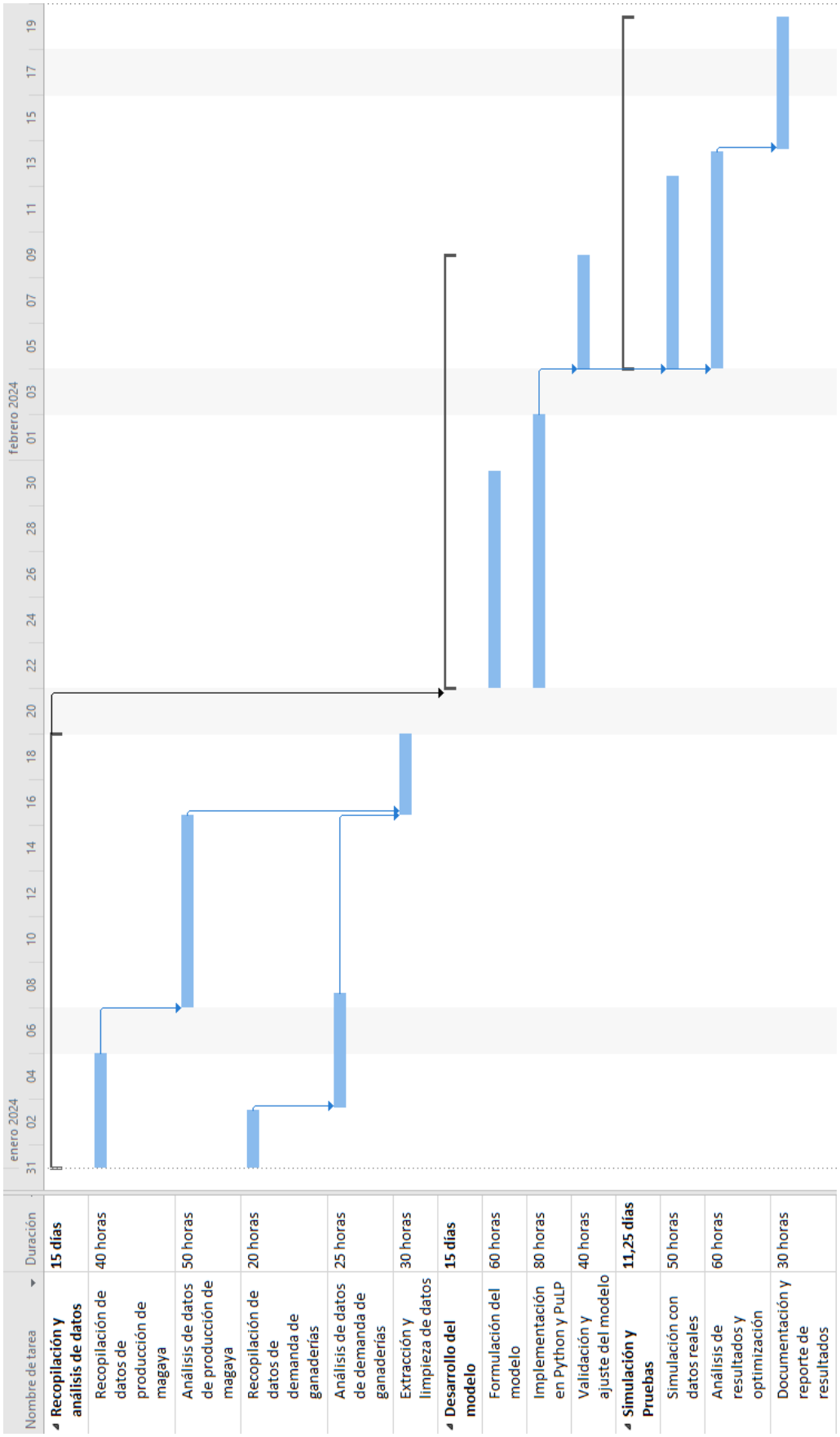

Figura 5.4.- Diagrama de Gantt.

6.- Conclusiones

El presente trabajo ha desarrollado y evaluado un modelo de optimización para la distribución de magaya en Asturias, destinado a mejorar la eficiencia logística y la sostenibilidad del proceso. A través de un análisis exhaustivo y una implementación detallada, se han alcanzado conclusiones significativas que resaltan la viabilidad y efectividad del modelo propuesto.

El modelo de optimización ha demostrado ser una herramienta eficaz para minimizar los costes operativos asociados con la logística de distribución de magaya. Al aplicar técnicas de programación lineal y algoritmos avanzados de enrutamiento, se ha logrado reducir las distancias recorridas y, consecuentemente, los gastos en combustible y mantenimiento de los vehículos. El enfoque de optimización garantiza que los costes sean minimizados al máximo dentro de las condiciones dadas.

El aprovechamiento de la magaya como alimento para el ganado contribuye significativamente a la sostenibilidad ambiental. Al reutilizar este subproducto, se reduce el desperdicio de alimentos y se promueve una economía circular. Además, la optimización de las rutas de transporte conlleva una disminución de la huella de carbono, mejorando la sostenibilidad del sistema de distribución.

El modelo desarrollado permite flexibilidad en la selección de vehículos y en la planificación de rutas. Esto asegura que el modelo pueda adaptarse a diferentes escenarios y demandas, ofreciendo soluciones viables y prácticas para la distribución de magaya.

Una estrategia clave en el modelo ha sido la selección manual de arcos basados en rutas reales, lo que ha permitido reducir el coste computacional sin sacrificar demasiado la precisión y realismo del modelo. Esta metodología asegura que las soluciones propuestas sean viables en la práctica, reflejando las condiciones del terreno.

El trabajo realizado sienta las bases para futuras investigaciones. Se sugiere la exploración de métodos de optimización más avanzados, como las metaheurísticas, para poder trabajar con un mayor volumen de datos. Además, investigar el impacto de las variaciones

estacionales en la producción de magaya y su demanda puede permitir ajustes más precisos al modelo, optimizando su funcionamiento durante todo el año.

7.- Bibliografía

Arronis, V. (2006). *Sistemas intensivos de producción Bovina. Alimentación*. http://www.mag.go.cr/bibliotecavirtual/AV-0887.PDF

Bedriñana, R., Picinelli, A., Rodríguez, R., Loureiro, M. D., & Suárez, B. (2021). Una segunda vida para la magaya de sidra. *El Campo de Asturias*, *60*, 26.

Çetinyamaç, E. (2024). *Vehicle Routing Problem: An Optimization Solution with Tabu Search Algorithm*.

Complexity Classes. (s. f.). Recuperado 10 de julio de 2024, de https://brilliant.org/wiki/complexity-classes/

Confederación Española de Transporte de Mercancías. (2024, mayo). *Precios del gasóleo, evolución mes a mes*. https://www.cetm.es/evolucion-precios-gasoleo/

Díaz, A., Mar, J., & Calzada, L. (2021). *Formulación de modelos programación matemática*. Paraninfo. https://www.paraninfo.es/catalogo/9788413661100/formulacion-de-modelos-programacion-matematica

Hispalink-Asturias. (2024, junio 26). *Predicciones económicas Cornisa Cantábrica*. https://www.unioviedo.es/hispalink/

Serida (2018). La magaya de la actividad sidrera: Fuente de compuestos bioactivos de elevado interés. Ácidos triterpénicos. (2018). http://www.serida.org/publicacionesdetalle.php

Li, Y., Han, B., Zhao, P., & Yang, R. (2023). Collaborative optimization for train stop planning and train timetabling on high-speed railways based on passenger demand. *PLoS One*, *18*(4), e0284747. https://doi.org/10.1371/journal.pone.0284747

Nanda Kumar, S., & Panneerselvam, R. (2012). A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, *04*. https://doi.org/10.4236/iim.2012.43010

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation.

Ministerio de Agricultura, Pesca y Alimentación (2023). Resultados de las encuestas de ganado bovino noviembre 2023. https://www.mapa.gob.es/es/estadistica/temas/estadisticas-agrarias/resultados_nov2023_bovinod_tcm30-675696.pdf

Robere, R. (2012). *Interior Point Methods and Linear Programming*. https://api.semanticscholar.org/CorpusID:14420172

SABI. (2024). *Sistema de Análisis de Balances Ibéricos*. https://sabi.informa.es/

The sustainable Development Goals Report 2023. (2023).

https://desapublications.un.org/publications/sustainable-development-goals-report-2023-special-edition

Wahyuningsih, S., Satyananda, D., & Hasanah, D. (2016). *Implementations of TSP-VRP variants for distribution problem*. *12*, 723-732.

Weisstein, E. W. (s. f.). *Traveling Salesman Problem*. Wolfram Research, Inc.

Anexo A

Tabla A.1.- Información sobre llagares y ganaderías.

NODO	NOMBRE	LATITUD	LONGITUD	INGRESOS (k€)	TIPO
1	VALLE, BALLINA Y FERNANDEZ, SA	43,493218	-5,423838	19.388	L
2	MANUEL BUSTO AMANDI SA	43,514838	-5,331241	11.654	L
3	SIDRA MENENDEZ SL	43,461208	-5,620963	4.913	L
4	SIDRA TRABANCO SA	43,470417	-5,644721	4.006	L
5	SIDRA CORTINA CORO SL	43,465735	-5,443813	3.196	L
6	INDUSTRIAL ZARRACINA SA	43,495166	-5,710661	2.386	L
7	SIDRA ANGELON SL	43,367538	-5,523982	2.200	L
8	LLAGAR CASTAÑON SL	43,514428	-5,569103	1.654	L
9	LAGARES DE GIJON SL.	43,505838	-5,698182	1.553	L
10	SIDRA PEÑON SL	43,564689	-5,742018	1.422	L
11	SIDRA FRAN SL	43,407625	-5,812945	942	L
12	INDUSTRIAS DERIVADAS DE LA MANZANA SL	43,525543	-5,637159	937	L
13	LLAGAR HERMINIO SL	43,376879	-5,804030	931	L
14	MACRILE SL	43,482846	-5,677829	878	L
15	SIDRA MUÑIZ SA	43,375267	-5,754421	782	L
16	SIDRA ESCANCIADOR SA	43,478484	-5,437045	724	L
17	COMERCIAL EMBOTELLADORA DIEZ SL	43,333774	-6,403656	680	L
18	SIDRA QUELO SL	43,363786	-5,749588	627	L
19	SIDRA FONCUEVA SL	43,410273	-5,510670	622	L
20	SIDRA CANAL SL	43,479655	-5,635361	603	L
21	SIDRA JR SL	43,533886	-5,634250	589	L
22	MARTINEZ SOPEÑA HERMANOS SL	43,519364	-5,440210	505	L
23	SIDRA CONTRUECES SL.	43,515493	-5,666773	468	L
24	SIDRA FANJUL SL	43,364997	-5,760090	422	L
25	VDA DE PALACIO SL	43,372457	-5,755150	404	L
26	SIDRA VIUDA DE CORSINO SA	43,358369	-5,509312	391	L
27	SIDRA FRUTOS SL	43,520719	-5,557823	387	L
28	LLANEZA MARTINEZ SL	43,376685	-5,731763	330	L
29	POMARADAS Y LLAGARES DE SARIEGO SL	43,360957	-5,844560	248	L
30	SIDRA ACEBAL SL	43,519616	-5,607549	240	L
31	SIDRA VALLINA SL	43,483643	-5,545004	229	L
32	HORNISAL SL	43,507714	-5,621410	196	L
33	SIDRA CABUENES S.L.	43,517901	-5,613432	188	L
34	GANADERÍA DIPLOMADA BADIOLA SL.	43,597727	-5,795173	58304	L

35	GANADERIA LA CORONA SL	43,598886	-5,907011	36332	G
36	GANADERIA SEVERIES SOCIEDAD AGRARIA DE TRANSFORMACION	43,425580	-4,861383	30984	G
37	GANADERIA MINGON HJL SOCIEDAD LIMITADA.	43,349862	-6,512211	29456	G
38	PERUYEIRA SA	43,539000	-7,019744	28118	G
39	SAT XACALEN Y GONZALEZ	43,373230	-6,366472	24987	G
40	GANADERIA BARDASQUERA SL.	43,581926	-5,867350	17462	G
41	GANADERIA RUFO SOCIEDAD LIMITADA	43,515691	-6,660150	16807	G
42	GANADERIA PALACIO SL.	43,512588	-6,753094	16133	G
43	GRANJA LAS TEERAS SL.	43,577551	-6,235306	15857	G
44	GANADERIA GALAN, SOCIEDAD COOPERATIVA ASTURIANA	43,478721	-6,406350	14494	G
45	GANADERIA ARUSE SL.	43,398183	-5,946701	13680	G
46	GANADERIA CAMPON SL	43,530429	-6,957177	11943	G
47	GANADERIA Y TURISMO FREIXE SL.	43,286290	-7,031521	9908	G
48	GANADERIA QUINTANA E IGLESIAS SL	43,559933	-6,294694	8526	G
49	LA MARTINIEGA, S. COOP. ASTUR.	43,434588	-6,183833	8066	G
50	GANADERIA EL MONTE SL	43,546012	-6,653594	7847	G
51	TINEOBOSS SL.	43,335250	-6,411271	7789	G
52	EL MOLINERO SERVICIOS AGROPECUARIOS SL.	43,462366	-5,824996	7689	G
53	GANADERIA ELOY SOCIEDAD LIMITADA	43,545767	-6,745172	6839	G
54	LA PUCHERINA SIGLO XXI S.L.	43,376829	-5,473205	5329	G
55	GANADERIA EL QUINTANAL SL	43,377457	-4,605306	4744	G
56	GANADERIA RUENES SL.	43,309092	-4,837639	3417	G
57	SOCIEDAD AGRARIA DE TRANSFORMACION JOMAFE	43,429330	-6,284111	3404	G
58	GANADERIA MENENDEZ ALVAREZ SL	43,370091	-6,003056	705	G
59	BOBIA GANADERA SOCIEDAD LIMITADA.	43,326262	-4,991167	611	G

Tabla A.2.- Datos de demanda y oferta promedios por cada llagar y ganadería.

NODO	NOMBRE	OFERTA (KG)	DEMANDA (KG)
1	VALLE, BALLINA Y FERNANDEZ, SA	8519	0
2	MANUEL BUSTO AMANDI SA	5121	0
3	SIDRA MENENDEZ SL	2159	0
4	SIDRA TRABANCO SA	1760	0
5	SIDRA CORTINA CORO SL	1404	0
6	INDUSTRIAL ZARRACINA SA	1049	0
7	SIDRA ANGELON SL	967	0
8	LLAGAR CASTAÑON SL	727	0
9	LAGARES DE GIJON SL.	683	0
10	SIDRA PEÑON SL	625	0
11	SIDRA FRAN SL	414	0
12	INDUSTRIAS DERIVADAS DE LA MANZANA SL	412	0
13	LLAGAR HERMINIO SL	409	0
14	MACRILE SL	386	0
15	SIDRA MUÑIZ SA	344	0
16	SIDRA ESCANCIADOR SA (EN LIQUIDACION)	318	0
17	COMERCIAL EMBOTELLADORA DIEZ SL	299	0
18	SIDRA QUELO SL	276	0
19	SIDRA FONCUEVA SL	273	0
20	SIDRA CANAL SL	265	0
21	SIDRA JR SL	259	0
22	MARTINEZ SOPEÑA HERMANOS SL	222	0
23	SIDRA CONTRUECES SL.	205	0
24	SIDRA FANJUL SL	186	0
25	VDA DE PALACIO SL	178	0
26	SIDRA VIUDA DE CORSINO SA	172	0
27	SIDRA FRUTOS SL	170	0
28	LLANEZA MARTINEZ SL	145	0
29	POMARADAS Y LLAGARES DE SARIEGO SL	109	0
30	SIDRA ACEBAL SL	106	0
31	SIDRA VALLINA SL	101	0
32	HORNISAL SL	86	0
33	SIDRA CABUENES S.L.	83	0
34	GANADERIA DIPLOMADA BADIOLA SL	0	4307
35	GANADERIA LA CORONA SL	0	2684
36	GANADERIA SEVERIES SOCIEDAD AGRARIA DE TRANSFORMACION	0	2289

37	GANADERIA MINGON HJL SOCIEDAD LIMITADA.	0	2176
38	PERUYEIRA SA	0	2077
39	SAT XACALEN Y GONZALEZ	0	1846
40	GANADERIA BARDASQUERA SL.	0	1290
41	GANADERIA RUFO SOCIEDAD LIMITADA	0	1242
42	GANADERIA PALACIO SL.	0	1192
43	GRANJA LAS TEERAS SL.	0	1171
44	GANADERIA GALAN, SOCIEDAD COOPERATIVA ASTURIANA	0	1071
45	GANADERIA ARUSE SL.	0	1011
46	GANADERIA CAMPON SL	0	882
47	GANADERIA Y TURISMO FREIXE SL.	0	732
48	GANADERIA QUINTANA E IGLESIAS SL	0	630
49	LA MARTINIEGA, S. COOP. ASTUR.	0	596
50	GANADERIA EL MONTE SL	0	580
51	TINEOBOSS SL.	0	575
52	EL MOLINERO SERVICIOS AGROPECUARIOS SL.	0	568
53	GANADERIA ELOY SOCIEDAD LIMITADA	0	505
54	LA PUCHERINA SIGLO XXI S.L.	0	394
55	GANADERIA EL QUINTANAL SL	0	350
56	GANADERIA RUENES SL.	0	252
57	SOCIEDAD AGRARIA DE TRANSFORMACION JOMAFE	0	251
58	GANADERIA MENENDEZ ALVAREZ SL	0	52
59	BOBIA GANADERA SOCIEDAD LIMITADA.	0	45

Anexo B

B.1.- Código en Python del modelo.

```
from pulp import *
import pandas as pd
import random

Modelo = LpProblem("Modelo", LpMinimize)

########## CONJUNTOS ##########

Nodos= list(range(60))

ArcosExistentes =  [(0, 11), (0, 35), (0, 40), (0, 43), (0, 45),
(0, 49), (0, 52), (0, 57), (0, 58), (1, 2), (1, 22), (1, 31), (2,
1), (2, 4), (2, 59), (3, 11), (3, 19), (3, 20), (3, 28), (3, 31),
(3, 36), (4, 3), (4, 14), (4, 20), (5, 16), (5, 19), (5, 31), (5,
54), (5, 59), (6, 9), (6, 11), (6, 14), (6, 52), (7, 19), (7, 28),
(7, 54), (8, 27), (8, 30), (9, 6), (9, 10), (9, 14), (9, 23), (10,
9), (10, 21), (10, 23), (10, 34), (10, 52), (11, 0), (11, 3), (11,
6), (11, 13), (11, 15), (11, 52), (12, 21), (12, 23), (12, 32),
(12, 33), (13, 11), (13, 15), (13, 29), (14, 4), (14, 6), (14, 9),
(14, 23), (15, 11), (15, 13), (15, 25), (15, 28), (16, 5), (17,
39), (17, 51), (18, 24), (18, 25), (18, 28), (19, 3), (19, 5), (19,
7), (19, 28), (19, 31), (19, 54), (20, 3), (20, 4), (20, 23), (20,
23), (20, 31), (20, 32), (20, 32), (21, 10), (21, 12), (21, 33),
(22, 1), (22, 27), (22, 31), (23, 9), (23, 10), (23, 12), (23, 14),
(23, 20), (23, 20), (24, 18), (24, 25), (24, 29), (25, 15), (25,
18), (25, 24), (26, 54), (27, 8), (27, 22), (27, 31), (28, 3), (28,
7), (28, 15), (28, 18), (28, 19), (29, 13), (29, 24), (29, 45),
(30, 8), (30, 33), (31, 1), (31, 3), (31, 5), (31, 19), (31, 20),
(31, 22), (31, 27), (31, 32), (32, 12), (32, 20), (32, 20), (32,
31), (32, 33), (33, 12), (33, 21), (33, 30), (33, 32), (34, 10),
(34, 40), (34, 52), (35, 0), (35, 40), (35, 43), (36, 2), (36, 55),
(36, 56), (36, 59), (37, 41), (37, 44), (37, 47), (37, 51), (38,
46), (38, 47), (39, 17), (39, 44), (39, 57), (40, 0), (40, 34),
(40, 35), (40, 52), (41, 37), (41, 42), (41, 50), (41, 53), (42,
41), (42, 46), (42, 47), (42, 50), (42, 53), (43, 0), (43, 35),
(43, 48), (43, 49), (43, 57), (44, 37), (44, 39), (44, 48), (44,
50), (44, 57), (45, 0), (45, 29), (45, 58), (46, 38), (46, 42),
(46, 53), (47, 37), (47, 38), (47, 42), (48, 43), (48, 44), (48,
49), (48, 50), (48, 57), (49, 0), (49, 43), (49, 48), (49, 57),
(49, 58), (50, 41), (50, 42), (50, 44), (50, 48), (50, 53), (51,
17), (51, 37), (52, 0), (52, 6), (52, 10), (52, 11), (52, 34), (52,
40), (53, 41), (53, 42), (53, 46), (53, 50), (54, 5), (54, 7), (54,
19), (54, 26), (54, 59), (55, 36), (55, 56), (56, 36), (56, 55),
(56, 59), (57, 0), (57, 39), (57, 43), (57, 44), (57, 48), (57,
49), (57, 58), (58, 0), (58, 45), (58, 49), (58, 57), (59, 2), (59,
5), (59, 36), (59, 54), (59, 56)]

NodosOrigen = [Arco[0] for Arco in ArcosExistentes]
NodosDestino = [Arco[1] for Arco in ArcosExistentes]
```

```
50
51  ArcosSalientes = {Nodo: [] for Nodo in Nodos}
52  ArcosEntrantes = {Nodo: [] for Nodo in Nodos}
53
54  for Origen, Destino in ArcosExistentes:
55      ArcosSalientes[Origen].append((Origen, Destino))
56      ArcosEntrantes[Destino].append((Origen, Destino))
57
58  K = ['K1', 'K2', 'K3', 'K4', 'K5']
59
60  ########## PARÁMETROS ##########
61
62  Periodos = ['Periodo0', 'Periodo1', 'Periodo2']
63
64  Alquiler = {'K1': 90,
65              'K2': 50,
66              'K3': 20,
67              'K4': 10,
68              'K5': 6}
69
70  Horas = {'K1': 16,
71           'K2': 16,
72           'K3': 16,
73           'K4': 16,
74           'K5': 16}
75
76  Capacidad = {'K1': 20000,
77               'K2': 10000,
78               'K3': 3000,
79               'K4': 1500,
80               'K5': 750}
81
82  Input1 = 'Matriz de Distancias.xlsx'
83  Input2 = 'Nodos (0).xlsx'
84
85  MatrizD = pd.read_excel(Input1, sheet_name = 'Matriz de
86  Distancias', index_col = 0).astype(float)
87  DF = pd.read_excel(Input2)
88  OD = DF['O/D'].values
89
90  TiempoRuta = {}
91  CosteRuta = {}
92
93  for Arco in ArcosExistentes:
94      TiempoRuta[Arco] = float(MatrizD.loc[Arco]) * 0.015
95      CosteRuta[Arco] = {'K1': float(MatrizD.loc[Arco]) * 0.3 * 1.6,
96                         'K2': float(MatrizD.loc[Arco]) * 0.2 * 1.6,
97                         'K3': float(MatrizD.loc[Arco]) * 0.15 * 1.6,
98                         'K4': float(MatrizD.loc[Arco]) * 0.10 * 1.6,
99                         'K5': float(MatrizD.loc[Arco]) * 0.07 * 1.6}
100
101 PV = 0.15
102 Demanda = {}
103 for Nodo in Nodos:
104     Demanda[Nodo] = {}
105     for Periodo in Periodos:
```

```
        if Periodo == 'Periodo0':
            Demanda[Nodo][Periodo] = 0
        elif 34 <= Nodo <= 59:
            Margen = OD[Nodo] * PV
            Demanda[Nodo][Periodo] = round(random.uniform(OD[Nodo]
- Margen, OD[Nodo] + Margen))
        else:
            Demanda[Nodo][Periodo] = 0

Oferta = {}
for Nodo in Nodos:
    Oferta[Nodo] = {}
    for Periodo in Periodos:
        if Periodo == 'Periodo0':
            Oferta[Nodo][Periodo] = 0
        elif 1 <= Nodo <= 33:
            Margen = OD[Nodo] * PV
            Oferta[Nodo][Periodo] = round(random.uniform(OD[Nodo] -
Margen, OD[Nodo] + Margen))
        else:
            Oferta[Nodo][Periodo] = 0

CosteInventario = {}
Penal = {}

for Nodo in Nodos:
    CosteInventario[Nodo] = 100
    Penal[Nodo] = 100000

########## VARIABLES DE DECISIÓN ##########

OfertaRecogida = LpVariable.dicts("OfertaRecogida", (Nodos,
Periodos[1:]), lowBound = 0, cat = 'Integer')
OfertaNoRecogida = LpVariable.dicts("OfertaNoRecogida", (Nodos,
Periodos[1:]), lowBound = 0, cat = 'Integer')
DemandaServida = LpVariable.dicts("DemandaServida", (Nodos,
Periodos[1:]), lowBound = 0, cat = 'Integer')
DemandaNoServida = LpVariable.dicts("DemandaNoServida", (Nodos,
Periodos[1:]), lowBound = 0, cat = 'Integer')
VContratados = LpVariable.dicts("VContratados", (K, Periodos),
lowBound = 0 , cat = 'Integer')
X = LpVariable.dicts("X", (Periodos[1:], K, ArcosExistentes),
lowBound = 0, cat = 'Integer')
CantidadTransportada = LpVariable.dicts("CantidadTransportada",
(Periodos[1:], K, ArcosExistentes), lowBound = 0, cat = 'Integer')
I = LpVariable.dicts("I", (Nodos, Periodos), lowBound = 0, cat =
'Integer')
U = LpVariable.dicts("U", (Nodos, Periodos[1:], K), lowBound =0,
cat = 'Integer')
Z = LpVariable.dicts("Z", (Periodos[1:], K, ArcosExistentes), cat =
'Binary')
P = LpVariable.dicts("P", (Nodos, Periodos[1:], K), lowBound = 0,
cat = 'Integer')

########## FUNCIÓN OBJETIVO ##########

```

```
Modelo += lpSum(Alquiler[Tipo] * VContratados[Tipo][Periodo]
          for Tipo in K for Periodo in Periodos[1:]) +
lpSum(Penal[Nodo] * DemandaNoServida[Nodo][Periodo] +
CosteInventario[Nodo] * I[Nodo][Periodo]
          for Nodo in Nodos for Periodo in Periodos[1:]) +
lpSum(CosteRuta[Arco][Tipo] * X[Periodo][Tipo][Arco]
          for Arco in ArcosExistentes for Tipo in K for Periodo in
Periodos[1:])

######### RESTRICCIONES ##########

### RESTRICCIÓN 1 ###
for Nodo in Nodos:
    for Periodo in Periodos[1:]:
        Modelo += DemandaServida[Nodo][Periodo] +
DemandaNoServida[Nodo][Periodo] == Demanda[Nodo][Periodo]

### RESTRICCIÓN 2 ###
for Nodo in Nodos:
    for Periodo in Periodos[1:]:
        Modelo += OfertaRecogida[Nodo][Periodo] +
OfertaNoRecogida[Nodo][Periodo] == Oferta[Nodo][Periodo]

### RESTRICCIÓN 3 ###
for N in Nodos:
    for Periodo in Periodos[1:]:
        for Tipo in K:
            Modelo += (lpSum(X[Periodo][Tipo][Arco] for Arco in
ArcosSalientes[N]) - lpSum(X[Periodo][Tipo][Arco] for Arco in
ArcosEntrantes[N]) == 0)
            Modelo += (lpSum(Z[Periodo][Tipo][Arco] for Arco in
ArcosSalientes[N]) - lpSum(Z[Periodo][Tipo][Arco] for Arco in
ArcosEntrantes[N]) == 0)

### RESTRICCIÓN 4 ###
for Tipo in K:
    for Periodo in Periodos[1:]:
        Modelo += lpSum(TiempoRuta[Arco] * X[Periodo][Tipo][Arco]
for Arco in ArcosExistentes) <= Horas[Tipo] *
VContratados[Tipo][Periodo]

### RESTRICCIÓN 5 ###
for Tipo in K:
    Modelo += VContratados[Tipo][Periodos[0]] == 0

### RESTRICCIÓN 6 ###
for Periodo in Periodos[1:]:
    for Tipo in K:
        Modelo += lpSum(X[Periodo][Tipo][Arco] for Arco in
ArcosSalientes[0]) == VContratados[Tipo][Periodo]

### RESTRICCIÓN 7 ###
for Periodo in Periodos[1:]:
    for Tipo in K:
        Modelo += lpSum(X[Periodo][Tipo][Arco] for Arco in
ArcosEntrantes[0]) == VContratados[Tipo][Periodo]
```

```
218
219 ### RESTRICCIÓN 8 ###
220 for Periodo in Periodos[1:]:
221     for Tipo in K:
222         for Arco in ArcosSalientes[0]:
223             Modelo += CantidadTransportada[Periodo][Tipo][Arco] ==
224 0
225         for Arco in ArcosEntrantes[0]:
226             Modelo += CantidadTransportada[Periodo][Tipo][Arco] ==
227 0
228
229 ### RESTRICCIÓN 9 ###
230 for Periodo in Periodos[1:]:
231     for Tipo in K:
232         for Arco in ArcosExistentes:
233             Modelo += CantidadTransportada[Periodo][Tipo][Arco] <=
234 Capacidad[Tipo] * X[Periodo][Tipo][Arco]
235
236 ### RESTRICCIÓN 10 ###
237 for Nodo in Nodos:
238     Modelo += I[Nodo][Periodos[0]] == 0
239
240 ### RESTRICCIÓN 11 ###
241 for Nodo in Nodos[1:]:
242     for Periodo in Periodos[1:]:
243         Modelo += OfertaRecogida[Nodo][Periodo] - I[Nodo][Periodo]
244 + I[Nodo][Periodos[Periodos.index(Periodo) - 1]] +
245 lpSum(CantidadTransportada[Periodo][Tipo][Arco]
246                   for Arco in ArcosEntrantes[Nodo] for Tipo in K) -
247 DemandaServida[Nodo][Periodo] -
248 lpSum(CantidadTransportada[Periodo][Tipo][Arco]
249                   for Arco in ArcosSalientes[Nodo] for Tipo in K)
250 == 0
251
252 ### RESTRICCIÓN 12 ###
253 M = 100000
254 for Tipo in K:
255     for Periodo in Periodos[1:]:
256         Modelo += U[Nodos[0]][Periodo][Tipo] == 1
257
258 for Tipo in K:
259     for Periodo in Periodos[1:]:
260         for Nodo in Nodos:
261             Modelo += U[Nodo][Periodo][Tipo] <= len(Nodos)
262             Modelo += U[Nodo][Periodo][Tipo] >= 1
263
264 for Tipo in K:
265     for Periodo in Periodos[1:]:
266             for NodoI in Nodos:
267                 for NodoJ in Nodos:
268                     if NodoJ != NodoI and NodoJ != 0 and (NodoI,
269 NodoJ) in ArcosExistentes:
270                         Modelo += U[NodoJ][Periodo][Tipo] -
271 U[NodoI][Periodo][Tipo] - 1 - M * (Z[Periodo][Tipo][(NodoI, NodoJ)]
272 - 1) >= 0
273
```

```
### RESTRICCIÓN 13 ###
for Nodo in Nodos[1:]:
    for Periodo in Periodos[1:]:
        for Tipo in K:
            Modelo += P[Nodo][Periodo][Tipo] ==
lpSum(Z[Periodo][Tipo][AE] for AE in ArcosEntrantes[Nodo]) +
lpSum(Z[Periodo][Tipo][AS] for AS in ArcosSalientes[Nodo])

for i in Nodos:
    for j in Nodos:
        for Periodo in Periodos[1:]:
            for Tipo in K:
                if (i, j) in ArcosExistentes:
                    Modelo += X[Periodo][Tipo][(i, j)] <= M *
P[i][Periodo][Tipo]

for Arco in ArcosExistentes:
    for Periodo in Periodos[1:]:
        for Tipo in K:
            Modelo += X[Periodo][Tipo][Arco] >=
Z[Periodo][Tipo][Arco]

Modelo.solve(GUROBI_CMD(msg=True))
value(Modelo.objective)

print("Valores de las variables:")
for var in Modelo.variables():
    if var.varValue>0:
        print(f"{var.name}: {var.varValue}")


```

B.2.- Código usado para calcular el centro de gravedad (depósito).

```
import pandas as pd

Input = 'Nodos.xlsx'
DF = pd.read_excel(Input)

Latitudes = DF['LATITUD'].values
Longitudes = DF['LONGITUD'].values

Nodos = len(Latitudes)

MatrizD = pd.DataFrame(index = DF.index, columns = DF.index)

Tipos = DF['TIPO'].values
Cantidades = DF['CANTIDAD'].values

Pesos = []
for i in range(len(Tipos)):
    if Tipos[i] == 'OFERTA':
        Pesos.append(Cantidades[i] * 2)
    else:
        Pesos.append(Cantidades[i])

PesoTotal = sum(Pesos)
print(Pesos)
CentroLat = sum(Lat * Peso for Lat, Peso in zip(Latitudes, Pesos))
/ PesoTotal
CentroLon = sum(Lon * Peso for Lon, Peso in zip(Longitudes,
Pesos)) / PesoTotal

CentroLat = sum(Latitudes) / Nodos
centroLon = sum(Longitudes) / Nodos

print(f'Centro de gravedad. Latitud: {CentroLat:.6f}, Longitud:
{CentroLon:.6f}')
```

B.3.- Código usado para el cálculo de las distancias entre cada nodo.

```
import pandas as pd
import math

def Haversine(Lon1, Lat1, Lon2, Lat2):
    Lon1, Lat1, Lon2, Lat2 = map(math.radians, [Lon1, Lat1, Lon2, Lat2])

    DLon = Lon2 - Lon1
    DLat = Lat2 - Lat1

    A = math.sin(DLat/2) ** 2 + math.cos(Lat1) * math.cos(Lat2) * math.sin(DLon/2) ** 2
    C = 2 * math.atan2(math.sqrt(A), math.sqrt(1 - A))

    R = 6371.0

    Distancia = C * R

    return Distancia

Input = 'Nodos (0).xlsx'
DF = pd.read_excel(Input)

Latitudes = DF['LATITUD'].values
Longitudes = DF['LONGITUD'].values

Nodos = len(Latitudes)

MatrizD = pd.DataFrame(index = DF.index, columns = DF.index)
print(MatrizD)

for i in range(Nodos):
    for j in range(Nodos):
        if i != j:
            MatrizD.iat[i, j] = Haversine(Longitudes[i], Latitudes[i], Longitudes[j], Latitudes[j])
        else:
            MatrizD.iat[i, j] = 0.0

Output = 'Matriz de Distancias.xlsx'
with pd.ExcelWriter(Output, engine = 'openpyxl') as writer:
    MatrizD.to_excel(writer, sheet_name = 'Matriz de Distancias')

print(f'Matrices guardadas en {Output}')
```