Universidad
de La Laguna

Resolución de un problema de rutas con ventanas de tiempo y periodicidad Aplicado a una empresa canaria

Solving a routing problem with time windows and periodicity Applied to a Canarian company

Alfonso Ruigómez González

Trabajo de Fin de Grado

Matemáticas, Estadística e Investigación Operativa

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 15 de Julio de 2015

Dr. D. **Hipólito Hernández Pérez**, con N.I.F. 45.452.715-T profesor Titular de Universidad adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

"*Resolución de un problema de rutas con ventanas de tiempo y periodicidad*"

ha sido realizada bajo su dirección por D. **Alfonso Ruigómez González**, con N.I.F. 78.854.085-B.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 15 de Julio de 2015.

Agradecimientos

A Hipólito Hernández por todo su tiempo dedicado para que esto saliera adelante.
A mis padres y mi abuelo Zenón.
En especial a mi abuela y tía Blanca, que aunque no estén, seguro que estarían muy orgullosas.
Gracias de todo corazón.

Resumen

El transporte, así como sus costes, están presentes en una gran cantidad de servicios. El incremento en el precio de los carburantes, así como el auge que se está viviendo en el comercio electrónico, están convirtiendo al transporte en uno de los costes más significativos de muchas compañías. Uno de los objetivos de la logística del transporte consiste en optimizar las rutas que tienen que seguir un conunto de vehículos. Estos vehículos generalmente parten de un único lugar llamado depósito. Las rutas deben satisfacer las demandas de los clientes, teniendo en cuenta la periodicidad de visita, los horarios y otros muchos factores. Es habitual que las compañías se enfrenten a estos problemas de cálculo de rutas de transporte de forma manual. En el presente trabajo se intentará modelar el problema de rutas para la empresa Biocontrol Canarias S.L.

Este trabajo se planteó como continuación de un proyecto anterior. Tras realizar las prácticas en Biocontrol Canarias S.L. se propuso estudiar y modelar la forma de optimizar la recogida de muestras de esta empresa. Anteriormente se consiguió modelar este problema con un "software" de pago llamado Xpress Mosel. El objetivo ahora es intentar modelar este programa con un "software" gratuito, resolviendo todos los inconvenientes que estos tipos de resolutores presentan.

Palabras clave: Optimización combinatoria, problema del viajante de comercio, problemas de rutas de vehículos, software libre, Gusek.

Abstract

Transport, as well as its costs, are presented within a large number of services. The increase in the price of the fuel plus the progress that is currently being experienced in the electronic field, are turning transport into one of the most significant costs for many companies. One of the main goals of transport logistics is to model in the most optimal way possible a route set, made by a number of vehicles, leaving from a place called depot. This routes must satisfy the demands of the customers, considering the frequency of visit, schedules and many other factors. It is normal for companies to face these route transportation problems in a manual way. In the current project, it is tried to model the route problems for the company Biocontrol Canarias S.L.

This project began as a continuation of a previous project. After performing the internship at Biocontrol Canarias S.L. it was proposed to study and model the way of optimising the collection of samples for this company. In the past, they achieved the modelling of this problem with a purchased software called Xpress Mosel. Following on from this, the objective today is to model this program with free software, resolving all the inconveniences that these types of resolutions present.

Keywords: *Combinatorial optimization, traveling salesman problem, vehicle routing problem, free software, Gusek.*

Índice general

1. Introducción **1**
1.1. Biocontrol Canarias S.L. . . . 2
1.2. Objetivos . . . 4

2. Herramientas matemáticas **5**
2.1. Problemas . . . 5
2.1.1. Problema del Viajante de Comercio . . . 5
2.2. Resolutores . . . 10
2.2.1. Resolutores de pago . . . 10
2.2.2. Resolutores Libres . . . 11

3. Formulación del problema **15**
3.1. Características . . . 15
3.2. Modelo con variables arco . . . 16
3.2.1. Datos . . . 16
3.2.2. Parámetros y variables . . . 16
3.2.3. Modelo matemático . . . 18
3.2.4. Lectura de datos . . . 19
3.3. Modelo con variables rutas . . . 20
3.3.1. Parámetros y variables . . . 21
3.3.2. Modelo matemático . . . 22
3.3.3. Lectura de datos . . . 22
3.3.4. Generación de rutas . . . 23
3.4. Modelo con variables ruta con vista a un año . . . 24
3.4.1. Parámetros y variables . . . 25
3.4.2. Modelo matemático . . . 26

4. Resultados computacionales **27**

5. Conclusiones **31**

A. Códigos con Gusek **32**
A.1. Modelo variables arcos . . . 32
A.2. Modelo variables rutas . . . 35

B. Códigos con Microsoft Solver Foundation y Visual Basic para Aplicaciones **37**
B.1. MSF 37
B.2. VBA 39

Bibliografía **53**

Índice de figuras

1.1. Logo de Biocontrol Canarias S.L. 2

2.1. Comparación de diferentes resolutores 12
2.2. Gusek 14
2.3. Microsoft Solver Foundation 14

3.1. Tabla de datos 20
3.2. Fichero 23

Índice de cuadros

4.1. Resultados computacionales con $\alpha = 0{,}3$ 28
4.2. Resultados computacionales con $\alpha = 0{,}1$ 29

Capítulo 1

Introducción

Hoy en día, una de las ramas a la que más investigadores e investigadoras se dedican, debido a su interdisciplinariedad y la gran aplicabilidad que tiene, es la programación combinatoria. Los problemas de optimización combinatoria aparecen por la necesidad que muestran ciertos sectores empresariales. El objetivo principal de estos problemas es maximizar o minimizar uno o varios objetivos, bajo un gran número de posibles soluciones.

En el presente proyecto se trata un problema de rutas de vehículos aplicado a la empresa **Biocontrol Canarias S.L.**

La principal motivación de este trabajo fue el trabajado realizado, para esta misma empresa, por una alumna de matemáticas en 2014. En su caso, para modelar el problema utilizó el entorno XPress-Mosel, un "software" de pago que Biocontrol Canarias S.L. no pudo utilizar. Este año, tras haber realizado las prácicas de empresa en Biocontrol Canarias S.L., uno de los objetivos propuestos por mi tutor, Christian Aarón Martín González, fue realizar el problema de estudiar rutas óptimas para efectuar la recogida de muestras, esta vez utilizando un "software" libre al que ellos pudieran tener acceso.

Este proyecto consta de cinco capítulos.

Este primer capítulo es una introducción donde se presenta la empresa Biocontrol Canarias S.L. y se introduce los objetivos del problema de rutas de vehículos. En el segundo capítulo se introducen unos fundamentos teóricos sobre la programación matemática y un estudio de los diferentes tipos de "softwares" que pueden resolver este tipo de problemas. En el capítulo tercero, tras conocer que resolutor utilizaremos, procedemos al modelado del programa en concreto y, seguidamente, en el cuarto capítulo vemos los resultados obtenidos. Por último, se redactan las conclusiones de esta memoria.

1.1. Biocontrol Canarias S.L.

Biocontrol Canarias S.L. es una empresa Canaria fundada en el año 1996, siendo pionera en el sector de la seguridad alimentaria, consolidándose como una empresa de prestigio y especializada en este sector.

Surge con el fin de cubrir las primeras necesidades del sector de la alimentación en su mas amplio sentido, derivado de las nuevas normativas que sobre la seguridad e higiene de los alimentos emanaban de directivas comunitarias y legislación española.

Su trabajo se basa en la prevención, cumpliendo con el nivel de calidad exigido por la legislación técnico sanitario vigente. Para ello, se basan en la implantación de Sistemas de Análisis y Puntos de Control Críticos (APPCC), realización de auditorías, ensayos físico-químicos y microbiológicos de alimentos, superficies y agua.

Figura 1.1: Logo de Biocontrol Canarias S.L.

El equipo humano de Biocontrol Canarias está formado por un grupo multidisciplinar de profesionales de la calidad y seguridad alimentaria con una amplia experiencia en el sector, aportando soluciones adaptadas a las necesidades de cada empresa.

La empresa Biocontrol Canarias S.L. está formada por dos laboratorios propios, un laboratorio microbiológico y un laboratorio físico-químico. Además disponen de un aula de formación, un departamento técnico, administración y recepción.

En estas instalaciones Biocontrol Canarias garantiza a sus clientes la higiene y seguridad de sus alimentos, basándose en certificaciones y normas de calidad mediante el establecimiento de un sistema de gestión de calidad ISO 9001:2008.

Biocontrol Canarias S.L. ofrece tres tipos de servicios principalmente: Análisis de laboratorio, asesoramiento y formación.

Análisis de laboratorio

En el laboratorio se encargan del control microbiológico y físico-químico de materias primas, alimentos, bebida y agua. Analizan organolépticos de alimentos, bebidas y agua. También verifican la limpieza y desinfección de instalaciones y manipuladores, así como un control ambiental.

Formación

Biocontrol Canarias S.L. ofrece cursos de formación en sus propias instalaciones. Cuando se trata de grupos grandes (una empresa particular) se desplazan a sus establecimientos.

Entre los cursos que ofrecen destacan:

- Higiene y manipulación de alimentos.
- Implantación del Sistema de Autocontrol y APPCC.
- Limpieza y desinfección en la industria alimentaria.
- Trazabilidad y etiquetado.

Asesoramiento

Aparte de asesorar a sus clientes, disponen de un departamento de calidad. Entre todas sus funciones destacamos:

- Diseño, desarrollo e implantación de sistemas de autocontrol basados en los principios del APPCC.
- Auditorias sanitarias.
- Seguridad alimentaria.
- Tramitación y gestión de actas de inspecciones sanitarias.
- Etiquetado de alimentos
- Informe e interpretación de los resultados de analíticas obtenidos
- Solicitud de comunicación previa de empresas alimentarias de comercio al por menor al incluir en el registro autonómico canario.
- Solicitud del Registro General Sanitario de Empresas Alimentarias y Alimentos (RGSEAA)

1.2. Objetivos

Tal y como ya se ha expuesto, Biocontrol Canarias realiza análisis de laboratorio. Estos análisis se realizan a través de muestras que se recogen en los los locales de los diferentes clientes.

A fecha de hoy, la empresa tiene 125 clientes, la gran mayoría en Tenerife. El resto en otras islas. El número de clientes puede variar a lo largo del tiempo.

Por lo tanto, el objetivo consiste en modelar el problema de rutas, estableciendo un itinerario que les permitan visitar a los clientes respetando sus preferencias en cuanto al periodo de tiempo previsto de visita y así poder realizar la planificación de recogidas de muestras de la forma mas óptima posible.

Las características del problema son:

- Hay 125 clientes, un vehículo y un depósito (Biocontrol Canarias S.L.).
- Se realiza a lo sumo una ruta cada día.
- Las distancias y los tiempos de desplazamiento son conocidos.
- Hay que respetar unas ventanas de tiempo según el cliente, es decir, dentro de un horario predeterminado.
- Las visitas solo pueden realizarse durante el horario de Biocontrol Canarias S.L., salvo cuando se visite en otras islas. En estos casos se rebasa el tiempo de llegada a Biocontrol Canarias (fijado por defecto a las 14:00 horas).
- Cada cliente tiene unas preferencias de visita, desde semanal hasta anual.
- Algunas empresas pueden visitarse cualquier día de la semana, pero otras tienen que ser visitadas en algún o algunos días en concreto.
- Dependiendo del tipo de trabajo, se necesitará un tiempo de proceso diferente.

Todos estos datos están recogidos en una tabla de Excel que Biocontrol Canarias facilita con los datos y requerimientos de cada empresa.

Sabiendo esto, el objetivo es modelar el programa en un “software” libre o de bajo costo.

Tras haber descrito los objetivos y las características del problema, introduciremos en el siguiente capítulo algunos conceptos teóricos sobre la programación combinatoria.

Capítulo 2

Herramientas matemáticas

2.1. Problemas

La programación matemática se apoya en modelos matemáticos, donde el objetivo a satisfacer se denomina función objetivo, la cual debe ser maximizada o minimizada. Esta función objetivo esta sujeta a una serie de restricciones formuladas como ecuaciones o inecuaciones.

2.1.1. Problema del Viajante de Comercio

El Problema del Viajante de Comercio, "Traveling Salesman Problem (TSP)", debido a su complejidad computacional es uno de los problemas más complejos que se conoce de la programación matemática.

Una visión general de este problema y que sirve para hacernos una idea, sería plantearnos la siguiente pregunta como se plantea Lorena-Stockdale [9] en su artículo:
Un viajante quiere visitar n ciudades, una y sólo una vez cada una, empezando por cualquiera de ellas y regresando al mismo lugar del que partió (depósito). Supongamos que la distancia entre cada dos ciudades es conocida. ¿Cuál es la ruta óptima que debe elegir para visitar todas estas ciudades?

Este problema se conoce como el problema del viajante, sencillo de enunciar pero complejo de resolver. De hecho, no existe ningún método de resolución capaz de obtener el resultado en tiempo polinomial. Esto se conoce como un problema NP-duro. Uno de los hallazgos más importantes del TSP se puede consultar Applegate, Bixby, Chvátal y Cook [4], en el cual se plantea resolver el problema el TSP para 85900 nodos.

Comenzaremos definiendo unos conceptos clave para el desarrollo de este problema.

Al par $G = (N, A)$, lo conoceremos como grafo, en donde $N = \{1, 2, ..., n\}$ es el conjunto de ciudades (nodos) y A el conjunto de ramas o arcos que denotan la conexión entre

ellas. Cada arco $a \in A$ lo denotaremos por (i, j) donde i sería el origen y j el destino. Denotaremos por c_{ij} el costo de ir de la ciudad i a la ciudad j.

A la sucesión de nodos $u_1, u_2, ..., u_n$ tal que $u_1, u_2, ..., u_{n-1}$ son distintos y $u_1 = u_n$ se le conoce como ciclo. Si el ciclo contiene todos los vértices se le conoce como ciclo hamiltoniano.

Matemáticamente podemos pensar en el TSP como un problema de hallar un ciclo hamiltoniano de mínima distancia en un grafo completo $G = (N, A)$.

Existen dos variantes dependiendo de como son los costos de viaje:

- Problema del viajante simétrico (STSP)

$$c_{ij} = c_{ji}, \forall (i, j) \in A$$

- Problema del viajante asimétrico (ATSP)

$$c_{ij} \neq c_{ji}, \forall (i, j) \in A$$

En el caso del STSP, el número de soluciones se reduce a la mitad.
Nuestro caso, sin embargo, sería asimétrico ya que nos podemos encontrar con calles de un solo sentido. Este tipo de grafos se conoce como grafos dirigidos

Si se desea ampliar información se recomienda Salazar-González [1].

Modelo matemático ATSP

Definimos nuestro grafo dirigido $G = (N, A)$, tal que:
$\delta^+(i) := |j \in N : (i, j) \in A| \equiv$ número de arcos que tienen a i como origen.
$\delta^-(i) := |j \in N : (j, i) \in A| \equiv$ número de arcos que tienen a i como destino.
Se define la variable x_a, como una variable de decisión asociada a cada uno de los arcos $a \in A$:

$$x_a = \begin{cases} 1, & \text{si el arco } a \text{ forma parte de la solución óptima} \\ 0, & \text{en otro caso} \end{cases}$$

Además, se define un subconjunto S contenido en N que ayudará a que no se produzcan subciclos.

$$\text{mín} \sum_{a \in A} c_a \cdot x_a \tag{2.1}$$

sujeto a:

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

$$\sum_{e\in E(S)} x_a \leq |S| - 1 \qquad \forall S \subset N \setminus \{1\} \tag{2.3}$$

$$x_a \in \{0, 1\} \tag{2.4}$$

Gracias a la restricción (2.2) conseguimos que la ruta llegue y salga de cada nodo exactamente una vez. Con la restricción (2.3) evita la aparición de subciclos, obligando a que de ese subconjunto S salga al menos una arista. La restricción (2.4) es una restricción de binariedad.

Este es el caso más sencillo del TSP. Para ayudarnos a modelar nuestro problema, veremos a continuación unas variantes del TSP relacionadas con los modelos propuestos en esta memoria. Para más información, se puede consultar el libro de Gutin y P. Punnen [8]. Estas variantes son:

- Problemas de rutas de vehículos (VRP).
- El problema del viajante de comercio con periodicidades (PTSP).
- El problema del viajante de comercio con ventanas de tiempo (TSPTW).

Empezaremos hablando acerca del problema de rutas de vehículos y seguidamente explicaremos los problemas de periodicidades y ventanas de tiempo.

Problemas de rutas de vehículos

En este caso, tenemos un conjunto de depósitos y de clientes, así como un conjunto de vehículos. El objetivo es visitar a todos estos clientes, comenzando y terminando cada ruta en los diferentes depósitos.

Las características de los clientes, depósitos y vehículos, así como diferentes restricciones operativas sobre las rutas, dan lugar a diferentes variantes del problema.

Una variante del VRP sería que los vehículos que dispone la empresa tengan capacidades limitadas. Este caso se conoce como Problemas de rutas de vehículos con capacidad limitada ("Capacited Vehicle Routing Problem, CVRP").

Se recomienda la lectura de Olivera [7] para mas información sobre este problema.

En el caso de Biocontrol Canarias S.L. se dispone de un solo vehículo, sin problema de capacidad.

El problema del viajante de comercio con periodicidades

Las visitas a los clientes pueden presentar distintas periodicidades, como por ejemplo semanales, quincenales, mensuales, etc. Es en estos casos cuando hablamos del problema del viajante con periodicidades.

En el artículo Paleta [2] se presenta un algoritmo heurístico para este problema.

Lo primero a tener en cuenta es que una misma ciudad no puede ser visitada dos o más veces el mismo día. Además, al finalizar cada día se debe volver a la ciudad inicial (depósito).

Para modelar el programa, se dispone un número K de días para visitar las N ciudades un número m_i de veces, $m_i \leq K$.

Siguiendo con el modelo descrito en la sección 2.1.1, ahora modificamos la variable x, siendo de la siguiente forma:

$$x_{ij}^k = \begin{cases} 1, & \text{si el arco } (i,j) \text{ está en la ruta el día } k \\ 0, & \text{en otro caso} \end{cases}$$

Es necesario introducir una nueva variable, y_i^k, tal que:

$$y_i^k = \begin{cases} 1, & \text{si el cliente } i \text{ se visita el día } k \\ 0, & \text{en otro caso} \end{cases}$$

El modelo matemático que nos queda sería:

$$\text{mín} \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij} \cdot x_{ij}^k \tag{2.5}$$

sujeto a:

$$\sum_{j\in N} x_{1j}^k = \sum_{j\in N} x_{j1}^k = y_i^k \qquad \forall k = 1,2,...,K \tag{2.6}$$

$$\sum_{i,j\in S} x_{ji}^k \leq |S| - 1 \qquad \forall S \subseteq N \setminus \{1\}, \forall k = 1,2,...,K \tag{2.7}$$

$$\sum_{k=1}^{K} y_i^k = m_i \qquad \forall i \in N \tag{2.8}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in N, \forall k = 1,...,K \tag{2.9}$$

Ahora, la restricción (2.6) nos obliga a que se entre y se salga del depósito todos los días. La restricción (2.8) indica el número de veces que es visitado el cliente i. La restricción (2.7) evita que haya un ciclo en un conjunto S que no contenga al 1 (el deposito).

A este modelo se le pueden añadir restricciones adicionales sobre la consecución de visitas a clientes. Por ejemplo, si un cliente no puede ser visitado dos días consecutivos se puede añadir la restricción:

$$y_i^k + y_i^{k+1} \leq 1, \quad \forall k = 1, ..., K-1 \tag{2.10}$$

El problema del viajante de comercio con ventanas de tiempo

En la vida real, no podemos visitar a los clientes en cualquier momento. A la hora de realizar una visita, se debe tener en cuenta su disponibilidad.

Esto se consigue con el problema del viajante con ventanas de tiempo. Igual que en los otros casos, el objetivo es encontrar el camino de coste mínimo, con la diferencia de que cada nodo dispone ahora de un periodo de tiempo fijado previamente.

Es decir, cada ciudad tiene un intervalo de tiempo, $[a_i, b_i]$, en el que tiene que ser visitada. Si el vehículo llega antes de la hora a_i, se permite esperar hasta que el nodo esté disponible, aunque esto suponga un aumento de costo, ya que estamos hablando de un tiempo en el que no se realiza ninguna acción.

El tiempo, t_{ij}, que se tarda de ir de la ciudad i a j es conocido. Además, en cada ciudad, para realizar el servicio solicitado se emplea un tiempo que denominaremos tiempo de proceso y lo denotaremos como P_i.

Por lo tanto, a la hora de escoger el arco (i, j), éste debe satisfacer la siguiente condición:

$$a_i + P_i + t_{ij} \leq b_j, \quad \forall i, j \in N$$

Si queremos construir el modelo matemático, solo nos haría falta añadir dos restricciones para que se cumplan las ventanas de tiempo:

$$a_i \leq T_i^k \leq b_i \qquad \forall i \in N, \forall k = 1, ..., 5 \tag{2.11}$$

$$x_{ij}^k (T_i^k - T_j^k + P_i + t_{ij}) \leq 0 \qquad \forall i, j \in N, \forall k = 1, ..., 5 \tag{2.12}$$

Donde T_i^k es una nueva variable no negativa que indica el tiempo de llegada a la empresa i el dia k.

La restricción (2.12) elimina los subciclos, por lo tanto no nos haría falta escribir en este modelo la restricción (2.7). El problema es que esta restricción no es lineal, para ello

utilizamos una constante M lo suficientemente grande y nos quedaría:

$$T_j^k \leq T_i^k + P_i + t_{ij} - M_{ij}(1 - x_{ij}^k) \qquad \forall i, j \in N, \forall k = 1, ..., 5 \tag{2.13}$$

2.2. Resolutores

Existen diversos métodos para la resolución de los diferentes problemas de programación matemática, por ello, realizaremos un pequeño estudio para ver que "software" utilizaremos teniendo en cuenta las ventajas e inconvenientes de cada uno de ellos.

A continuación se muestran las diferentes posibilidades que se plantearon a la hora de elegir el "software" que se empleará para el desarrollo de este proyecto, dividiéndolos según su licencia en resolutores comerciales y resolutores no comerciales o de código libre. Para finalmente realizar un estudio en profundidad de Microsoft Solver Fundation, Gusek, SCIP y LPSolve IDE.

2.2.1. Resolutores de pago

Estos resolutores se caracterizan por poseer entornos gráficos de gran utilidad y fáciles de aprender con una gran cantidad de heurísticos que ayudan a resolver de manera más rápida los problemas. La principal desventaja es el coste para adquirirlos.

Algunos ejemplos de este tipo de resolutores son el XPRESS, CPLEX, GUROBI y Microsoft Solver Foundation (Solver de Excel).

XPRESS: Uno de los más importantes programas de optimización en general, y de resolución de problemas de programación lineal entera en particular. Consta de entorno gráfico y además tiene su propio lenguaje de programación, muy potente, llamado Mosel. Ofrece solucionadores potentes y versátiles que ayudan a modelar con mayor precisión y resuelven problemas del mundo real de gran complejidad. El año pasado, se utilizó este programa para modelar el problema de rutas.

Xpress proporciona una amplia gama de sofisticados algoritmos de optimización, robustos para la solución de problemas lineales de gran escala y problemas enteros mixtos, así como problemas no lineales.

CPLEX: Su competidor más cercano, en cuanto a la capacidad de resolver problemas muy complejos con gran cantidad de variables y restricciones, que, al igual que Xpress, tiene IDE,entorno de desarrollo integrado, y además su propio lenguaje de programación (OPL). Es considerado uno de los mejores programas comerciales y académicos de programación lineal y lineal entera.

Proporciona la potencia necesaria para resolver grandes problemas de optimización en el mundo real, así como la velocidad requerida para las aplicaciones de optimización de decisiones interactivas de hoy. Ofrece a los desarrolladores una variedad de

formas de interactuar con él durante el desarrollo y despliegue de sus aplicaciones. Con C, C ++, Java, Visual Basic, Python y Fortran, los desarrolladores pueden integrar algoritmos CPLEX poderosos y modelos OPL dentro de sus programas de aplicación.

GUROBI: Su flexibilidad y comodidad lo hacen ideal para la creación rápida de prototipos y el desarrollo del modelo. Como desventaja con respecto a XPRESS y CPLEX, éste no incluye un entorno de desarrollo integrado. Su código fue construido para explotar plenamente el paralelismo. No es un código secuencial que se haya paralelizado, sino un código fundamentalmente paralelo al que también se puede optar por ejecutar secuencialmente.

También posee nuevas clases de cortes disponibles únicamente en Gurobi con soporte para todo tipo de problemas y con buenos tiempos de resolución

Microsoft Solver Foundation: [5] Ofrece la ventaja del fácil uso de herramientas y una profunda integración con los sistemas de trabajo como Microsoft Office.

Solver Foundation sirve para simular problemas, encontrar el máximo o mínimo valor óptimo de las funciones y modelar sistemas complejos mediante el uso de variables de decisión y restricciones. Permite utilizar la programación lineal, programación no lineal, programación cuadrática y la programación lineal entera mixta entre otras.

Solver no posee un lenguaje propio de programación y al ser un complemento de Excel aparece dentro de él, con lo que no tiene tampoco un IDE específico, sino que utiliza la interfaz de Excel. Es idóneo para problemas no muy complejos.

Microsoft tiene una asociación con Gurobi para proporcionar su solucionador de MIP en Fundation Solver, cuya licencia incluye automáticamente una licencia Gurobi.

2.2.2. Resolutores Libres

Los resolutores de código libre son gratuitos y se puede acceder a su código. No tienen ningún tipo de limitaciones (como uso comercial, o la cantidad de variables o restricciones que pueden resolver), que sí tienen las versiones gratuitas de los programas comerciales. Estos resolutores están menos desarrollados que los comerciales, lo que provoca mayor tiempo de ejecución y menos versatilidad a la hora de mostrar los resultados.

Algunos ejemplos de resolutores de código libre son SoPlex, R, SCIP, Gusek y LPSolve.

SoPlex: Es un resolutor de programación lineal que se basa en el algoritmo simplex revisado, implementado en C++. Agiliza el desarrollo y despliegue de modelos de optimización, combinando solucionadores avanzados con un lenguaje de modelo perfectamente integrado.

R: Un entorno de "software" libre para computación y gráficos estadísticos, con un manejo eficaz de los datos y con un lenguaje de programación bien desarrollado, simple y eficaz.

SCIP con Soplex 2.0: Uno de los resolutores no comerciales más rápidos, como vemos en la Figura 1.1, para la programación entera mixta y la programación no lineal entera mixta. Éste nos permite un control total del proceso de solución y el acceso a la información detallada del solucionador. Se basa en el procedimiento Ramificación y Acotación, resolviendo programas enteros de restricción (CIP). SCIP con Soplex 2.0 soporta variables continuas, binarias, enteras, semi-continuas, semi-enteras, variables indicadoras. SCIP con Soplex 2.0 es configurable para poner diferentes motores que resuelven los LP (Problemas Lineares).

GUSEK: [3] Es un interfaz libre para windows que utiliza un motor de optimización gratuito para resolver modelos matemáticos lineales. Este motor es GLPK (GNU Linear Programming Kit), y es una alternativa a XPRESS. Tiene un lenguaje de modelado diferente, denominado GMPL.

El GNU Linear Programming Kit se trata de un "software" para la resolución de problemas a gran escala de programación lineal (LP), programación entera mixta (MIP), y otros problemas relacionados.

Los problemas pueden ser modelados en el idioma GNU MathProg que comparte muchas partes de la sintaxis con AMPL y resueltos con GLPSOL, un solucionador independiente.

LPSolve IDE: Resuelve problemas de programación lineal entera mixta. Se puede acceder a toda la funcionalidad de LPSolve a través de una aplicación fácil de usar. Todo se controla de forma gráfica y con el ratón. Este IDE trabaja con problemas en todos los formatos soportados por LPSolve y transforma el modelo desde cualquier formato soportado a cualquier otro formato soportado. Tiene extensiones altamente configurables para integrar diferentes formatos de archivo.

Simplemente proporcionas tu modelo al programa y él dará los resultados.

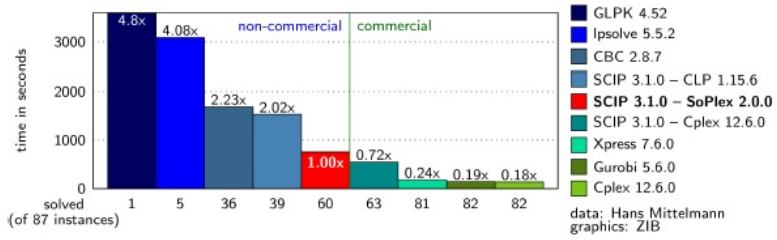


Figura 2.1: Comparación de diferentes resolutores

Como vemos, en la figura 2.1 se muestra un ejemplo comparativo de varios resolutores según su tiempo de ejecución. Podemos ver que la opción de usar el SCIP con Soplex 2.0 es la más rápida dentro de los resolutores libres. Por otra parte, su uso es más complejo que el de LPSolve y Excel. Sin embargo, LPSolve tiene un tiempo de ejecución bastante alto comparado con el resto.

En un primer instante se planteó la opción de utilizar Microsoft Solver Foundation ya que Solver se utiliza cuando queremos encontrar los valores de determinadas celdas de una hoja de cálculo que optimicen un determinado objetivo. También al ser un complemento de Excel, a pesar de ser comercial, es un "software" que la mayoría de las empresas dispone. En este caso, Biocontrol Canarias S.L. dispone de la última versión de Excel y todos sus datos están recogidos en hojas de Excel.

Intentando modelar con Microsoft Solver Foundation, nos encontramos con varios problemas que poco se fueron solucionando, pero en el momento de introducir tres índices a nuestras variables, parece que el programa no terminaba de responder de forma adecuada. Tras varios intentos con ejemplos mas pequeños, no encontramos sentido a la solución que éste nos proporcionaba.

Pensando en otro resolutor cómodo para la empresa, planteamos utilizar Gusek, ya que Gusek permite importar y exportar datos de Excel. Por lo tanto es más accesible a la hora de introducir los datos y a la hora de ver los resultados.

En cualquier caso, en el apéndice B.1 se encuentra la última versión del código implementado con Microsoft Solver Foundation, con las restricciones citadas en la sección 3.2.3. El entorno de este resolutor se puede ver en la figura 2.3.

En la figura 2.2 podemos ver el entorno Gusek que se utilizará para modelar el problema de rutas de Biocontrol Canarias S.L.

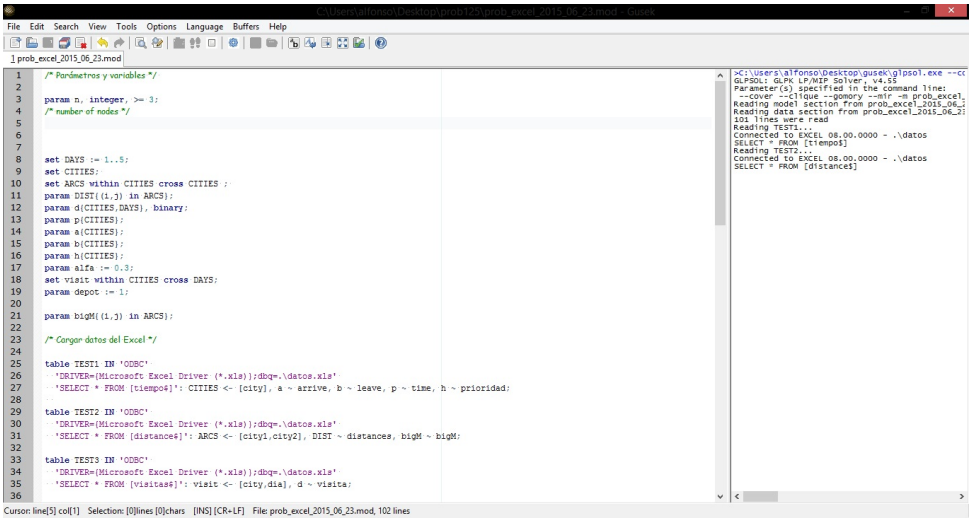


Figura 2.2: Gusek

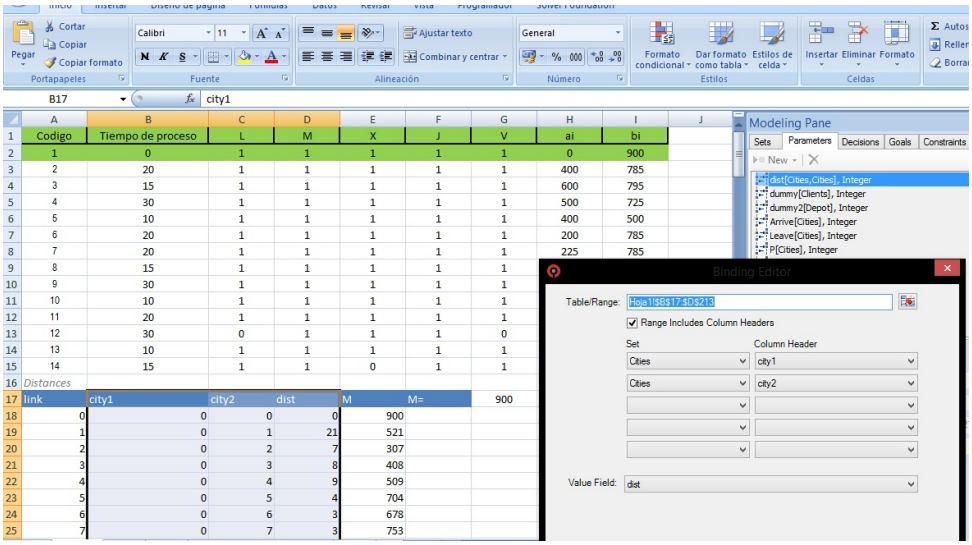


Figura 2.3: Microsoft Solver Foundation

Capítulo 3

Formulación del problema

En este capítulo procederemos finalmente al modelado del problema de rutas aplicado a Biocontrol Canarias S.L. Empezaremos recordando las características del problema, seguidamente se explica como se obtienen los datos y después de definir los parámetros y variables, presentamos el modelo matemático final.

3.1. Características

Para resolver el problema empezaremos considerando una función objetivo en la que tenemos que minimizar el tiempo que se tarda en recoger las muestras y añadiremos a nuestra función objetivo un beneficio constante que obtenemos al visitar a cada cliente. Recordemos las características de nuestro problema citadas en la sección 1.2 :

- Hay 125 clientes, un vehículo y un depósito (Biocontrol Canarias S.L.).
- Se realiza a lo sumo una ruta cada día.
- Las distancias y los tiempos de desplazamiento son conocidos.
- Hay que respetar unas ventanas de tiempo según el cliente, es decir, dentro de un horario predeterminado.
- Las visitas solo pueden realizarse durante el horario de Biocontrol Canarias S.L., salvo cuando se visite en otras islas.
- Cada cliente tiene unas preferencias de visita, desde semanal hasta anual.
- Algunas empresas pueden visitarse cualquier día pero otras tienen que ser visitadas un día en concreto.
- Dependiendo del tipo de trabajo, se necesitará un tiempo de proceso diferente.

3.2. Modelo con variables arco

En esta sección se presenta el modelo que hemos formulado con las características citadas anteriormente. Describiremos como se han obtenido los datos, definiremos los parámetros y variables del modelo y finalmente se procede con el modelo matemático. Este modelo fue utilizado por Rodríguez-García [10].

3.2.1. Datos

El programa recoge todos los datos de un archivo de Excel con varias hojas.

Lo primero que necesitamos conocer es la distancia que hay entre todos los clientes, para ello creamos una matriz distancia de dimensión 125x125. Esta matriz la transformamos en formato columnas con fórmulas de Excel para poder leer los datos desde el Gusek.

En el resto de columnas anotamos las preferencias de cada cliente (ventanas de tiempo, tiempo de proceso, días de visita, periodicidad...).

Tenemos también la columna prioridades, esta columna solo toma valores cero, uno, dos y tres. Estos valores los iremos cambiando con diferentes pruebas para ir obteniendo diferentes resultados computacionales en el capítulo 4. Este dato corresponde con el parámetro h explicado en la siguiente sección 3.2.2.

3.2.2. Parámetros y variables

En esta sección definiremos los parámetros y las variables de nuestro modelo matemático.

Parámetros

Tenemos un grafo $G = (N, A)$, en el que $N = \{1, 2, ..., n\}$ es el conjunto de vértices (empresas), siendo 1 Biocontrol Canarias S.L. y A el conjunto de conexiones entre todos los clientes, es decir, el conjunto de arcos (i, j) donde i es el origen y j el destino de los dos clientes que conecta.

Otras notaciones que usaremos son:

- $t_{i,j}$ lo denotaremos como el tiempo que se tarda en ir de i a j.
- P_i será el de proceso en el cliente i.
- a_i lo denotaremos por la hora mínima a la que se puede visitar la empresa i.
- De la misma forma, b_i la hora límite de visitar la empresa i.
- El conjunto de noodos j tal que $(i, j) \in A$ lo denotaremos por $\delta^+(i)$. Por lo tanto, $\delta^-(i)$ será el conjunto de vértices j tal que $(j, i) \in A$.
- Una de las características del problema era que cada empresa puede tener preferencias en cuanto al día de visita, para ello creamos el parámetro d_i^k tal que:

$$d_i^k = \begin{cases} 1, & \text{si el día } k \text{ se puede visitar la empresa } i \\ 0, & \text{en otro caso} \end{cases}$$

- Para que una restricción del modelo sea lineal crearemos un parámetro M para cada arco (i, j) definido como:

$$M_{ij} = \text{ máx } \{b_i + P_i + t_{i,j} - a_j, 0\}$$

- Para conocer la prioridad de visita de cada cliente y poder así generar un beneficio artificial por visitar dicha empresa, crearemos un parámetro g_i que definiremos de la siguiente manera:

$$g_i = \alpha(3 - h_i)(t_{1,i} + t_{i,1}), \ \ \forall i \in N$$

  Con h_i definida tal que:

$$h_i = \begin{cases} 0, & \text{si hay que visitar la empresa } i \text{ a lo largo de la semana} \\ 1, & \text{si se puede visitar la empresa } i \text{ con un beneficio grande} \\ 2, & \text{si se puede visitar la empresa } i \text{ con un beneficio no tan grande} \\ 3, & \text{no hay que visitar la empresa } i \end{cases}$$

  y $\alpha \in (0, 0.5)$

Con esto conseguimos que, si para la empresa i tenemos que $h_i = 0$ el beneficio que genera visitarla es proporcional al triple de la suma del tiempo que supondría ir desde Biocontrol Canarias a la empresa i y volver; si $h_i = 1$, el beneficio es proporcional al doble de la suma del tiempo que supondría ir desde Biocontrol Canarias a la empresa i y volver; si $h_i = 2$, el beneficio es proporcional a la suma del tiempo de ir desde Biocontrol Canarias a la empresa i y volver; y por último, si no hay que visitar a la empresa, $h_i = 3$, no se obtiene ningún beneficio.

De esta forma, igual interesa visitar clientes lejanos del depósito (Biocontrol Canarias S.L.) que no estén en la ruta, ya que el vehículo pasa por una empresa en ruta cercana a ésta.

Variables

El modelo estará formado por tres variables, las variables x e y de tipo binario y la variable T no negativa, definidas tal que:

$$x_{ij}^k = \begin{cases} 1, & \text{si el arco } (i, j) \text{ está en la ruta el día } k \\ 0, & \text{en otro caso} \end{cases}$$

$$y_i^k = \begin{cases} 1, & \text{si la empresa } i \text{ se le visita el día } k \\ 0, & \text{en otro caso} \end{cases}$$

$T_i^k =$ tiempo de llegada a la empresa i el día k

Una vez definidos todos lo parámetros y las variables ya estamos en condiciones de definir nuestro modelo matemático.

3.2.3. Modelo matemático

El objetivo es minimizar el tiempo empleado cada semana, restándole a la función objetivo los beneficios que genera visitar una empresa según su prioridad.

$$\text{mín} \sum_{k=1}^{5} \sum_{i,j \in A} t_{ij} \cdot x_{ij}^k - \sum_{k=1}^{5} \sum_{i \in N} g_i \cdot y_i^k \tag{3.1}$$

sujeto a:

$$\sum_{j \in \delta^+(i)} x_{ij}^k = \sum_{j \in \delta^-(i)} x_{ji}^k = y_i^k \qquad \forall i, j \in N, \forall k = 1, ..., 5 \tag{3.2}$$

$$a_i \leq T_i^k \leq b_i \qquad \forall i \in N, \forall k = 1, ..., 5 \tag{3.3}$$

$$T_j^k \leq T_i^k + P_i + t_{ij} - M_{ij}(1 - x_{ij}^k) \qquad \forall i, j \in N, \forall k = 1, ..., 5 \tag{3.4}$$

$$y_i^k \leq d_i^k \qquad \forall i \in N, \forall k = 1, ..., 5 \tag{3.5}$$

$$\sum_{k=1}^{5} y_i^k = 1 \qquad \forall i \in N, h_i = 0 \tag{3.6}$$

$$\sum_{k=1}^{5} y_i^k \leq 1 \qquad \forall i \in N, h_i \in \{1, 2\} \tag{3.7}$$

$$y_i^k = 0 \qquad \forall i \in N, h_i = 3, \forall k = 1, ..., 5 \tag{3.8}$$

$$y_1^k \geq y_i^k \qquad \forall i \in N \setminus \{1\}, \forall k = 1, ..., 5 \tag{3.9}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i, j \in N, \forall k = 1, ..., 5 \tag{3.10}$$

$$y_i^k \in \{0, 1\} \qquad \forall i \in N, \forall k = 1, ..., 5 \tag{3.11}$$

Restricciones

La restricción (3.2) sirve para asegurar que solo entrará y se saldrá una sola vez de una empresa elegida en la ruta.

La restricción (3.3) impone que se visite a cada cliente dentro de su ventana de tiempo.

La restricción (3.4) nos asegura que la siguiente empresa en visitar debe hacerse en un tiempo superior a la suma de todos los tiempos de la visita anterior. Esta restricción también nos evita los subciclos.

La restricción (3.5) establece las preferencias en cuanto al día de la semana.

Las restricciones (3.6), (3.7) y (3.8) hacen referencia a la prioridad de cada empresa. Como dijimos antes, la empresa será visitada si tiene prioridad cero, no será visitada si tiene prioridad tres, y las empresas con prioridad uno o dos pueden o no ser visitadas.

La restricción (3.9) obliga a pasar por el depósito la ruta realizada el dia k.

Las dos últimas restricciones, (3.10), (3.11), son las llamadas restricciones de binariedad.

El código del modelo se encuentra en la sección A.1.

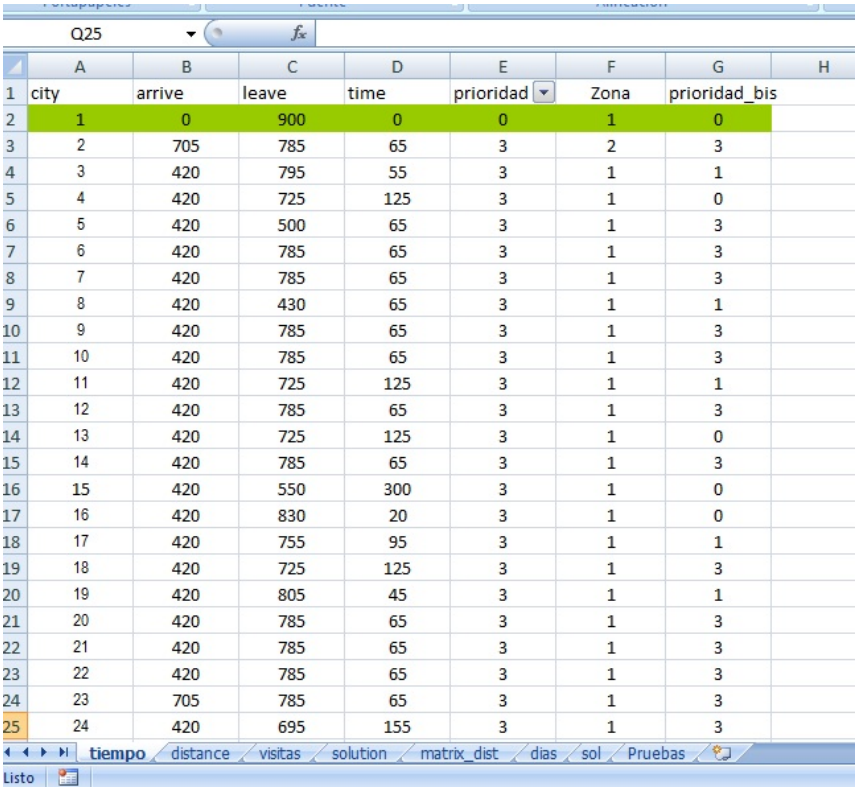
3.2.4. Lectura de datos

Una de las ventajas de Gusek es que, como ya comentamos, puede leer los datos de una hoja de Excel.

Esto lo consigue mediante el driver ODBC. La forma de leer los datos se puede ver en el código que es bastante sencilla. Primero se nombra al fichero donde están los datos y seguidamente elegimos la hoja de la cual queremos extraer los valores, añadiendo alguna restricción si es necesario.

A la hora de darle valores a los conjuntos, primero se escribe el nombre del conjunto tal y como lo hemos definido, seguidamente utilizamos el símbolo $<-$ y, por último, entre corchetes, el encabezado de la columna de Excel donde están los valores.
Si queremos leer los valores de los parámetros, definidos previamente, simplemente basta con nombrar el parámetro y relacionarlo mediante el símbolo $\sim$ con el encabezado de la columna de Excel que nos interese.

Podemos ver un ejemplo de estas tablas en la figura 3.1.



	A	B	C	D	E	F	G
1	city	arrive	leave	time	prioridad	Zona	prioridad_bis
2	1	0	900	0	0	1	0
3	2	705	785	65	3	2	3
4	3	420	795	55	3	1	1
5	4	420	725	125	3	1	0
6	5	420	500	65	3	1	3
7	6	420	785	65	3	1	3
8	7	420	785	65	3	1	3
9	8	420	430	65	3	1	1
10	9	420	785	65	3	1	3
11	10	420	785	65	3	1	3
12	11	420	725	125	3	1	1
13	12	420	785	65	3	1	3
14	13	420	725	125	3	1	0
15	14	420	785	65	3	1	3
16	15	420	550	300	3	1	0
17	16	420	830	20	3	1	0
18	17	420	755	95	3	1	1
19	18	420	725	125	3	1	3
20	19	420	805	45	3	1	1
21	20	420	785	65	3	1	3
22	21	420	785	65	3	1	3
23	22	420	785	65	3	1	3
24	23	705	785	65	3	1	3
25	24	420	695	155	3	1	3

Figura 3.1: Tabla de datos

3.3. Modelo con variables rutas

Debido a las grandes dimensiones del modelo anterior, el "software" que utilizamos, Gusek, tiene un tiempo computacional muy alto. Es por ello que planteamos un nuevo modelo con el fin de reducir este tiempo computacional y obtener resultados de manera mas rápida.

Para este modelo suponemos que Biocontrol Canarias visita como máximo a cuatro clientes por cada ruta que realiza. El objetivo ahora es hallar este número de rutas y el costo de cada una de ellas.

Sería interesante en un futuro plantear este mismo modelo con cinco clientes por ruta, este año por falta de tiempo no se ha podido realizar.

Estas rutas son calculadas con Visual Basic para Aplicaciones (VBA) [6], el código se puede ver en el Apéndice B.2.

La forma de obtener estas rutas está explicado detalladamente en la sección 3.3.4.

Una vez que conocemos las rutas posibles, podemos plantear nuestro modelo y definir

las variables y los parámetros.

3.3.1. Parámetros y variables

Parámetros

Empezaremos definiendo $R = \{1, 2, ..., r\}$ como el conjunto de rutas, $K = \{1, ..., 5\}$ días de la semana y $N = \{1, 2, ..., n\}$ el conjunto de clientes.

- C_r representa el costo de la ruta r.
- Crearemos una matriz m formada por unos y ceros para saber que cliente hay en cada ruta.

$$m_{ri} = \begin{cases} 1, & \text{si la ruta } r \text{ pasa por el cliente } i \\ 0, & \text{en otro caso} \end{cases}$$

- Para saber que días se puede realizar una determinada ruta r, creamos el parámetro d. Una ruta no se puede hacer un cierto día si hay algún clientes que no pueda ser visitado ese día. Por lo tanto:

$$d_r^k = \begin{cases} 1, & \text{si la ruta } r \text{ se puede hacer el día } k \\ 0, & \text{en otro caso} \end{cases}$$

- Igual que en el modelo 3.2.3 generamos un beneficio artificial g_i de la siguiente manera:

$$g_i = \alpha(3 - h_i)(t_{1,i} + t_{i,1}), \ \ \forall i \in N$$

Con h_i :

$$h_i = \begin{cases} 0, & \text{si hay que visitar la empresa } i \text{ a lo largo de la semana} \\ 1, & \text{si se puede visitar la empresa } i \text{ con un beneficio grande} \\ 2, & \text{si se puede visitar la empresa } i \text{ con un beneficio no tan grande} \\ 3, & \text{no hay que visitar la empresa } i \end{cases}$$

y $\alpha \in (0, 0.5)$

Variables

El modelo estará formado por una única variable x de tipo binario, tal que:

$$x_r^k = \begin{cases} 1, & \text{si la ruta } r \text{ se hace el día } k \\ 0, & \text{en otro caso} \end{cases}$$

Estamos en condiciones de definir el modelo matemático.

3.3.2. Modelo matemático

El objetivo es minimizar el número de rutas con el menor costo posible.

$$\text{mín} \sum_{r \in R} \sum_{k=1}^{5} x_r^k \cdot C_r - \sum_{k=1}^{5} \sum_{i \in N} \sum_{r \in R} g_i \cdot m_{ri} \cdot x_r^k \tag{3.12}$$

sujeto a:

$$\sum_{k=1}^{5} \sum_{r \in R} m_{ri} \cdot x_r^k = 1 \qquad \forall i \in N, h_i = 0 \tag{3.13}$$

$$\sum_{k=1}^{5} \sum_{r \in R} m_{ri} \cdot x_r^k \leq 1 \qquad \forall i \in N, h_i \in \{1, 2\} \tag{3.14}$$

$$\sum_{k=1}^{5} \sum_{r \in R} m_{ri} \cdot x_r^k = 0 \qquad \forall i \in N, h_i = 3 \tag{3.15}$$

$$x_r^k \leq d_r^k \qquad \forall r \in R, \forall k = 1, ..., 5 \tag{3.16}$$

$$\sum_{r \in R} x_r^k \leq 1 \qquad \forall k = 1, ..., 5 \tag{3.17}$$

Restricciones

Las restricciones (3.13), (3.14) y (3.15), igual que en el modelo anterior, hacen referencia a la prioridad de cada empresa y nos obliga a que cada cliente esté a lo sumo en una ruta. Así cuando $h_i = 0$ el cliente tiene que ser visitado, cuando $h_i = 1, 2$ el cliente puede ser visitado o no y cuando $h_i = 3$ el cliente no tiene que ser visitado.

La restricción (3.16) establece las preferencias en cuanto al día de la semana.

La restricción (3.17) nos impone que solo se puede realizar una ruta al día.

El código del modelo de rutas se puede ver en el apéndice A.2

3.3.3. Lectura de datos

Almacenar todas estas rutas en una hoja de Excel puede suponer un problema. Esto es debido a que genera gran cantidad de datos y se supera el límite de filas de Excel. Por ello, para este modelo, Gusek carga los datos desde fichero.

Como vemos en el código, la lectura se realiza de forma muy parecida al modelo anterior. La diferencia es que ahora son archivos con formato CSV.

En la figura 3.2 vemos un ejemplo de un fichero, donde se muestran los costos de cada ruta para un caso particular.

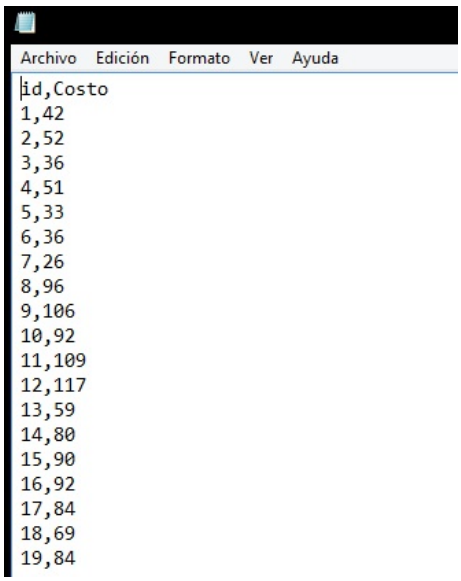


Figura 3.2: Fichero

3.3.4. Generación de rutas

En esta sección se explica la forma de obtener las diferentes rutas del segundo modelo, cuyo código podemos ver en el apéndice B.2.

Para ello se utilizó Visual Basic para Aplicaciones. VBA viene integrado como uno de los complementos de Microsoft Excel. Los códigos creados no se pueden compilar separadamente de la hoja de cálculo en que fue creada. Se crea una macro y, en este caso, desde un botón que hemos añadido a la hoja de cálculo se compila.

Lo primero que se hace es cargar los datos (distancias, ventanas de tiempo, periodicidades...) de las propias hojas de Excel.

Seguidamente se buscan las variables de los clientes según la zona en la que se encuentren y se comprueba que estas variables son factibles en los días, es decir, hay uno o mas días que se pueden visitar todos los clientes de esa ruta.

Ahora que se sabe que son factibles, se calcula el costo de esta ruta, con todas las combinaciones posibles. Esto es, para 4 clientes tenemos 4! posibilidades, para 3 clientes, 3! y para 2 clientes 2 posibilidades. Finalmente nos quedamos con la que menor coste tenga y la almacenamos.

En un principio se planteó borrar ciertas rutas ya que se obtenían demasiadas. Para ello se buscaban los mejores costos en relación con el número de clientes y se marcaban las que podían ser borradas. Finalmente, al ver la velocidad con la que se obtenían resultados con este modelo, se decidió dejar todas las rutas. En el código aparece comentada la llamada a la función que elimina las rutas.

3.4. Modelo con variables ruta con vista a un año

Aparte del modelo anterior, se ha planteado un nuevo caso donde se planeen todas las visitas a los clientes durante un año completo. Para este modelo no se ha creado un código, lo cual podría ser interesante para futuros trabajos.

Lo primero que hay que tener en cuenta es la periodicidad de visita de cada cliente. Consideremos que un año tiene 48 semanas (eliminamos Semana Santa, Navidad,etc.).

Definimos f_i como la frecuencia de visita, es decir, f_i representa cada cuantas semanas debemos visitar al cliente i.

La periodicidad de los clientes de Biocontrol puede ser:

- Semanal: $f_i = 1$, 48 visitas al año.
- Quincenal: $f_i = 2$, 24 visitas al año.
- Mensual: $f_i = 4$, 12 visitas al año.
- Bimestral: $f_i = 8$, 6 visitas al año.
- Trimestral: $f_i = 12$, 4 visitas al año.
- Cuatrimestral: $f_i = 16$, 3 visitas al año.
- Semestral: $f_i = 24$, 2 visitas al año.
- Anual: $f_i = 48$, 1 visitas al año.

El modelo que vamos a plantear es un modelo estricto. Esto quiere decir que si un cliente tiene periodicidad anual, la próxima visita debe ser justo 48 semanas después de haberlo visitado. Lo mismo ocurriría con un cliente con periodicidad mensual, si se visita la primera semana de un determinado mes, debe ser visitado justo 4 semanas después.

Sería interesante en un futuro plantear un modelo que admita un rango de visita según la periodicidad. Es decir, que un cliente con periodicidad anual que se visitó en marzo, pueda ser visitado en febrero, abril... del año que viene. En el caso de periodicidad mensual, podría ser visitado la segunda semana del siguiente mes, si fuera mas conveniente para la empresa.

Biocontrol Canarias S.L. dispone de un seguimiento continuo de las visitas que va realizando. Por lo tanto, llamaremos u_i al número de semanas que hace desde que se visitó al cliente i. Conociendo este dato, podemos saber en qué semana se debe realizar de nuevo la próxima visita.

3.4.1. Parámetros y variables

Los parámetros y variables de este modelo son muy parecidos al descrito en la sección 3.3.1.

Parámetros

De la misma manera, $R = \{1, 2, ..., r\}$ es el conjunto de rutas, $K = \{1, ..., 5\}$ días de la semana y $N = \{1, 2, ..., n\}$ el conjunto de clientes. Nos hace falta un nuevo conjunto $w = \{1, 2, ..., 48\}$, número de semanas.

- C_r representa el costo de la ruta r.
- f_i representa la frecuencia de visita.
- u_i número de semanas que hace desde la última visita.
- Tenemos el parámetro m:

$$m_{ri} = \begin{cases} 1, & \text{si la ruta } r \text{ pasa por el cliente } i \\ 0, & \text{en otro caso} \end{cases}$$

- Parámetro d para determinar los días posibles, de la misma manera que en el anterior modelo:

$$d_r^k = \begin{cases} 1, & \text{si la ruta } r \text{ se puede hacer el día } k \\ 0, & \text{en otro caso} \end{cases}$$

Se podría considerar de la misma manera un beneficio artificial g_i. En este caso no lo incluiremos en nuestro modelo.

Variables

El modelo volvería a estar formado por una única variable x de tipo binario, esta vez:

$$x_r^{kw} = \begin{cases} 1, & \text{si la ruta } r \text{ se realiza el día } k \text{ de la semana } w \\ 0, & \text{en otro caso} \end{cases}$$

3.4.2. Modelo matemático

El objetivo es minimizar el costo total de las rutas escogidas.

$$\text{mín} \sum_{r \in R} \sum_{k=1}^{5} \sum_{w=1}^{48} x_r^{kw} \cdot C_r \tag{3.18}$$

sujeto a:

$$\sum_{k=1}^{5} \sum_{r \in R} m_{ri} x_i^{kw} = \begin{cases} 1, & \text{si } (u_i + w) \bmod f_i = 0 \\ 0, & \text{en otro caso} \end{cases} \qquad \forall i \in N, \forall w = 1, ..., 48 \tag{3.19}$$

$$\sum_{r \in R} x_r^{kw} \leq 1 \qquad \forall k, w \tag{3.20}$$

$$x_r^{kw} \leq d_r^k \qquad \forall r \in R, \forall k, w \tag{3.21}$$

Restricciones

La restricción (3.19) nos determina en que semana debe ser visitado el cliente. Por ejemplo, un cliente con periodicidad mensual, $f_i = 4$, que se visitó hace dos semanas, $u_i = 2$, si estamos en la semana $w = 6$, entonces $(2 + 6) \bmod 4 = 0$, por lo tanto tocaría visitarlo.

La restricción (3.20) nos indica que solo se puede realizar una ruta por día.

La restricción (3.21) establece las preferencias en cuanto al día de la semana.

Capítulo 4

Resultados computacionales

En este capítulo se presentan en tablas los resultados obtenidos con ambos modelos. Para ello se ha ido compilando diferentes conjuntos de clientes, cambiando las prioridades, el valor h. Recordar que este beneficio es algo ficticio que se introduce para visitar los clientes próximos que no haya que visitar esa semana pero que se podrían visitar. Estos depende del parámetro alfa. Haremos dos tablas para ver las diferencias.

Se compararán también los tiempos de computación y los costes entre ambos modelos. En el segundo modelo no se contempla el tiempo que tarda Visual Basic en obtener las rutas ya que es bastante rápido.

Las columnas de la tabla representan:

- **S**: La semilla, el conjunto de clientes.
- $\#h_i = 0$: El número de clientes con prioridad cero. Recordemos que esto significa que estamos obligados a visitar ese cliente.
- $\#h_i = 1$: El número de clientes con prioridad uno. Es decir, si visitamos a la empresa, obtenemos un beneficio grande.
- **T**: Tiempo computacional. Medido en segundos.
- **C**: Coste que supone visitar a los clientes.
- **g**: Beneficio obtenido por visitar a determinados clientes.
- **Coste Total**: El coste menos el beneficio obtenido.

A la hora de obtener los resultados, como el primer modelo (modelo con arcos) es mucho más lento, se ha puesto tiempo límite de una hora. Cuando llega a este tiempo, no obtenemos el valor del coste y el benificio, Gusek solo nos indica la solución óptima que ha encontrado en ese momento.

Clientes			**Modelo arcos**				**Modelo rutas**			
S	$\#h_i = 0$	$\#h_i = 1$	**T**	**C**	**g**	**Coste Total**	**T**	**C**	**g**	**Coste Total**
A	4	0	4.8	88	103	-15	0.7	93	103	-10
B			4.3	170	184	-14	0.4	190	184	6
C			4.6	154	251	-97	0.5	163	251	-88
A	4	4	13.9	186	342	-156	0.4	174	294	-120
B			22.7	191	320	-128	0.5	177	264	-87
C			616.3	157	363	-206	0.9	291	470	-179
A	4	8	23.9	231	380	-149	0.5	183	300	-117
B			186.8	235	334	-99	0.6	321	354	-33
C			78	292	434	-142	2	325	410	-85
A	4	12	3600	-	-	-202.2	3.9	231	410	-179
B			3600	-	-	-157.9	1.8	257	408	-151
C			3600	-	-	-256.8	3.1	177	353	-176
A	8	0	13.5	253	317	-64	0.7	281	317	-36
B			21.2	272	511	-239	0.5	336	511	-175
C			489.8	694	878	-184	0.5	773	878	-105
A	8	4	302.6	174	354	-180	0.9	195	354	-159
B			433.5	228	358	-130	0.8	234	358	-124
C			3600	-	-	-194.3	0.8	422	540	-118
A	8	8	3600	-	-	-161.1	2.1	266	425	-159
B			3600	-	-	-82.3	2.7	159	240	-81
C			3600	-	-	-159.7	2.2	2395	1140	1255
A	12	0	977.3	192	279	-87	0.9	197	279	-82
B			3600	-	-	-252.5	0.7	391	580	-189
C			3549.6	266	425	-159	1	274	425	-151
A	12	4	3600	-	-	-76.3	3.7	137	201	-64
B			3600	-	-	-349.4	1.8	383	674	-291
C			3600	-	-	-274	1.6	384	639	-255
A	16	0	3600	-	-	-90.2	4.5	132	214	-82
B			3600	-	-	-357	1.2	457	846	-389
C			3600	-	-	-597.3	1.5	484	1005	-521

Cuadro 4.1: Resultados computacionales con $\alpha = 0{,}3$

En la tabla 4.1 se pueden ver los resultados obtenidos. En este caso se ha usado $\alpha = 0{,}3$. Se puede apreciar una gran diferencia entre los tiempos computacionales de uno y otro modelo, a medida que van aumentando el número de clientes. De la misma manera, en el primer modelo, el coste total es más bajo que en el segundo modelo, esto es debido a que con el modelo de arcos tenemos disponibles todas las rutas posibles y con el segundo modelo solo rutas de cuatro clientes máximo. Por lo tanto el coste de la ruta es menor

con el primer modelo. En los casos en que es mayor, es gracias a que obtenemos un mayor beneficio al visitar a más clientes.

A continuación se muestran otros resultados para los mismos conjuntos de clientes, esta vez utilizando $\alpha = 0{,}1$

Clientes			**Modelo arcos**				**Modelo rutas**			
S	$\#h_i = 0$	$\#h_i = 1$	**T**	**C**	**g**	**Coste Total**	**T**	**C**	**g**	**Coste Total**
A	4	0	4.9	88	34	54	0.5	93	34	59
B			4.5	170	61	109	0.5	190	61	129
C			4.6	154	84	70	0.5	163	84	79
A	4	4	7.4	186	114	72	0.6	174	98	76
B			7.3	163	93	70	0.5	152	77	75
C			17.5	157	121	36	0.5	235	121	114
A	4	8	6.2	143	83	60	0.7	167	89	78
B			6.9	204	95	109	0.9	267	88	179
C			57	199	79	120	0.6	218	79	139
A	4	12	3600	-	-	80.6	2.1	204	124	80
B			81.7	129	66	64	1.9	118	51	67
C			175.2	115	117	-2	1.7	89	73	16
A	8	0	13.1	253	106	147	0.5	281	106	175
B			26.1	272	170	102	0.6	336	170	166
C			290.6	694	293	401	0.6	773	293	480
A	8	4	140.3	174	118	56	1.1	195	118	77
B			70.6	198	100	98	0.8	198	100	98
C			3600	-	-	126.1	0.9	342	142	200
A	8	8	3600	-	-	90.3	2.1	211	117	94
B			3600	-	-	58.7	1.8	123	62	61
C			1383.9	843	302	541	1.5	2248	302	1946
A	12	0	672.5	192	93	99	1.2	197	93	104
B			3600	-	-	134.5	1.1	391	194	197
C			3515.2	266	142	124	1.2	274	142	132
A	12	4	3600	-	-	56.3	4.5	111	54	57
B			3600	-	-	86.4	1.7	370	220	150
C			3600	-	-	91	1.9	273	162	111
A	16	0	3600	-	-	50.6	3.9	132	71	61
B			3600	-	-	163	1.7	457	282	175
C			3600	-	-	81.9	1.5	484	335	149

Cuadro 4.2: Resultados computacionales con $\alpha = 0{,}1$

Como vemos en la tabla 4.2, ocurre lo mismo que en la tabla anterior a la hora de comparar un modelo con otro, en lo que respecta a tiempos y costes. El segundo caso sigue siendo mucho más rápido.

La diferencia que se ve con el caso $\alpha = 0{,}3$ es que, para el mismo conjunto de clientes, el tiempo computacional es menor y el coste total mayor. Esto es debido a que el beneficio que se obtiene con $\alpha = 0{,}1$ es mucho más inferior que en el caso previo, es decir, se visitan menos clientes que no estamos obligados a visitar ya que nos aportan un beneficio menor.

En conclusión, el modelo de rutas es muy rápido, aunque por lo general da peores resultados. Esto se produce ya que no están implementadas las rutas que contienen cinco o más clientes con Visual Basic. Al implementarlo se conseguirían los mismos resultados que con el modelo de arcos y en tiempos muy pequeños.

Es importante destacar que cuando h_i es grande los tiempos computacionales se disparan con el modelo de arcos. Se aprecia claramente en la tabla 4.1, en el caso de 4 clientes con $h_i = 0$ y 12 con $h_i = 1$.

Capítulo 5

Conclusiones

A lo largo de esta memoria se ha modelado un problema de rutas que se le plantea a la empresa Biocontrol Canarias S.L. Anteriormente, se realizó con un "software" de pago (XPRESS). Ahora se ha utilizado un "software" libre, o de bajo coste (Excel), lo cual ha supuesto muchas dificultades.

Primero se intentó modelar con Microsoft Solver Foundation, ya que en principio parecía bastante cómodo al leer los parámetros y variables utilizando las propias celdas de Excel. Se empezó trabajando con ejemplos pequeños y poco a poco aumentando las restricciones. Finalmente, como ya se comentó, aparecieron problemas a la hora de trabajar con tres índices.

La segunda y última opción fue la herramienta Gusek. Era importante que este resolutor nos leyera los datos de Excel. Según se iba avanzando con el modelo, ya se veía que se iba a tardar demasiado en obtener resultados y se empezó a pensar en un nuevo modelo mas rápido.

Para este nuevo modelo de rutas, se utilizó Visual Basic para Aplicaciones. No dejaron de aparecer nuevos inconvenientes, ya que la cantidad de datos que se obtenían eran demasiado grandes para ser escritos en una hoja de Excel y el código no fue sencillo de implementar. Tras varios días se obtuvo el código que aquí presentamos, un código bastante denso que nos permite obtener un número adecuado de rutas, impresas en un fichero, que Gusek lee y resuelve sin problemas de forma rápida.

Sería interesante en un futuro ampliar el código de Visual Basic para que pueda generar rutas de cinco clientes, ya que con el primer modelo de arcos sí se obtienen rutas con cinco clientes y unas soluciones buenas pero empleando mucho tiempo. Sin embargo, si esto estuviera implementado obtendríamos buenos resultados en poco tiempo.

Otro futuro estudio podría ser intentar programar el modelo propuesto en la sección 3.4, ampliando el rango de visita y de esta manera planear a un año vista las visitas de Biocontrol Canarias S.L.

Apéndice A

Códigos con Gusek

A.1. Modelo variables arcos

```
###############################################################################
# Problema.mod
###############################################################################
#
# Alfonso Ruigómez González
#
#
# Resolutor problema de rutas Biocontrol Canarias S.L.
#
###############################################################################

/* Parámetros y variables */

param n, integer, >= 3;
/* number of nodes */


set DAYS := 1..5;
set CITIES;
set ARCS within CITIES cross CITIES ;
param DIST{(i,j) in ARCS};
param d{CITIES,DAYS}, binary;
param p{CITIES};
param a{CITIES};
param b{CITIES};
param h{CITIES};
param alfa := 0.3;
set visit within CITIES cross DAYS;
param depot := 1;

param bigM{(i,j) in ARCS}, default 2000;

/* Cargar datos del Excel */

table TEST1 IN 'ODBC'
```

```
'DRIVER={Microsoft Excel Driver (*.xls)};dbq=.\datosVBA.xls'
'SELECT * FROM [tiempo$] WHERE prioridad < 3': CITIES <- [city], a ~ arrive, b ~ leave,
p ~ time, h ~ prioridad;

table TEST2 IN 'ODBC'
'DRIVER={Microsoft Excel Driver (*.xls)};dbq=.\datosVBA.xls'
'SELECT * FROM [distance$] WHERE prioridad < 3': ARCS <- [city1,city2], DIST ~ distances,
bigM ~ bigM;

table TEST3 IN 'ODBC'
'DRIVER={Microsoft Excel Driver (*.xls)};dbq=.\datosVBA.xls'
'SELECT * FROM [visitas$] WHERE prioridad < 3': visit <- [city,dia], d ~ visita;

param NEXTC, integer, >= 0;
/* Next city after i in the solution */
var x{(i,j) in ARCS, k in DAYS}, binary;
/* x[i,j] = 1 means that the salesman goes from node i to node j */


param BEN{i in CITIES} := if (i = depot) then 0 else alfa*(3-h[i])*(DIST[depot,i]+DIST[i,depot]);

var t{CITIES, k in DAYS}, >= 0;

var y{i in CITIES, k in DAYS}, binary;

/*MODELO*/

minimize total: sum{(i,j) in ARCS, k in DAYS} DIST[i,j] * x[i,j,k]
- sum{i in CITIES, k in DAYS} BEN[i] * y[i,k];

/*Restricciones */

s.t. leave{i in CITIES, k in DAYS}: sum{(i,j) in ARCS} x[i,j,k] = y[i,k];
/* the salesman leaves each node i exactly once */

s.t. enter{j in CITIES, k in DAYS}: sum{(i,j) in ARCS} x[i,j,k] = y[j,k];
/* the salesman enters each node j exactly once */

s.t. arrive{i in CITIES, k in DAYS}: t[i,k] >= a[i];

s.t. leave2{i in CITIES, k in DAYS}: t[i,k] <= b[i];

s.t. time{(i,j) in ARCS, k in DAYS: i != depot}: t[i,k] + p[i] + DIST[i,j]
- bigM[i,j]*(1- x[i,j,k]) <= t[j,k] ;

s.t. days{i in CITIES, k in DAYS}: y[i,k]<=d[i,k];

s.t. day_used{i in CITIES, k in DAYS: i != depot}:  y[depot,k] >= y[i,k];

s.t. priority0{i in CITIES: h[i] = 0 and i != depot}: sum{k in DAYS} y[i,k] = 1;

s.t. priority1{i in CITIES: h[i] = 1}: sum{k in DAYS} y[i,k] <= 1;

s.t. priority2{i in CITIES: h[i] = 2}: sum{k in DAYS} y[i,k] <= 1;
```

```
#s.t. priority3{i in CITIES, k in DAYS: h[i] = 3}: y[i,k] = 0;

/*
printf                " i   a_i   b_i  p_i h_i ben[i]\n";
printf{ i in CITIES } "%2d  %4d  %4d  %3d %3d %5d\n", i, a[i], b[i], p[i], h[i], BEN[i];

printf                " i  j dist bigM\n";
printf{ (i,j) in ARCS } "%2d %2d %4d %4d\n", i, j, DIST[i,j], bigM[i,j];

printf                " i day visita\n";
printf{ i in CITIES, k in DAYS } "%2d %3d %6d\n", i, k, d[i,k];
*/
solve;

/* Imprime la solución */

printf "Optimal tour has length %d\n",
sum{(i,j) in ARCS, k in DAYS} DIST[i,j] * x[i,j,k]+ sum{i in CITIES,k in DAYS} p[i]*y[i,k];
printf("Day  From    To  Arrive  Distance Process\n");

printf{k in DAYS, (i,j) in ARCS: x[i,j,k]} "%3d%6d%6d%8g%10g%8g\n", k, i, j, t[j,k], DIST[i,j], p[i];


printf("cus vist\n");
printf{i in CITIES} "%3d%3d\n", i, sum{k in DAYS} y[i,k];



/*Muestra la solución en otra hoja del Excel */

table solution {k in DAYS, (i,j) in ARCS: x[i,j,k]} OUT 'ODBC'
'DRIVER={Microsoft Excel Driver (*.xls)};READONLY=FALSE;dbq=.\datosVBA.xls'
'INSERT INTO [solution$] VALUES (?,?,?,?,?,?)':
k ~ Day, i ~ From, j ~ Destination, t[j,k] ~ Arrive, DIST[i,j] ~ Distance, p[i] ~ Process;

data;


end;
```

A.2. Modelo variables rutas

```
###############################################################################
# Problema.mod
###############################################################################
#
# Alfonso Ruigómez González
#
#
# Resolutor problema de rutas Biocontrol Canarias S.L.
#
###############################################################################


/* Parámetros y variables */

param n, integer, >= 3;
/* number of nodes */

set CITIES;
set DAYS := 1..5;
set RUTAS;


set RUTAS_DIAS within RUTAS cross DAYS;
set RUTAS_CITIES within RUTAS cross CITIES;

param c{RUTAS};
#param m{RUTAS,CITIES}, binary;
param d{RUTAS,DAYS}, binary;
param rc{RUTAS,CITIES}, integer, default 0;
param h{CITIES};
param dist_depot{CITIES};
param depot := 1;

param alfa := 0.3;

/* Cargar datos del Excel */

table TEST1 IN 'ODBC'
'DRIVER={Microsoft Excel Driver (*.xls)};dbq=.\datosVBA.xlsm'
'SELECT * FROM [tiempo$] WHERE prioridad < 3': CITIES <- [city], h ~ prioridad, dist_depot ~ dist_depot;


table TEST2 IN "CSV" "rutas_costos.csv" : RUTAS <- [id], c ~ Costo;

table TEST3 IN "CSV" "rutas_clientes.csv" : RUTAS_CITIES <- [id,cliente], rc ~ visit;

table TEST4 IN "CSV" "rutas_dias.csv" : RUTAS_DIAS <- [id,dia], d ~ visit;

param BEN{i in CITIES} := alfa*(3-h[i])*dist_depot[i];

#display c;
```

```
#display rc;
#display d;

#table TEST4 IN "CSV" "rutas_dias.csv" : ;

var x{r in RUTAS, k in DAYS}, binary;

/*MODELO*/

minimize total:   sum{r in RUTAS, k in DAYS} x[r,k] * c[r]
- sum{i in CITIES, r in RUTAS, k in DAYS} BEN[i] * rc[r,i] * x[r,k];

/*Restricciones */

s.t. restr1{i in CITIES: h[i] = 0 and i != depot}: sum{r in RUTAS, k in DAYS} rc[r,i] * x[r,k] = 1;

s.t. restr2{i in CITIES: h[i] = 1}: sum{r in RUTAS, k in DAYS} rc[r,i] * x[r,k] <= 1;

s.t. restr3{i in CITIES: h[i] = 2}: sum{r in RUTAS, k in DAYS} rc[r,i] * x[r,k] <= 1;

#s.t. restr4{i in CITIES: h[i] = 3}: sum{r in RUTAS, k in DAYS} rc[r,i] * x[r,k] <= 0;

s.t. restr5{r in RUTAS, k in DAYS}: x[r,k] <= d[r,k];

s.t. restr6{k in DAYS}: sum{r in RUTAS} x[r,k] <= 1;

solve;

printf "Optimal tour has length %d\n",
sum{r in RUTAS, k in DAYS} x[r,k] * c[r];


printf("Dia   Ruta  Clientes\n");
for{k in DAYS, r in RUTAS : x[r,k]} {

printf " %d  %d ", k, r;
printf{i in CITIES : rc[r,i]} " %d,", i;
printf "\n";
}



data;


end;
```

Apéndice B

Códigos con Microsoft Solver Foundation y Visual Basic para Aplicaciones

B.1. MSF

```
/##########################################################################################
# problema.xlsx
##########################################################################################
#
# Alfonso Ruigómez González
#
#
#
# Última versión del modelo implementado con Solver para 13 clientes
#
##########################################################################################

Model[
Parameters[
Sets[Integers[0, Infinity]],
Cities = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
],
Parameters[
Sets[Any],
Clients = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13},
Depot = {0},
Days = {1, 2, 3, 4, 5}
],
Parameters[
Integers[-Infinity, Infinity],
dist[Cities, Cities],
dummy[Clients],
dummy2[Depot],
Arrive[Cities],
Leave[Cities],
```

```
P[Cities]
],
Parameters[
Reals[-Infinity, Infinity],
M
],
Parameters[
Booleans,
d[Cities, Days]
],
Decisions[
Reals[0, Infinity],
T[Cities, Days]
],
Decisions[
Integers[0, 1],
x[Cities, Cities, Days],
y[Cities, Days]
],
Constraints[
Constraint1 -> Foreach[{i,Cities},{k,Days},FilteredSum[{j,Cities},dist[i,j]>0, x[i,j,k]] == y[i,k]],
Constraint2 -> Foreach[{i,Cities},{k,Days},FilteredSum[{j,Cities},dist[i,j]>0, x[j,i,k]] == y[i,k]],
Constraint3 -> Foreach[{i,Cities},{j,Clients},{k,Days},T[i,k]+P[i]+dist[i,j]-T[j,k] <= M * (1-x[i,j,k])],
Constraint4 -> Foreach[{i,Cities},{k,Days}, Arrive[i] <= T[i,k]],
Constraint5 -> Foreach[{i,Cities},{k,Days}, T[i,k] <= Leave[i]],
Constraint6 -> Foreach[{i,Clients},{k,Days},y[0,k]>=y[i,k]],
Constraint7 -> Foreach[{i,Clients},Sum[{k,Days}, y[i,k] == 1]],
Constraint8 -> Foreach[{i,Cities},{k,Days},y[i,k]<=d[i,k]]
],
Goals[
Minimize[
Goal1 -> Annotation[FilteredSum[{i,Cities},{j,Cities},{k,Days}, dist[i,j]>0,dist[i,j]*x[i,j,k]],
"order", 0]
]
]
]
```

B.2. VBA

```
Const INFINITE_COST As Long = 2000
Const MAX_N As Long = 150
Const MAX_RUTAS As Long = 200000
Const MAX_CLIENTES_EN_RUTA As Integer = 4
Const MIN_NUM_RUTAS_CLIENTE As Integer = 100
Const DIR As String = "C:\Users\alfonso\Desktop\GusekVBAFinal\"


Dim d(1 To MAX_N, 1 To 5) As Long
Dim Zona(1 To MAX_N) As Long
Dim p(1 To MAX_N) As Long
Dim b(1 To MAX_N) As Long
Dim a(1 To MAX_N) As Long
Dim h(1 To MAX_N) As Long
Dim Dist(1 To MAX_N, 1 To MAX_N) As Long

' Almacenamos en memoria los costos y ciudades que visita cada ruta
Dim CostoRuta(1 To MAX_RUTAS) As Integer
Dim CitiesRutas(1 To MAX_RUTAS, 1 To MAX_CLIENTES_EN_RUTA) As Integer
' Utilizo eun array para ver si hay que borrarlo o no
Dim Borrar(1 To MAX_RUTAS) As Boolean




Sub prob()
Dim n As Long
n = 125

' Cargamos las distancias
Dim i, j, k, l, z As Long
For i = 1 To n
For j = 1 To n
Dist(i, j) = Worksheets("matrix_dist").Cells(i, j).Value
Next j
Next i


' Cargamos las ventanas de tiempo
For i = 1 To n
a(i) = Worksheets("tiempo").Cells(i + 1, 2).Value
Next i
For i = 1 To n
b(i) = Worksheets("tiempo").Cells(i + 1, 3).Value
Next i

' Cargamos los tiempos de proceso
For i = 1 To n
p(i) = Worksheets("tiempo").Cells(i + 1, 4).Value
Next i
```

```
' Cargamos las prioridades
For i = 1 To n
h(i) = Worksheets("tiempo").Cells(i + 1, 5).Value
Next i

For i = 1 To n
For k = 1 To 5
d(i, k) = Worksheets("dias").Cells(i, k).Value
Next k
Next i


For i = 1 To n
Zona(i) = Worksheets("tiempo").Cells(i + 1, 6).Value
Next i

Dim var_count As Long
Dim aux_count As Long

var_count = 0

aux_count = BuscaVariables1Clientes(n, var_count)
For z = 2 To 5
' Cuando todos los clientes son de la zona 1 se buscan con la zona 2
aux_count = BuscaVariables2ClientesPorZona(z, n, var_count)
aux_count = BuscaVariables3ClientesPorZona(z, n, var_count)
aux_count = BuscaVariables4ClientesPorZona(z, n, var_count)

'MsgBox ("Se han generado un total de " & var_count & " rutas")

'Call Borrar_Rutas_Mayor_Costo(var_count, n)

'MsgBox ("Despues del borrado de rutas hay un total de " & var_count & " rutas")
Next z


'MsgBox ("Se han generado un total de " & var_count & " rutas")

'Call Borrar_Rutas_Mayor_Costo(var_count, n)

MsgBox ("Despues del borrado de rutas hay un total de " & var_count & " rutas")


' Escribimos las rutas en fichero
Call ImprimirRutasFicheros(var_count)

End Sub

' Busca rutas que contienen a tres clientes entre la zona 1 y la zona z (<> 1)
' Si z=2 entonces los 4 clientes pueden ser de la zona 1
Public Function BuscaVariables4ClientesPorZona(ByVal z As Long, ByVal n As Long,
ByRef count As Long) As Long
Dim i, j, l, m As Long
Dim Ct As Long
```

```
Dim orden As Long
Dim city1 As Long
Dim city2 As Long
Dim city3 As Long
Dim city4 As Long

city1 = 1: city2 = 1: city3 = 1: city4 = 1
Dim first_count As Long
first_count = count

For i = 2 To n
If ((Zona(i) = 1 Or Zona(i) = z) And h(i) < 3) Then
For j = i + 1 To n
If ((Zona(j) = 1 Or Zona(j) = z) And h(j) < 3) Then
For l = j + 1 To n
If ((Zona(l) = 1 Or Zona(l) = z) And h(l) < 3) Then
For m = l + 1 To n
If ((Zona(m) = 1 Or Zona(m) = z) And h(m) < 3) Then

' En el caso de que las 3 ciudades sean de la zona 1 y z es distinto de 2
' no añadimos la ruta ya que se supone que fue añadida antes
If (z = 2 Or Zona(i) > 1 Or Zona(j) > 1 Or Zona(l) > 1 Or Zona(m) > 1) Then
If (FactibleDias(i, j, l, 1) = True) Then
Ct = Costo4ClientesPosibilidades(i, j, l, m, city1, city2, city3, city4)
If (Ct < INFINITE_COST) Then
count = count + 1
CostoRuta(count) = Ct
Call AlmacenaRuta(n, count, Ct, city1, city2, city3, city4)
'Call ImprimirRuta(n, count, Ct, city1, city2, city3, 0)
End If
End If
End If
End If
Next m
End If
Next l
End If
Next j
End If
Next i
BuscaVariables4ClientesPorZona = count - first_count
End Function


' Busca rutas que contienen a tres clientes entre la zona 1 y la zona z (<> 1)
' Si z=2 entonces los 3 clientes pueden ser de la zona 1
Public Function BuscaVariables3ClientesPorZona(ByVal z As Long,
ByVal n As Long, ByRef count As Long) As Long
Dim i, j, l As Long
Dim Ct As Long
Dim orden As Long
Dim city1 As Long
Dim city2 As Long
Dim city3 As Long
```

```
city1 = 1
city2 = 1
city3 = 1
Dim first_count As Long
first_count = count

For i = 2 To n
If ((Zona(i) = 1 Or Zona(i) = z) And h(i) < 3) Then
For j = i + 1 To n
If ((Zona(j) = 1 Or Zona(j) = z) And h(j) < 3) Then
For l = j + 1 To n
If ((Zona(l) = 1 Or Zona(l) = z) And h(l) < 3) Then
' En el caso de que las 3 ciudades sean de la zona 1 y z es distinto de 2
' no añadimos la ruta ya que se supone que fue añadida antes
If (z = 2 Or Zona(i) > 1 Or Zona(j) > 1 Or Zona(l) > 1) Then
If (FactibleDias(i, j, l, 1) = True) Then
Ct = Costo3ClientesPosibilidades(i, j, l, city1, city2, city3)
If (Ct < INFINITE_COST) Then
count = count + 1
CostoRuta(count) = Ct
Call AlmacenaRuta(n, count, Ct, city1, city2, city3, 0)
'Call ImprimirRuta(n, count, Ct, city1, city2, city3, 0)
End If
End If
End If
End If
Next l
End If
Next j
End If
Next i
BuscaVariables3ClientesPorZona = count - first_count
End Function

Public Function BuscaVariables2ClientesPorZona(ByVal z As Long, ByVal n As Long,
ByRef count As Long) As Long
Dim i, j As Long
Dim Ct As Long
Dim orden As Long
Dim city1 As Long
Dim city2 As Long

city1 = 1
city2 = 1
Dim first_count As Long
first_count = count

For i = 2 To n
If ((Zona(i) = 1 Or Zona(i) = z) And h(i) < 3) Then
For j = i + 1 To n
If ((Zona(j) = 1 Or Zona(j) = z) And h(j) < 3) Then
' En el caso de que las 2 ciudades sean de la zona 1 y z es distinto de 2
' no añadimos la ruta ya que se supone que fue añadida antes
```

```
If (z = 2 Or Zona(i) > 1 Or Zona(j) > 1) Then
If (FactibleDias(i, j, 1, 1) = True) Then
Ct = Costo2ClientesPosibilidades(i, j, city1, city2)
If (Ct < INFINITE_COST) Then
count = count + 1
CostoRuta(count) = Ct
Call AlmacenaRuta(n, count, Ct, city1, city2, 0, 0)
'Call ImprimirRuta(n, count, Ct, city1, city2, 0, 0)
End If
End If
End If
End If
Next j
End If
Next i
BuscaVariables2ClientesPorZona = count - first_count
End Function


Public Function BuscaVariables1Clientes(ByVal n As Long, ByRef count As Long) As Long
Dim i, j As Long
Dim Ct As Long
Dim orden As Long
Dim city1 As Long

Dim first_count As Long
first_count = count

For i = 2 To n
If (h(i) < 3) Then
Ct = Costo1Clientes(i)
count = count + 1
CostoRuta(count) = Ct
Call AlmacenaRuta(n, count, Ct, i, 0, 0, 0)
End If
Next i
BuscaVariables1Clientes = count - first_count
End Function




Public Function Costo3ClientesPosibilidades(ByVal i As Integer, ByVal j As Integer, ByVal l As Integer, _
city1 As Long, city2 As Long, city3 As Long) As Long
Dim Ct, BestCt As Long ' Coste de tiempos de viaje y espera

BestCt = INFINITE_COST
Ct = Costo3Clientes(i, j, l)
If (Ct < BestCt) Then
BestCt = Ct
city1 = i
city2 = j
city3 = l
```

```
End If
Ct = Costo3Clientes(i, l, j)
If (Ct < BestCt) Then
BestCt = Ct
city1 = i
city2 = l
city3 = j
End If
Ct = Costo3Clientes(j, i, l)
If (Ct < BestCt) Then
BestCt = Ct
city1 = j
city2 = i
city3 = l
End If
Ct = Costo3Clientes(j, l, i)
If (Ct < BestCt) Then
BestCt = Ct
city1 = j
city2 = l
city3 = i
End If
Ct = Costo3Clientes(l, i, j)
If (Ct < BestCt) Then
BestCt = Ct
city1 = l
city2 = i
city3 = j
End If
Ct = Costo3Clientes(l, j, i)
If (Ct < BestCt) Then
BestCt = Ct
city1 = l
city2 = j
city3 = i
End If
Costo3ClientesPosibilidades = BestCt
End Function


Public Function Costo4ClientesPosibilidades(ByVal i As Integer, ByVal j As Integer, ByVal l As Integer,
ByVal m As Integer,
city1 As Long, city2 As Long, city3 As Long, city4 As Long) As Long
Dim Ct, BestCt As Long ' Coste de tiempos de viaje y espera


BestCt = INFINITE_COST
' ponemos las 4!= 24 formas de recorrer los 4 clientes
Ct = Costo4Clientes(i, j, l, m)
If (Ct < BestCt) Then BestCt = Ct: city1 = i: city2 = j: city3 = l: city4 = m
Ct = Costo4Clientes(i, j, m, l)
If (Ct < BestCt) Then BestCt = Ct: city1 = i: city2 = j: city3 = m: city4 = l
Ct = Costo4Clientes(i, l, j, m)
If (Ct < BestCt) Then BestCt = Ct: city1 = i: city2 = l: city3 = j: city4 = m
```

```
Ct = Costo4Clientes(i, l, m, j)
If (Ct < BestCt) Then BestCt = Ct: city1 = i: city2 = l: city3 = m: city4 = j
Ct = Costo4Clientes(i, m, j, l)
If (Ct < BestCt) Then BestCt = Ct: city1 = i: city2 = m: city3 = j: city4 = l
Ct = Costo4Clientes(i, m, l, j)
If (Ct < BestCt) Then BestCt = Ct: city1 = i: city2 = m: city3 = l: city4 = j

Ct = Costo4Clientes(j, i, l, m)
If (Ct < BestCt) Then BestCt = Ct: city1 = j: city2 = i: city3 = l: city4 = m
Ct = Costo4Clientes(j, i, m, l)
If (Ct < BestCt) Then BestCt = Ct: city1 = j: city2 = i: city3 = m: city4 = l
Ct = Costo4Clientes(j, l, i, m)
If (Ct < BestCt) Then BestCt = Ct: city1 = j: city2 = l: city3 = i: city4 = m
Ct = Costo4Clientes(j, l, m, i)
If (Ct < BestCt) Then BestCt = Ct: city1 = j: city2 = l: city3 = m: city4 = i
Ct = Costo4Clientes(j, m, i, l)
If (Ct < BestCt) Then BestCt = Ct: city1 = j: city2 = m: city3 = i: city4 = l
Ct = Costo4Clientes(j, m, l, i)
If (Ct < BestCt) Then BestCt = Ct: city1 = j: city2 = m: city3 = l: city4 = i

Ct = Costo4Clientes(l, i, j, m)
If (Ct < BestCt) Then BestCt = Ct: city1 = l: city2 = i: city3 = j: city4 = m
Ct = Costo4Clientes(l, i, m, j)
If (Ct < BestCt) Then BestCt = Ct: city1 = l: city2 = i: city3 = m: city4 = j
Ct = Costo4Clientes(l, j, i, m)
If (Ct < BestCt) Then BestCt = Ct: city1 = l: city2 = j: city3 = i: city4 = m
Ct = Costo4Clientes(l, j, m, i)
If (Ct < BestCt) Then BestCt = Ct: city1 = l: city2 = j: city3 = m: city4 = i
Ct = Costo4Clientes(l, m, i, j)
If (Ct < BestCt) Then BestCt = Ct: city1 = l: city2 = m: city3 = i: city4 = j
Ct = Costo4Clientes(l, m, j, i)
If (Ct < BestCt) Then BestCt = Ct: city1 = l: city2 = m: city3 = j: city4 = i

Ct = Costo4Clientes(m, i, j, l)
If (Ct < BestCt) Then BestCt = Ct: city1 = m: city2 = i: city3 = j: city4 = l
Ct = Costo4Clientes(m, i, l, j)
If (Ct < BestCt) Then BestCt = Ct: city1 = m: city2 = i: city3 = l: city4 = j
Ct = Costo4Clientes(m, j, i, l)
If (Ct < BestCt) Then BestCt = Ct: city1 = m: city2 = j: city3 = i: city4 = l
Ct = Costo4Clientes(m, j, l, i)
If (Ct < BestCt) Then BestCt = Ct: city1 = m: city2 = j: city3 = l: city4 = i
Ct = Costo4Clientes(m, l, i, j)
If (Ct < BestCt) Then BestCt = Ct: city1 = m: city2 = l: city3 = i: city4 = j
Ct = Costo4Clientes(m, l, j, i)
If (Ct < BestCt) Then BestCt = Ct: city1 = m: city2 = l: city3 = j: city4 = i

Costo4ClientesPosibilidades = BestCt
End Function
```

```
Public Function Costo2ClientesPosibilidades(ByVal i As Integer, ByVal j As Integer, city1 As Long,
```

```
 city2 As Long) As Long
Dim Ct, BestCt As Long ' Coste de tiempos de viaje y espera
'Dim city1, city2, city3 As Long

BestCt = INFINITE_COST
Ct = Costo2Clientes(i, j)
If (Ct < BestCt) Then
BestCt = Ct
city1 = i
city2 = j
End If
Ct = Costo2Clientes(j, i)
If (Ct < BestCt) Then
BestCt = Ct
city1 = j
city2 = i
End If
Costo2ClientesPosibilidades = BestCt
End Function

Public Function Costo1Clientes(ByVal city1 As Long) As Long
Dim Ct As Long ' Coste de tiempos de viaje y espera
Dim t As Long ' Tiempo
' Tiempo en llegar a city2
Costo1Clientes = INFINITE_COST
Ct = 0
t = 0
Call AñadirCliente(1, city1, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function
Call AñadirCliente(city1, 1, t, Ct)
Costo1Clientes = Ct
End Function


Public Function Costo2Clientes(ByVal city1 As Long, ByVal city2 As Long) As Long
Dim Ct As Long ' Coste de tiempos de viaje y espera
Dim t As Long ' Tiempo
' Tiempo en llegar a city2
Costo2Clientes = INFINITE_COST
Ct = 0
t = 0
Call AñadirCliente(1, city1, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function
Call AñadirCliente(city1, city2, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function
Call AñadirCliente(city2, 1, t, Ct)
Costo2Clientes = Ct
End Function


Public Function Costo3Clientes(ByVal city1 As Long, ByVal city2 As Long, ByVal city3 As Long) As Long
Dim Ct As Long ' Coste de tiempos de viaje y espera
Dim t As Long ' Tiempo
' Tiempo en llegar a city2
```

```
Costo3Clientes = INFINITE_COST
Ct = 0
t = 0
Call AñadirCliente(1, city1, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city1, city2, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city2, city3, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city3, 1, t, Ct)
Costo3Clientes = Ct
End Function

Public Function Costo4Clientes(ByVal city1 As Long, ByVal city2 As Long, ByVal city3 As Long,
ByVal city4 As Long) As Long
Dim Ct As Long ' Coste de tiempos de viaje y espera
Dim t As Long ' Tiempo
' Tiempo en llegar a city2
Costo4Clientes = INFINITE_COST
Ct = 0
t = 0
Call AñadirCliente(1, city1, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city1, city2, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city2, city3, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city3, city4, t, Ct)
If (Ct = INFINITE_COST) Then Exit Function

Call AñadirCliente(city4, 1, t, Ct)
Costo4Clientes = Ct
End Function


' Modificación de los parámetros tiempo y costo al añadir una nuevo cliente
' Se supone que los datos son cuando se empieza a servir al cliente
(se ha esperado para la entrada pero no se ha procesado)
Public Sub AñadirCliente(ByVal prev_city As Long, ByVal next_city As Long, ByRef t As Long,
ByRef Ct As Long)
Dim Ce As Long 'Tiempo de espera

t = t + p(prev_city) + Dist(prev_city, next_city)
If (t > b(next_city)) Then
Ct = INFINITE_COST
Exit Sub
End If
Ce = 0
```

```
If (t < a(next_city)) Then
Ce = a(next_city) - t
t = a(next_city)
End If
Ct = Ct + Dist(prev_city, next_city)

' Consideramos los costos de espera con el mismo peso que los costos de desplazamiento

'If (prev_city <> 1) Then Ct = Ct + Ce

End Sub



Public Function FactibleDias(ByVal i As Long, ByVal j As Long, ByVal l As Long, ByVal m As Long) As Boolean

FactibleDias = False
For k = 1 To 5
If (FactibleDia(i, j, l, m, k) = True) Then
FactibleDias = True
Exit Function
End If
Next k
End Function


Public Function FactibleDia(ByVal i As Long, ByVal j As Long, ByVal l As Long, ByVal m As Long, _
ByVal k As Long) As Boolean

FactibleDia = False
If (d(i, k) * d(j, k) * d(l, k) * d(m, k) >= 1) Then
FactibleDia = True
End If
End Function


Public Function FactibleDiaRuta(ByVal ruta As Long, ByVal dia As Long) As Boolean

Dim val As Integer
Dim num_clientes As Integer
val = 1
num_clientes = CalculaNumeroClientes(ruta)
For k = 1 To num_clientes
If (d(CitiesRutas(ruta, k), dia) < 1) Then
FactibleDiaRuta = False
Exit Function
End If
Next k
FactibleDiaRuta = True
End Function



Public Sub AlmacenaRuta(ByVal n As Long, ByVal count As Long, ByVal coste As Long, ByVal city1 As Long,
```

```
 ByVal city2 As Long, ByVal city3 As Long, ByVal city4 As Long)

CostoRuta(count) = coste
CitiesRutas(count, 1) = city1
CitiesRutas(count, 2) = city2
CitiesRutas(count, 3) = city3
CitiesRutas(count, 4) = city4
End Sub



Public Sub ImprimirRuta(ByVal n As Long, ByVal count As Long, ByVal coste As Long,
ByVal city1 As Long, ByVal city2 As Long, ByVal city3 As Long, ByVal city4 As Long)

Dim k As Integer
Dim i As Integer

Worksheets("sol").Cells(count + 1, 1).Value = count
Worksheets("sol").Cells(count + 1, 2).Value = coste
Worksheets("sol").Cells(count + 1, 3).Value = city1
Worksheets("sol").Cells(count + 1, 4).Value = city2
Worksheets("sol").Cells(count + 1, 5).Value = city3
Worksheets("sol").Cells(count + 1, 6).Value = city4
For k = 1 To 5
If FactibleDia(city1, city2, city3, 1, k) = True Then
Worksheets("sol").Cells(count + 1, 6 + k).Value = 1
Else
Worksheets("sol").Cells(count + 1, 6 + k).Value = 0
End If
Next k

End Sub

Public Sub ImprimirRutasFicheros(ByVal num_rutas As Long)

Dim k As Integer
Dim ruta As Long

Open DIR & "rutas_costos.csv" For Output As #1
Open DIR & "rutas_clientes.csv" For Output As #2
Open DIR & "rutas_dias.csv" For Output As #3
Print #1, "id,Costo"
Print #2, "id,cliente,visit"
Print #3, "id,dia,visit"

For ruta = 1 To num_rutas
' Imprimimos en el fichero de rutas
Print #1, ruta & "," & CostoRuta(ruta)           'Con punto y coma al final no imprime el salto de línea

' Imprimimos en el fichero de Rutas Clientes
Print #2, ruta & ",1,1"
For k = 1 To MAX_CLIENTES_EN_RUTA
If CitiesRutas(ruta, k) <> 0 Then
Print #2, ruta & "," & CitiesRutas(ruta, k) & ",1"
```

```
End If
Next k

' Imprimimos el fichero de rutas_dias
For k = 1 To 5
If FactibleDiaRuta(ruta, k) Then
Print #3, ruta & "," & k & ",1"
Else
Print #3, ruta & "," & k & ",0"
End If

Next k

Next ruta

Close #3
Close #2
Close #1

End Sub

Public Sub Borrar_Rutas_Mayor_Costo(ByRef num_rutas As Long, ByVal n As Long)

Dim ruta As Long
Dim i, j, k As Long
Dim num_clientes_ruta As Integer
Dim ver As Boolean
Dim minimos(1 To MAX_N, 1 To MIN_NUM_RUTAS_CLIENTE) As Integer
Dim nuevo_num_rutas As Long


For i = 1 To n
For j = 1 To MIN_NUM_RUTAS_CLIENTE
minimos(i, j) = INFINITE_COST
Next j
Next i

' Buscamos los mejores costos en relación cun el número de clientes
For ruta = 1 To num_rutas
num_clientes_ruta = CalculaNumeroClientes(ruta)
ver = True
For k = 1 To num_clientes_ruta
i = CitiesRutas(ruta, k)
If h(i) = 3 Then
ver = False
Exit For
End If
Next k
If (ver) Then
For k = 1 To num_clientes_ruta
i = CitiesRutas(ruta, k)
If (CostoRuta(ruta) / num_clientes_ruta < minimos(i, MIN_NUM_RUTAS_CLIENTE)) Then
j = MIN_NUM_RUTAS_CLIENTE - 1
Do While (j > 0)
```

```
If CostoRuta(ruta) / num_clientes_ruta >= minimos(i, j) Then Exit Do
minimos(i, j + 1) = minimos(i, j)
j = j - 1
Loop
minimos(i, j + 1) = CostoRuta(ruta) / num_clientes_ruta
End If
Next k
End If
Next ruta

' Marcamos las rutas que hay que borrar
For ruta = 1 To num_rutas
Borrar(ruta) = True
Next ruta

For ruta = 1 To num_rutas
num_clientes_ruta = CalculaNumeroClientes(ruta)
ver = True
For k = 1 To num_clientes_ruta
i = CitiesRutas(ruta, k)
If h(i) = 3 Then
ver = False
Exit For
End If
Next k
If (ver) Then
For k = 1 To num_clientes_ruta
i = CitiesRutas(ruta, k)
If (CostoRuta(ruta) / num_clientes_ruta <= minimos(i, MIN_NUM_RUTAS_CLIENTE)) Then
Borrar(ruta) = False
End If
Next k
Else
Borrar(ruta) = True
End If
Next ruta

' Ahora sobreescribimos las que queremos borrar
nuevo_num_rutas = 0
ruta = 1


For ruta = 1 To num_rutas
If Borrar(ruta) = True Then Exit For
Next ruta

'Do While Borrar(ruta) = False
'    ruta = ruta + 1
'Loop

nuevo_num_rutas = ruta - 1

For ruta = nuevo_num_rutas + 1 To num_rutas
If Borrar(ruta) = False Then
```

```
nuevo_num_rutas = nuevo_num_rutas + 1
CostoRuta(nuevo_num_rutas) = CostoRuta(ruta)
For k = 1 To MAX_CLIENTES_EN_RUTA
CitiesRutas(nuevo_num_rutas, k) = CitiesRutas(ruta, k)
Next k

End If
Next ruta

num_rutas = nuevo_num_rutas

End Sub

Public Function CalculaNumeroClientes(ByVal ruta As Long)
Dim k As Integer
For k = 1 To MAX_CLIENTES_EN_RUTA
If (CitiesRutas(ruta, k) <= 1) Then
CalculaNumeroClientes = k - 1
Exit Function
End If
Next k
CalculaNumeroClientes = MAX_CLIENTES_EN_RUTA
End Function
```

Bibliografía

[1] Juan José Salazar-González. *Programación Matemática*, Díaz de Santos, 2001.

[2] G. Paletta. *The period traveling salesman problem: a new heuristic algorithm*, Computers Operations Research, September, 2002.

[3] *Modeling Language GNU MathProg*. Language Reference for GLPK Version 4.50, May 2013.

[4] David L. Applegate, Robert E.Bixby, Vasek Chvátal, William J. Cook. *The Traveling Salesman Problem*, Princeton University Press, 2006.

[5] *MSDN Library*. Development Tools and Languages. Microsoft Solver Foundation 3.1.

[6] *MSDN Library*. Herramientas y lenguajes de desarrollo. Herramientas XML en Visual Studio. Visual Basic y Visual C#. Visual Basic

[7] Alfredo Olivera. *Heurísticas para Problemas de Ruteo de Vehículo*, Agosto 2004.

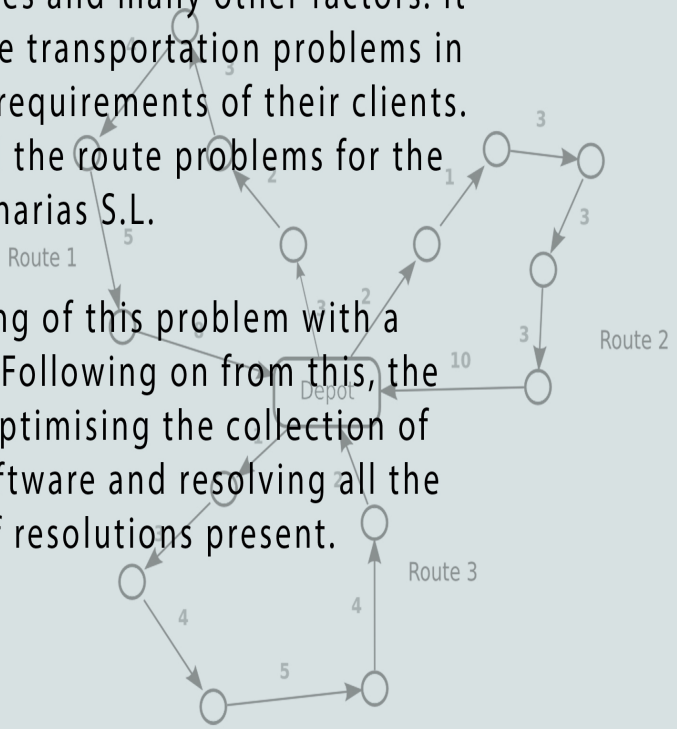
[8] Gregory Gutin, Abraham P. Punnen. *The Traveling Salesman Problem and its variations*, 2002.

[9] María Lorena Stockdale. *El problema del viajante: un algoritmo heurístico y una aplicación*, Noviembre 2011.

[10] Irema Rodríguez-García. *Un problema de rutas con periodicidad y ventanas de tiempo*, Septiembre 2014.

Solving a routing problem with time windows and periodicity

Transport, as well as its costs, are presented within a large number of services. The increase in the price of the fuel plus the progress that is currently being experienced in the electronic field, are turning transport into one of the most significant costs for many companies. One of the main goals of transport logistics is to model in the most optimal way possible a route set, made by a number of vehicles, leaving from a place called deposit. This routes must satisfy the demands of the customers, considering the frequency of visit, schedules and many other factors. It is normal for companies to face these route transportation problems in a manual way, and by trying to satisfy the requirements of their clients. In the current project, it is tried to model the route problems for the company Biocontrol Canarias S.L.



In the past, they achieved the modelling of this problem with a purchased software called Xpress Mosel. Following on from this, the objective today is to model the way of optimising the collection of samples for this company using a free software and resolving all the inconveniences that these types of resolutions present.

Alfonso Ruigómez González

Faculty of Science, Math Section

University of La Laguna